T-Router: Learning Thalamic Routing for Reasoning with Parameter-Efficient Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Parameter-efficient reinforcement learning aims to improve reasoning with a compact trainable interface to a pretrained model. We introduce the *Thalamic Router* (T-Router), which concentrates adaptation on the reuse of completed computations. A compressed, addressable bank preserves block changes; a depth-recurrent controller conditions their selection and relative-scale writeback. This coupling gives thalamic context-dependent routing a concrete computational form: learn which earlier contributions a receiving layer uses, and with what influence. Correctness rewards train the interface while preserving backbone parameters and layer order. On an 8.95B-parameter backbone, T-Router allocates 41.73M parameters—0.466% of the backbone—and achieves 83.64 ± 1.16 MathAvg after GSM8K RL, compared with 73.79 ± 1.83 for full-parameter GRPO across three evaluation rounds. At a comparable parameter budget and with matched retries, it exceeds LoRA's 77.28 ± 1.95 MathAvg, improving all three task families and raising mean AIME accuracy from 48.33 to 60.56. Capacity-controlled comparisons favor addressable block changes and recurrent context; separate search training extends the interface to tool-mediated reasoning. These results establish controlled computation reuse as an effective route to parameter-efficient reasoning reinforcement learning.

1 Introduction

Reinforcement learning (RL) improves mathematical reasoning by rewarding successful solutions (Shao et al., 2024). Making this adaptation efficient requires placing trainable capacity where reward feedback can improve reasoning while preserving the bulk of pretrained parameters (Sidahmed et al., 2024). A model already contains useful transformations; an additional opportunity is to learn how their intermediate contributions are organized. Our starting point is therefore *adaptation efficiency*: train a compact *Thalamic Router* (T-Router) for *controlled reuse of completed computations* around a frozen backbone.

LoRA and adapters place trainable capacity in weight updates or local transformations (Hu et al., 2022; Houlsby et al., 2019). In the resulting residual stream, earlier block contributions continue to mix with subsequent updates. Keeping these contributions separately addressable exposes an additional learning target: which completed changes a later computation reuses, and with what influence. Learned depth connectivity already develops this direction through attention to prior computation, additive delta reuse, and frozen-backbone residual routing (Kimi Team et al., 2026; Luo et al., 2026a; Oldenburg et al., 2026). The question for a compact RL interface is how to couple reusable content, the receiving computation's context, and writeback strength.

Thalamic circuits regulate cortical communication according to task demands. The pulvinar coordinates information transmission between cortical areas through attention-dependent synchronization (Saalmann et al., 2012); mediodorsal thalamic input amplifies functional prefrontal connectivity to sustain task representations (Schmitt et al., 2017). We take these findings as a routing principle: use context to select information sources and regulate their influence on a receiving computation. In T-Router, frozen decoder blocks perform computation, while the complete auxiliary pathway controls which completed block changes a later layer reuses and at what strength. Figure 1 makes this correspondence explicit.

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.

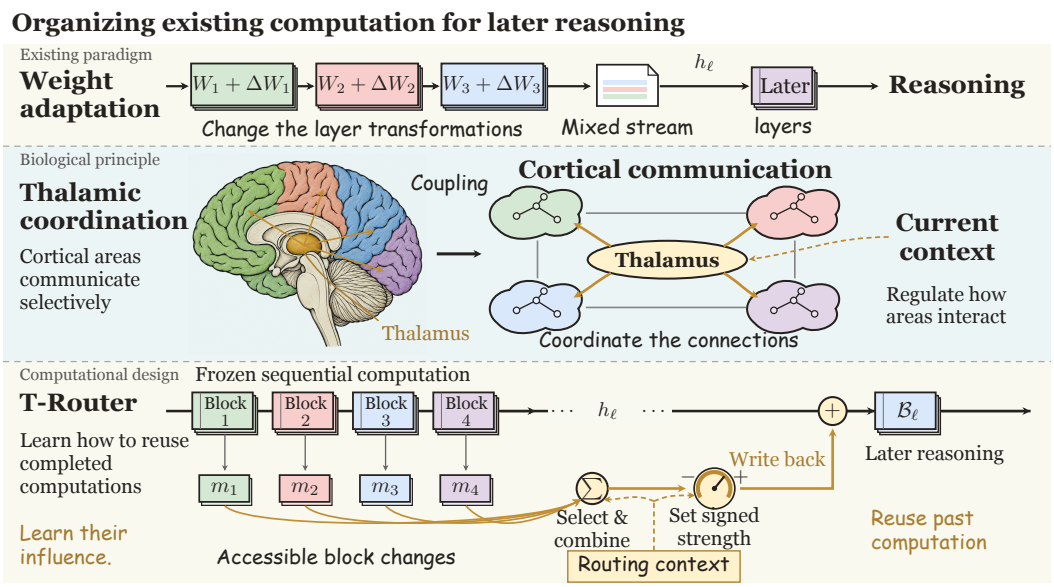


Figure 1: **From adapting transformations to learning their communication.** Weight adaptation changes transformations (top). Thalamic circuits regulate cortical communication according to task context (middle). T-Router applies this routing principle across depth: select addressable block changes and regulate their influence before a later frozen layer executes (bottom).

T-Router preserves compressed block changes in a source bank alongside the evolving residual. Later layers can recombine these records according to their current state and accumulated depth context. Controller slots S accumulate that history; attention queried by the current hidden state reads a context vector P from the updated slots. Together with the hidden state, P conditions both source attention and a signed gate. Selected content is projected into the receiving residual and added before the frozen layer executes. The bank defines *what is available*; the router and gate learn *which mixture to reuse and at what strength*.

Correctness rewards train this communication interface end to end. Its defining feature is the coupling of *addressable content*, *recurrent control*, and *relative influence*: the bank preserves reusable changes, depth context guides their selection, and a signed gate regulates the resulting writeback. RMS calibration expresses the gate's strength relative to the receiving residual. Every pretrained layer still executes in its original order. RL can therefore optimize how completed contributions serve a later computation while preserving the backbone's pretrained transformations.

The mathematics comparison demonstrates this efficiency at two adaptation budgets. With 41.73M parameters, 0.466% of the 8.95B backbone, T-Router reaches 83.64 ± 1.16 MathAvg versus 73.79 ± 1.83 for full-parameter GRPO across three evaluation rounds. At a compact budget, matched-retry LoRA uses 43.28M parameters and reaches 77.28 ± 1.95. T-Router improves all three mathematical families, including 92.73 versus 87.93 on MATH-500 and 60.56 versus 48.33 mean AIME accuracy after GSM8K training. Capacity-controlled comparisons favor the full interface over hidden-state memory (67.46 MathAvg) and a feedforward controller replacement (72.80). Separate search training applies the same compact interface to tool-mediated reasoning.

Our contributions are: (i) a parameter-efficient RL interface coupling compressed, addressable block changes to independent depth-recurrent control and relative-scale writeback; (ii) stronger mathematical reasoning than full-parameter GRPO with 0.466% of the backbone's parameter count, and than matched-retry LoRA at a comparable budget, with an application to agentic search; and (iii) capacity-controlled comparisons and a formal characterization of the content and control pathways. Together they establish controlled computation reuse as a concrete route to efficient RL adaptation.

2 RELATED WORK

Parameter-efficient reinforcement learning. Adapters and LoRA reduce the trainable footprint of task adaptation (Houlsby et al., 2019; Hu et al., 2022); PE-RLHF applies parameter-efficient learning to reward modeling and reinforcement learning (Sidahmed et al., 2024). GRPO uses group-relative rewards (Shao et al., 2024); Section 3 identifies the single-update policy gradient used to train the module. RO-GRPO shapes rewards using routing statistics in LoRA mixtures (Ma et al., 2026), while S-GRPO samples output-token positions for the training loss (Lee & Tong, 2025). DAPO oversamples rollout groups and filters all-correct or all-incorrect groups (Yu et al., 2025); T-Router uses a bounded retry rule, analyzed in Appendix B. T-Router retains input-dependent cross-layer communication at inference; Appendix A details the parameter and execution costs.

Learning communication across depth. DenseNet enables feature reuse through dense concatenation of preceding feature maps (Huang et al., 2017). DenseFormer learns depth-weighted combinations (Pagliardini et al., 2024); MUDDFormer makes cross-layer connections dynamic and stream-specific (Xiao et al., 2025). Attention Residuals introduces attention over earlier computation, including a block variant (Kimi Team et al., 2026). Delta Attention Residuals adds attention-weighted sublayer or block changes to the residual (Luo et al., 2026a), providing the closest source-reuse precedent. Multi-Head Attention Residuals gives feature subspaces separate distributions over depth (Luo et al., 2026b). Hyper-connection constraints and adaptive reassignment further expand depth connectivity (Xie et al., 2025; Zhang et al., 2026). In the frozen-backbone setting, mHC-based PEFT learns read/write routing over multiple residual streams and investigates learned versus identity residual mixing (Oldenburg et al., 2026). These methods establish depth connectivity as an adaptation axis alongside local weight updates.

T-Router targets parameter-efficient RL through a coupled content-and-control interface: compressed, origin-indexed changes supply content; a separate depth-recurrent controller supplies routing context; and calibrated signed writeback supplies relative influence. A current-state query reads P from slots S to condition both selection and strength. Correctness rewards train this compact interface while preserving pretrained transformations and layer order. Appendix D develops these computational distinctions.

Thalamic regulation of cortical communication. The pulvinar coordinates cortical information transmission through attention-dependent synchronization (Saalmann et al., 2012). Mediodorsal thalamic input amplifies functional prefrontal connectivity, sustaining rule representations (Schmitt et al., 2017). These mechanisms motivate context-dependent control over the contributions of distributed computations. T-Router realizes this principle through source selection and writeback strength, conditioned on the receiving residual and recurrent depth context. HippoRAG draws on hippocampal indexing theory for external knowledge retrieval (Gutiérrez et al., 2024).

3 T-ROUTER

T-Router concentrates trainable capacity in a compact communication path around a frozen decoder. A source bank stores compressed block changes, a recurrent controller summarizes depth history, and a query-dependent readout conditions source attention and writeback strength. The router and gate combine this signal with the current residual to form an intervention. All pretrained layers execute in order; correctness rewards train how earlier contributions are reused.

3.1 RECORDING CHANGES ACROSS DEPTH

Let $H_\ell \in \mathbb{R}^{B\times T\times d}$ be the residual tensor before layer ℓ, for batch size B and sequence length T. With the batch index suppressed, $h_{\ell,t} \in \mathbb{R}^d$ denotes its token-t vector; R_ℓ stacks the token interventions $R_{\ell,t}$. The complete frozen decoder layer $\mathcal{B}_\ell$ acts on the sequence tensor, including its causal token interactions and residual connections:

$$\widetilde{H}_\ell = H_\ell + R_\ell, \qquad H_{\ell+1} = \mathcal{B}_\ell(\widetilde{H}_\ell). \tag{1}$$

Writing $\mathcal{B}_\ell(X) = X + F_\ell(X)$ gives $H_{\ell+1} = H_\ell + R_\ell + F_\ell(H_\ell + R_\ell)$: the intervention enters both the residual connection and the frozen transformation. Training updates only the auxiliary module.

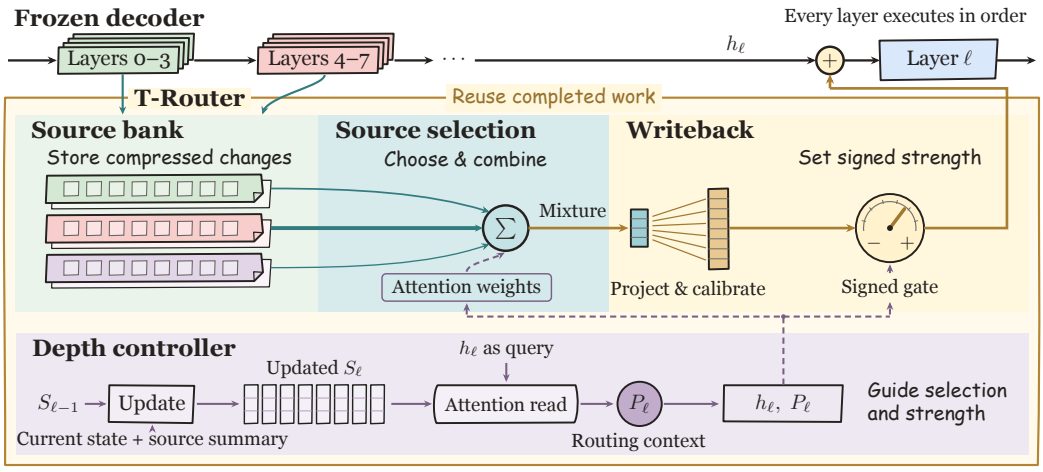


Figure 2: **How T-Router reuses completed computation.** Completed blocks supply compressed changes to the source bank. The depth controller updates its state S, then reads routing context P using the current hidden state. This context conditions which sources are mixed and how strongly the projected mixture is written back. The contribution is added before the receiving frozen layer; all layers execute in order. Teal and gold paths carry reusable content; dashed purple branches condition selection and strength. Equations 5–7 specify the operations.

We group consecutive layers into blocks of size s. Upon completing block b, T-Router compresses the change between its first actual input and final output:

$$D_{b,t} = h_{s(b+1),t} - \widetilde{h}_{sb,t}, \qquad m_{b,t} = C_b D_{b,t} \in \mathbb{R}^r. \tag{2}$$

Here $C_b \in \mathbb{R}^{r\times d}$ is learned. This change is measured along the intervened forward pass: writebacks inside the block can contribute to D_b. Before layer ℓ, only completed blocks $\mathcal{J}_\ell = \{b \in \mathbb{N}_0 : s(b+1) \le \ell\}$ are available. Each record pairs $m_{b,t}$ with a learned source embedding e_b^{src}. The bank stores changes at the same token position; it introduces no new attention over token positions.

We use $L = 32$, $s = 4$, and $r = 256$. The first four layers receive zero writeback; later layers can read up to seven completed blocks from the same forward pass.

3.2 From depth state to controller context

The controller summarizes the token's computation across layers to condition source selection and writeback strength. Its state comprises K slots $S_{\ell,t,k} \in \mathbb{R}^p$, initialized as $S_{-1,t,k} = S_k^0$ from a learned array. At every layer, the update combines the current residual, the mean $\bar{m}_{\ell,t}$ of completed records (zero for an empty bank), and the receiving-layer embedding e_ℓ:

$$z_{\ell,t} = [A_h h_{\ell,t}; A_m \bar{m}_{\ell,t}; e_\ell], \qquad [u_{\ell,t}; v_{\ell,t}] = \mathrm{MLP}(z_{\ell,t}). \tag{3}$$

Write attention ω distributes the shared proposal across the previous slots:

$$\begin{aligned} \omega_{\ell,t,k} &= \mathrm{softmax}_k\big(S_{\ell-1,t,k}^\top W_s z_{\ell,t}/\sqrt{p}\big)\,, \\ S_{\ell,t,k} &= \gamma S_{\ell-1,t,k} + (1-\gamma)\omega_{\ell,t,k}\big[\sigma(v_{\ell,t}) \odot \tanh(u_{\ell,t})\big]. \end{aligned} \tag{4}$$

We define *controller context* $P_{\ell,t} \in \mathbb{R}^p$ as an attention readout of these updated slots. The current residual supplies the query $Q_s h_{\ell,t}$; slot projections supply keys $K_s S_{\ell,t,k}$ and values $V_s S_{\ell,t,k}$:

$$\xi_{\ell,t,k} = \mathrm{softmax}_k\big((Q_s h_{\ell,t})^\top K_s S_{\ell,t,k}/\sqrt{p}\big)\,, \qquad P_{\ell,t} = \sum_{k=1}^{K} \xi_{\ell,t,k} V_s S_{\ell,t,k}. \tag{5}$$

Here ξ is normalized over the K slots. S is the recurrent state; P is its current control signal.

The router and gate combine P with h to determine source weights (Equation 6) and writeback strength (Equation 7). Source attention supplies the transported content c from the bank. We use $K = 8$, $p = 256$, and $\gamma = 0.9$; Appendix B.4 expands the history represented by S and P.

The slots and bank reset at each forward pass and maintain separate depth trajectories for each token. Information from preceding tokens reaches them through the backbone's causal representations and, during cached decoding, its cache.

3.3 Routing completed computations

For every layer with a nonempty bank, routing combines the current state, controller context, and layer identity:

$$\begin{aligned} q_{\ell,t} &= W_q[h_{\ell,t}; P_{\ell,t}; e_\ell], & k_{b,t} &= W_k[m_{b,t}; e_b^{\mathrm{src}}], \\ a_{\ell,t,b} &= \mathrm{softmax}_{b\in\mathcal{J}_\ell}\left(q_{\ell,t}^\top k_{b,t}/\sqrt{r}\right), & c_{\ell,t} &= \sum_{b\in\mathcal{J}_\ell} a_{\ell,t,b} W_v[m_{b,t}; e_b^{\mathrm{src}}]. \end{aligned} \tag{6}$$

Dense softmax mixes the visible sources in each forward pass. Source and layer embeddings identify origin and destination; the hidden state and controller context determine the query. Learned C_b, W_v, and U_ℓ determine how the selected content reaches the frozen layer. At a fixed receiving layer, all interventions lie in the at-most-r-dimensional column space of U_ℓ before dtype rounding, even after calibration and scalar gating. Equation 24 gives the aggregate transport form.

3.4 Calibrating the intervention

A gate on an unnormalized projection does not directly specify the size of its effect: the same scalar can multiply writeback directions with very different norms. We first map the retrieved source mixture to $w_{\ell,t} = U_\ell c_{\ell,t} \in \mathbb{R}^d$ and calibrate its RMS to the receiving state. Define $\mathrm{RMS}(x) = \sqrt{d^{-1}\sum_j x_j^2}$ and stop-gradient sg. The implemented update is

$$\begin{aligned} \widehat{w}_{\ell,t} &= \begin{cases} w_{\ell,t}\dfrac{\mathrm{sg}(\mathrm{RMS}(h_{\ell,t}))}{\mathrm{RMS}(w_{\ell,t})}, & \mathrm{RMS}(w_{\ell,t}) > \epsilon, \\ w_{\ell,t}, & \text{otherwise}, \end{cases} \\ g_{\ell,t} &= g_{\max}\tanh\!\left(b_\ell + w_g^\top[h_{\ell,t}; P_{\ell,t}]\right), \qquad R_{\ell,t} = g_{\ell,t}\widehat{w}_{\ell,t}. \end{aligned} \tag{7}$$

We use $\epsilon = 10^{-6}$ and $g_{\max} = 0.05$. The signed gate can reinforce or oppose the retrieved direction. Orthogonal initialization of U_ℓ, zero initialization of w_g, and $b_\ell = \mathrm{arctanh}(0.4)$ give an initial calibrated magnitude of 2%.

When $\mathrm{RMS}(w_{\ell,t}) > \epsilon$ and $h_{\ell,t} \neq 0$, calibration gives the local scale identity

$$\rho_{\ell,t} := \frac{\|R_{\ell,t}\|_2}{\|h_{\ell,t}\|_2} = |g_{\ell,t}| < g_{\max}. \tag{8}$$

The gate specifies the local relative intervention size independently of the raw projection scale. Subsequent nonlinear computation can amplify or attenuate its effect. The small-norm fallback is part of the implementation; Appendix B.5 derives the active-branch Jacobian and characterizes its threshold behavior.

3.5 Training the auxiliary module

For mathematics, the GRPO trainer implements a single-evaluation group-relative policy gradient with a frozen-base reference. For each prompt, it samples four completions and retries a correctness-uniform group at most three additional times, retaining the first mixed-correctness group or the final attempt. Binary parsed-correctness rewards are centered and standardized within valid completions to form A_i; responses that are both truncated and unparseable are omitted from this preference term.

The policy is the adapted language model's completion distribution. A sequence-normalized group-relative gradient trains source transport and control end to end, with one differentiable evaluation per retained group. Appendix B derives the objective and its credit-assignment paths. We optimize

$$\mathcal{L} = \mathcal{L}_{\mathrm{policy}} + \beta\mathcal{L}_{\mathrm{KL}} + \lambda\Omega, \qquad \beta = 0.02, \quad \lambda = 0.01. \tag{9}$$

Here $\mathcal{L}_{\mathrm{KL}}$ is the sampled token-level divergence surrogate to the backbone, and Ω penalizes routing non-uniformity and source-mixture magnitude using the full-tensor reduction. Appendix B specifies the exact surrogate, masks, and sampling distribution. Resampling changes the generated-data budget independently of the adapter. The module has 40,681,724 loss-connected parameters (0.454% of the backbone); its allocation is 41,730,332 (0.466%), including an unused terminal compressor and source-embedding row.

4 Experiments

4.1 Evaluation setting

We evaluate reasoning quality, trainable allocation, and the organization of that capacity. The backbone is Qwen3.5-9B-Base (Qwen Team, 2026), with 8,953,803,264 parameters. T-Router uses 32 layers, four-layer blocks, rank 256, and eight controller slots of dimension 256. It allocates 41.730M trainable parameters (0.466%); 40.682M are connected to the loss.

All trained methods in the mathematics comparison use the full GSM8K training set of 7,473 prompts. Evaluation covers all 1,319 GSM8K test questions (Cobbe et al., 2021), the 500-question MATH-500 split (Hendrycks et al., 2021; Lightman et al., 2024), and 30 questions from each of AIME 2024 and 2025. We report means $\pm$ sample standard deviations (SD) across three evaluation rounds with seeds 42, 43, and 44. Within each round, the equal-family score is

$$\mathrm{MathAvg} = \frac{\mathrm{GSM8K} + \mathrm{MATH\text{-}500} + \frac{1}{2}(\mathrm{AIME24} + \mathrm{AIME25})}{3}. \tag{10}$$

Aggregate SD is computed across round-level aggregates; paired differences match round indices before computing their mean and SD. Baselines include the frozen model, full-parameter GRPO, RFT, LoRA at ranks 2 and 16, LoRA-MoE with RO-GRPO, LoRA with S-GRPO, and rank-16 LoRA with matched informative retries. The last compares weight adaptation and controlled reuse under the same retry rule at a similar trainable scale. Appendix A details scoring and aggregation.

4.2 Parameter-efficient mathematical reasoning

Table 1: **Mathematical reasoning after GSM8K RL.** Accuracy (%), mean $\pm$ SD across three evaluation seeds. AIME averages the two annual scores. LoRA-r16 (retries) matches T-Router's retry rule. Parameters are in millions; a dash denotes no update. Bold/underlined means are best/second-best, including ties; fewer trainable parameters are preferred.

Method	Params (M)	GSM8K	MATH-500	AIME mean	MathAvg
Frozen base	—	88.73 $\pm$0.95	75.60 $\pm$2.25	31.67 $\pm$3.33	65.33 $\pm$0.89
Full-parameter GRPO	8954.000	93.86 $\pm$0.26	83.07 $\pm$0.70	44.44 $\pm$5.36	73.79 $\pm$1.83
RFT	8954.000	91.86 $\pm$1.16	82.93 $\pm$2.47	46.11 $\pm$2.55	73.64 $\pm$1.33
LoRA-r2 + GRPO	**5.410**	93.51 $\pm$0.23	84.53 $\pm$0.81	42.22 $\pm$12.73	73.42 $\pm$4.28
LoRA-r16 + GRPO	43.278	93.81 $\pm$0.77	85.13 $\pm$2.53	45.56 $\pm$6.31	74.83 $\pm$1.41
LoRA-MoE + RO-GRPO	173.112	95.30 $\pm$0.61	87.47 $\pm$1.45	44.44 $\pm$2.55	75.74 $\pm$1.47
LoRA + S-GRPO	43.278	94.64 $\pm$0.46	84.33 $\pm$1.86	<u>48.89</u> $\pm$8.22	75.95 $\pm$2.43
LoRA-r16 + GRPO, retries	43.278	<u>95.58</u> $\pm$0.50	<u>87.93</u> $\pm$0.12	48.33 $\pm$6.01	<u>77.28</u> $\pm$1.95
T-Router + GRPO	<u>41.730</u>	**97.62** $\pm$0.79	**92.73** $\pm$0.70	**60.56** $\pm$3.47	**83.64** $\pm$1.16

Higher accuracy with 0.466% of the backbone's parameters. T-Router reaches 83.64 ± 1.16 MathAvg with 41.73M trainable parameters, versus 73.79 ± 1.83 for full-parameter GRPO over all 8.95B parameters (Table 1). The paired improvement is 9.85 ± 2.93 points. The compact interface leads all three mathematical families, combining 97.62 on GSM8K with 92.73 on MATH-500 and 60.56 mean AIME accuracy. These results link parameter efficiency to reasoning performance beyond the GSM8K training domain.

Stronger generalization at a comparable adaptation budget. Matched-retry LoRA uses 43.28M parameters and reaches 77.28 ± 1.95 MathAvg. T-Router improves the mean by 6.36 points with

41.73M parameters. Its advantage spans MATH-500 (+4.80), GSM8K (+2.04), and AIME (about +12.2). The gains on broader and competition mathematics show that learning reuse supports generalization across mathematical task families. Figure 3a,b,d relates this improvement to trainable allocation under the matched retry rule.

4.3 AGENTIC SEARCH

Table 2: **Agentic-search comparison.** Mean ± SD across three evaluation seeds. BrowseComp Plus uses answer token-F1 ×100; ASearch Test uses a 0–100 score. Δ is the paired difference from full-parameter GRPO; dashes mark the reference. Rankings follow Table 1.

Method	BrowseComp F1	Δ vs GRPO	ASearch	Δ vs GRPO
Frozen base	5.11 ±0.30	−28.12 ±1.25	59.76 ±0.41	−10.19 ±0.56
Full-parameter GRPO	33.23 ±1.54	—	69.94 ±0.53	—
RFT	31.20 ±0.75	−2.03 ±2.07	65.81 ±0.92	−4.14 ±1.18
LoRA-r2 + GRPO	32.98 ±0.56	−0.25 ±1.25	69.19 ±0.59	−0.75 ±1.01
LoRA-r16 + GRPO	36.06 ±0.56	+2.83 ±1.80	70.59 ±0.18	+0.64 ±0.64
LoRA-MoE + RO-GRPO	**37.00** ±1.17	**+3.77** ±2.56	71.46 ±0.55	+1.52 ±0.65
LoRA + S-GRPO	35.50 ±0.57	+2.27 ±1.58	70.67 ±0.19	+0.73 ±0.34
LoRA-r16 + GRPO, retries	36.90 ±1.13	+3.67 ±1.32	72.80 ±0.75	+2.86 ±0.63
T-Router + GRPO	36.98 ±1.16	+3.75 ±0.91	**73.98** ±0.74	**+4.04** ±1.27

The search models are trained separately using ASearcher data (Gao et al., 2025) and the `search-agent-rl` implementation (siqi654321, 2026), which generates multi-turn trajectories with a retrieval-and-summary tool. T-Router reaches 36.98 ± 1.16 on BrowseComp Plus (Chen et al., 2025) and 73.98 ± 0.74 on ASearch Test (Table 2). Its paired gains over full-parameter GRPO are 3.75 ± 0.91 F1 points and 4.04 ± 1.27 ASearch points. It achieves the highest ASearch mean, 1.18 points above matched-retry LoRA. On BrowseComp Plus, its 36.98 F1 is comparable to LoRA-MoE's 37.00 with 24.1% of the latter's trainable allocation. These results extend the quality–allocation tradeoff to tool-mediated information search.

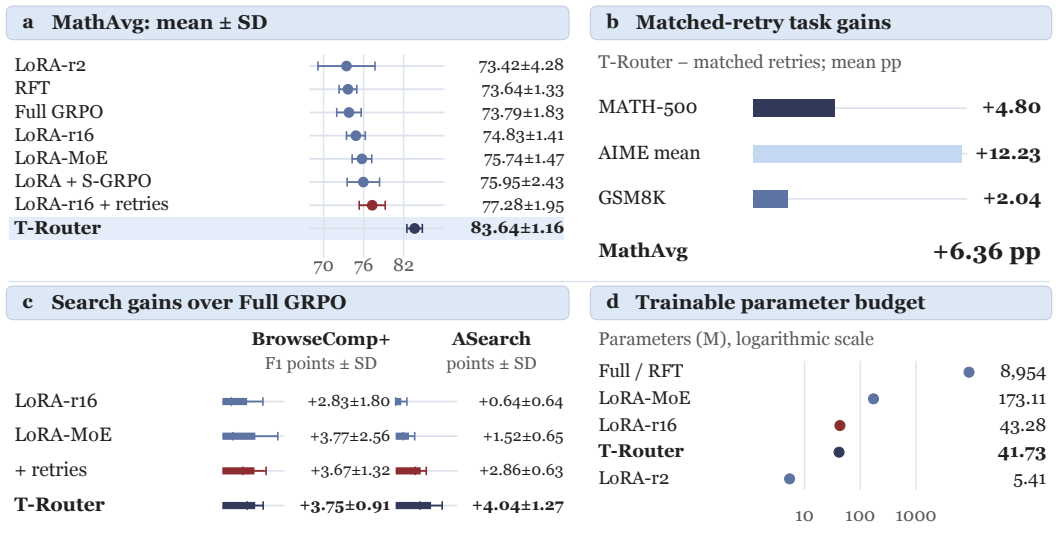


Figure 3: **Reasoning quality and parameter efficiency.** (a) MathAvg mean ± SD. (b) Task-family gains over matched-retry LoRA, computed from displayed means. (c) Paired gains over full-parameter GRPO in each search metric, mean ± SD. (d) Trainable allocations on a logarithmic scale; LoRA-r16 variants share the same budget. Error bars summarize three evaluation rounds.

Table 3: **Components of the reuse interface.** Accuracy (%), mean ± SD across evaluation seeds. Δ is the paired MathAvg difference from full T-Router; a dash marks the reference. The MLP retains comparable parameter capacity; hidden-state memory keeps the full allocation. Appendix C.2 includes both annual AIME scores.

Configuration	GSM8K	MATH-500	MathAvg	Δ
Full T-Router	**97.62** ±0.79	**92.73** ±0.70	**83.64** ±1.16	—
Without cross-layer retrieval	96.26 ±0.59	79.40 ±1.56	69.48 ±2.36	−14.16 ±1.78
Hidden-state memory	96.79 ±0.53	77.80 ±1.31	67.46 ±2.19	−16.18 ±2.96
Learned static routing	96.01 ±0.68	77.27 ±0.81	64.98 ±1.71	−18.66 ±2.78
State replaced by matched MLP	95.02 ±0.75	83.93 ±1.92	72.80 ±4.10	−10.84 ±3.09
Without layer/block identity	94.29 ±0.42	66.13 ±1.80	58.10 ±1.06	−25.53 ±0.56
Without RMS alignment	94.57 ±0.64	67.93 ±3.80	56.39 ±0.41	−27.25 ±1.24
Without token-conditioned gate	96.26 ±0.42	66.73 ±1.81	63.78 ±2.61	−19.86 ±1.47
Without informative retries	94.39 ±0.55	74.87 ±2.55	62.53 ±1.20	−21.11 ±2.26

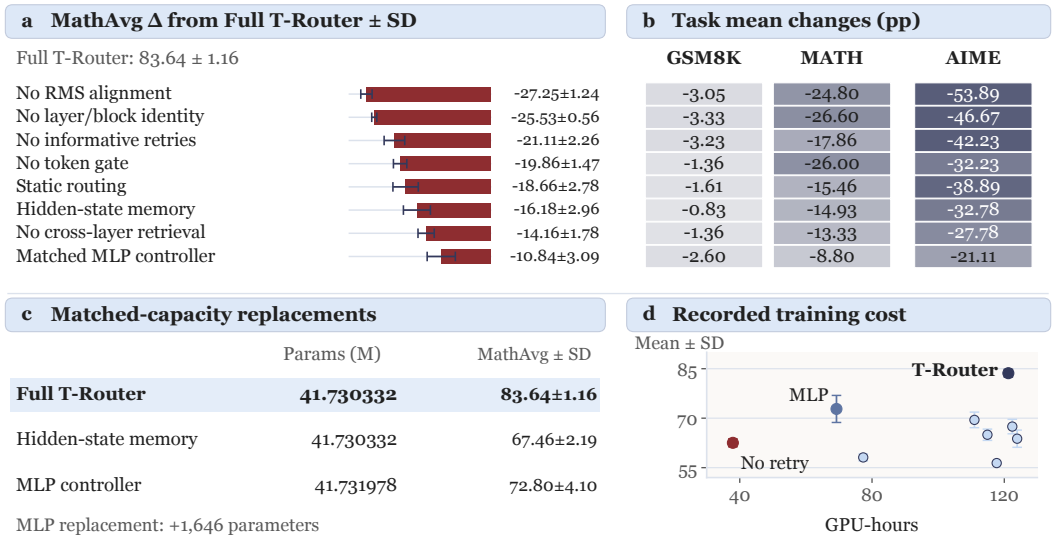


Figure 4: **Dissecting controlled reuse.** (a) Paired MathAvg changes from full T-Router, mean ± SD. (b) Family-wise changes in means; AIME averages both years. (c) Memory and controller replacements at matched capacity. (d) MathAvg versus recorded GPU-hours for all configurations. Numerical summaries in (c) and error bars in (d) show MathAvg SD across evaluation rounds. The no-retry setting also changes the generation budget.

4.4 Addressable Content and Recurrent Control

Table 3 tests the source bank, controller, writeback, and training recipe. Figure 4 aligns task responses with capacity and cost; Appendix C gives complete numerical profiles.

Preserve contributions as reusable content. With exactly 41,730,332 parameters in both settings, block-change memory reaches 83.64 ± 1.16 MathAvg versus 67.46 ± 2.19 for hidden-state memory. Removing cross-layer retrieval yields 69.48 ± 2.36. The full interface favors a particular use of its capacity: preserve completed contributions as individually addressable sources for later computation. Compression retains each block's identity, allowing a receiver to revisit its contribution through a low-dimensional record while preserving backbone width.

Use depth history as routing context. The recurrent interface reaches 83.64 ± 1.16 MathAvg versus 72.80 ± 4.10 for a capacity-matched MLP with 41,731,978 parameters. Here S accumulates depth history, and its readout P conditions the router and gate while content comes from the bank.

4.5 Source identity and relative-scale writeback

Table 3 tests the coordinates used to learn reuse. Layer and block identities specify the contribution's origin and receiving location; the full setting reaches 83.64 MathAvg versus 58.10 without them. The token-conditioned gate sets the mixture's signed influence; its removal gives 63.78. RMS alignment makes this influence relative to the receiving residual (Equation 8); omitting it gives 56.39. This calibration changes both forward intervention scale and gradient geometry, as derived in Appendix B.5. It provides a scale-defined optimization interface for learning writeback. Appendix C further characterizes the interaction between writeback and routing structure.

4.6 Learning and executing the reuse policy

Informative groups support learning the reuse policy. The full recipe records 121.23 GPU-hours and 83.64 ± 1.16 MathAvg; without informative retries, the corresponding values are 37.92 GPU-hours and 62.53 ± 1.20. Retries invest additional generation in finding mixed-reward groups that carry a policy-gradient signal. Matched-retry LoRA applies the same rule and reaches 77.28 ± 1.95 (Table 1). The capacity-matched MLP records 69.25 GPU-hours and 72.80 ± 4.10. Figure 4c,d shows these quality–cost profiles; parameter allocation and recorded training effort are distinct efficiency dimensions.

Summary: learning what to reuse and how strongly. At each receiver, source transports share U_ℓ, constraining the intervention to an at-most-r-dimensional subspace (Appendix B.3). Source attention, recurrent context, and signed gating choose an input-dependent intervention within that space. The method comparison establishes gains across three mathematical families over matched-retry LoRA; the capacity-controlled comparisons favor block changes and recurrent context. Correctness rewards optimize these content and control pathways together. Appendix C contains complete configuration profiles, costs, auxiliary metrics, and analytical score reaggregations.

5 Conclusion

T-Router improves the parameter efficiency of reasoning RL by learning how completed computations are reused. Its compressed source bank, depth-recurrent controller, and relative-scale writeback form a compact thalamic routing interface around a frozen backbone. With 0.466% of the backbone's parameter count, it reaches 83.64 MathAvg versus 73.79 for full-parameter GRPO and 77.28 for matched-retry LoRA. Capacity-controlled comparisons favor addressable changes and recurrent context; separate search training extends the interface to tool use. The central result is an efficient allocation of adaptation capacity: preserve pretrained transformations, and learn how their contributions serve subsequent reasoning.

Reproducibility statement

Section 3 defines the module and training objective. Appendix A gives tensor shapes, exact parameter allocation, execution order, and evaluation procedures. Appendix B supplies derivations. Appendix C includes the complete component results and costs, evaluation means and variances, configuration comparisons, auxiliary metrics, and definitions of every analytical reaggregation. The accompanying implementation includes the module, configuration constructor, training launcher, and evaluators; figure sources expose the tabulated numerical inputs.

AI use statement

We used AI assistants in two roles. First, for manuscript preparation, including conceptual framing, drafting, language editing, literature retrieval, mathematical derivations, and result interpretation. Second, for coding, LaTeX production, and figure preparation, including the AI-generated brain illustration. The authors made the final decisions on the method, experimental design, and implementation, verified all AI-assisted output, and take full responsibility for this paper.

REFERENCES

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. BrowseComp-Plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*, 2025. URL https://arxiv.org/abs/2508.06600.

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. URL https://arxiv.org/abs/2110.14168.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous RL. *arXiv preprint arXiv:2508.07976*, 2025. URL https://arxiv.org/abs/2508.07976.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 59532–59569, 2024. doi: 10.52202/079017-1902. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/6ddc001d07ca4f319af96a3024f6dbd1-Paper-Conference.pdf.

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, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/be83ab3ecd0db773eb2dc1b0a17836a1-Paper-round2.pdf.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799, 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

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

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017. URL https://openaccess.thecvf.com/content_cvpr_2017/html/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.html.

Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, Yutian Chen, Junjie Yan, Ming Wei, Y. Zhang, Fanqing Meng, Chao Hong, Xiaotong Xie, Shaowei Liu, Enzhe Lu, Yunpeng Tai, Yanru Chen, Xin Men, Haiqing Guo, Y. Charles, Haoyu Lu, Lin Sui, Jinguo Zhu, Zaida Zhou, Weiran He, Weixiao Huang, Xinran Xu, Yuzhi Wang, Guokun Lai, Yulun Du, Yuxin Wu, Zhilin Yang, and Xinyu Zhou. Attention residuals. *arXiv preprint arXiv:2603.15031*, 2026. URL https://arxiv.org/abs/2603.15031.

Alan Lee and Harry Tong. Token-efficient RL for LLM reasoning. In *ICML 2025 Workshop on Tiny Titans: The Next Wave of On-Device Learning for Foundational Models*, 2025. URL https://openreview.net/forum?id=XO8whE6Tx3.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Cheng Luo, Zefan Cai, and Junjie Hu. Delta attention residuals. *arXiv preprint arXiv:2605.18855*, 2026a. URL https://arxiv.org/abs/2605.18855.

Cheng Luo, Zefan Cai, and Junjie Hu. Multi-head attention residuals. *arXiv preprint arXiv:2607.27230*, 2026b. URL https://arxiv.org/abs/2607.27230.

Changlian Ma, Zizheng Huang, Xiangyu Zeng, Yi Wang, Cheng Liang, Kun Tian, Xinhai Zhao, and Limin Wang. Balancing the experts: Unlocking LoRA-MoE for GRPO via mechanism-aware rewards. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/7f403d7240d8a5b5335e60635ea2e975-Abstract-Conference.html.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Valentijn Oldenburg, Floris de Kam, Bente Zuijdam, Lieve Eberson, Nicky van Zutphen, Stef de Wildt, and Ivo Verhoeven. Manifold-constrained hyper-connections for parameter-efficient finetuning. *arXiv preprint arXiv:2607.18130*, 2026. URL https://arxiv.org/abs/2607.18130.

Matteo Pagliardini, Amirkeivan Mohtashami, Francois Fleuret, and Martin Jaggi. DenseFormer: Enhancing information flow in transformers via depth weighted averaging. In *Advances in Neural Information Processing Systems*, volume 37, pp. 136479–136508, 2024. doi: 10.52202/079017-4336. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/f67449c7ab72f441d3a713b046c6818c-Paper-Conference.pdf.

Qwen Team. Qwen3.5-9B-Base model card. Hugging Face, 2026. URL https://huggingface.co/Qwen/Qwen3.5-9B-Base. Accessed September 26, 2026.

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. URL https://openreview.net/forum?id=Ti67584b98.

Yuri B. Saalmann, Mark A. Pinsk, Liang Wang, Xin Li, and Sabine Kastner. The pulvinar regulates information transmission between cortical areas based on attention demands. *Science*, 337(6095): 753–756, 2012. doi: 10.1126/science.1223082. URL https://pubmed.ncbi.nlm.nih.gov/22879517/.

L. Ian Schmitt, Ralf D. Wimmer, Miho Nakajima, Michael Happ, Sima Mofakham, and Michael M. Halassa. Thalamic amplification of cortical connectivity sustains attentional control. *Nature*, 545 (7653):219–223, 2017. doi: 10.1038/nature22073. URL https://www.nature.com/articles/nature22073.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Hakim Sidahmed, Samrat Phatale, Alex Hutcheson, Zhuonan Lin, Zhang Chen, Zac Yu, Jarvis Jin, Simral Chaudhary, Roman Komarytsia, Christiane Ahlheim, Yonghao Zhu, Bowen Li, Saravanan Ganesh, Bill Byrne, Jessica Hoffmann, Hassan Mansoor, Wei Li, Abhinav Rastogi, and Lucas Dixon. Parameter efficient reinforcement learning from human feedback. *arXiv preprint arXiv:2403.10704*, 2024. URL https://arxiv.org/abs/2403.10704.

siqi654321. Search agent RL. GitHub software repository, 2026. URL https://github.com/siqi654321/search-agent-rl. Accessed September 26, 2026.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. MuSR: Testing the limits of chain-of-thought with multistep soft reasoning. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=jenyYQzue1.

Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. MUDDFormer: Breaking residual bottlenecks in transformers via multiway dynamic dense connections. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 68440–68458, 2025. URL https://proceedings.mlr.press/v267/xiao25d.html.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, Liang Zhao, Shangyan Zhou, Zhean Xu, Zhengyan Zhang, Wangding Zeng, Shengding Hu, Yuqing Wang, Jingyang Yuan, Lean Wang, and Wenfeng Liang. mHC: Manifold-constrained hyper-connections. *arXiv preprint arXiv:2512.24880*, 2025. URL https://arxiv.org/abs/2512.24880.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, pp. 113222–113244, 2025. doi: 10.52202/085713-3775. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a4277440d50f1f15d2cb4c14f7e0c0d2-Paper-Conference.pdf.

Yilang Zhang, Bingcong Li, Niao He, and Georgios B. Giannakis. ANCRe: Adaptive neural connection reassignment for efficient depth scaling. In *ICLR 2026 2nd Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*, 2026. URL https://openreview.net/forum?id=0BbbfFst1X.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. URL https://arxiv.org/abs/2311.07911.

T-Router

Learning Thalamic Routing
for Reasoning with Parameter-Efficient Reinforcement Learning

Appendix

Computational specification, formal analysis, and the supporting experimental results.

A Implementation and evaluation details **14**
- A.1 Notation and tensor organization . . . 14
- A.2 Parameter allocation and initialization . . . 15
- A.3 Layer schedule and source availability . . . 16
- A.4 State lifetime, precision, and differentiation . . . 17
- A.5 Training configuration and optimization flow . . . 18
- A.6 Evaluation tasks, repeated rounds, and scoring . . . 19
- A.7 Agentic search data and evaluation . . . 19
- A.8 Prompt construction and final-answer extraction . . . 21
- A.9 Auxiliary metric definitions . . . 22
- A.10 Analytical storage and projection work . . . 23
- A.11 How architectural dimensions allocate capacity . . . 23

B Formal analysis of cross-layer communication **25**
- B.1 Notation and completed-block event ordering . . . 25
- B.2 What a block change contains . . . 26
- B.3 Source transport and the receiving subspace . . . 26
- B.4 Controller history, rank, and state bounds . . . 27
- B.5 RMS geometry, gradients, and threshold behavior . . . 28
- B.6 Stop-gradient and scale-invariant optimization . . . 30
- B.7 Conditional propagation through the frozen decoder . . . 30
- B.8 The differentiated group-relative objective . . . 31
- B.9 Routing regularization and its reduction . . . 32
- B.10 Bounded informative resampling and generation budgets . . . 33

C Complete empirical results and design analysis **35**
- C.1 Method comparisons across mathematics and search . . . 35
- C.2 Complete component ablations . . . 36
- C.3 Parameter and training-cost profiles of the ablations . . . 36
- C.4 Configuration landscape and annual task profiles . . . 37
- C.5 Benchmark contributions to aggregate differences . . . 38
- C.6 Writeback configurations and task response . . . 39
- C.7 Writeback coefficients and endpoint diagnostics . . . 39
- C.8 Structure–writeback combinations . . . 41
- C.9 Auxiliary task profiles . . . 41
- C.10 Sensitivity to the choice of family weights . . . 42
- C.11 Question resolution and alternative aggregation . . . 43
- C.12 Evaluation-seed dispersion and complete variance record . . . 44

D Design interpretation and functional correspondence **47**
- D.1 Computation and coordination as separate roles . . . 47
- D.2 Why the source bank and controller carry different information . . . 47
- D.3 Three coordinates of a residual intervention . . . 48
- D.4 Relation to alternative adaptation interfaces . . . 49
- D.5 Connecting the empirical comparisons to the interface . . . 49

Reading routes Implementation A / Theory B / Results C / Interpretation D

GUIDE TO THE APPENDICES

The appendices provide the full computational specification, mathematical analysis, and configuration-level results supporting the communication interface. Table 4 organizes the material by the question it answers.

Table 4: **Appendix guide.** Each part develops a distinct aspect of the method and study.

Question	Material	Page
How is the module executed?	Tensor shapes, exact parameter counts, layer schedule, state lifetimes, training settings, prompts, scoring, and analytical storage	14
What properties follow from the design?	Causal source availability, transport subspaces, slot history, calibration derivatives, gradient flow, and finite group sampling	25
What do the configurations reveal?	Full-test comparisons, component results and costs, writeback and initialization comparisons, routing interactions, auxiliary diagnostics, evaluation variance, and metric sensitivity	35
How do the components fit together?	Thalamic functional correspondence, bank/controller roles, intervention coordinates, and adaptation interfaces	47

Tables 1–3 present the primary method and component comparisons. The results atlas includes their complete mean/SD profiles, sample variances, and resource measurements, alongside the historical configuration studies. Arithmetic decompositions and analytical illustrations are identified by their formulas and captions. The execution specification and derivations use zero-based decoder-layer indices, matching the source-availability schedule. Symbols retain the definitions in Section 3 throughout.

A IMPLEMENTATION AND EVALUATION DETAILS

This appendix specifies the computational objects, execution schedule, parameter allocation, and training and evaluation interfaces used by T-Router. The module operates on the residual input of a decoder layer. Source recording occurs after a completed block; the controller and receiving-layer intervention execute before the next frozen transformation. Keeping these events explicit gives a direct correspondence between the mathematical definition and a causal language-model implementation.

A.1 NOTATION AND TENSOR ORGANIZATION

The batch dimension is B, the number of positions in the current forward call is T, and the backbone residual width is d. We use L decoder layers, source blocks of size s, memory width r, K controller slots of width p, controller MLP width c, and source/layer embedding width e. Canonical values are

$$(L, s, d, r, K, p, c, e) = (32, 4, 4096, 256, 8, 256, 256, 32).$$

A token position is a position processed by the current call. During prompt prefill, T can be the entire prompt length. During ordinary one-token cached decoding, $T = 1$. All auxiliary operations broadcast over B and T; attention over source blocks and attention over controller slots are separate from the backbone's attention over token positions.

The ordering of the bank is chronological in depth. Every record keeps its own compressor and source identity even when its current routing weight is small. Layer embeddings identify receiving locations, while source embeddings identify completed blocks. Their roles are asymmetric: a receiver asks a context-dependent question about the available records, and each record provides an origin-labelled key and value. The canonical configuration retains all completed blocks within the forward call.

Controller context is the p-dimensional readout $P_{\ell,t}$ in Equation 5. The learned, bias-free maps $Q_s \in \mathbb{R}^{p\times d}$ and $K_s, V_s \in \mathbb{R}^{p\times p}$ are shared across layers. Attention compares $Q_s h_{\ell,t}$ with the K updated slot keys, normalizes over slots, and averages their projected values. The recurrent state S combines residual states, source summaries, and layer identity; the readout P conditions source

Table 5: **Runtime tensors and their axes.** $J_\ell = |\mathcal{J}_\ell|$ is the number of completed sources visible before layer ℓ. No source or slot axis is a token-history axis.

Object	Tensor shape	Role
Residual H_ℓ	$B \times T \times d$	Input to a receiving decoder layer
Block anchor A	$B \times T \times d$	Post-intervention input at the start of a block
Block displacement D_b	$B \times T \times d$	Completed block output minus its anchor
Compressed memory M_b	$B \times T \times r$	Source-specific record of that displacement
Visible bank	$B \times T \times J_\ell \times r$	Completed records available to a receiver
Controller slots S_ℓ	$B \times T \times K \times p$	Depth history used to condition retrieval
Controller context P_ℓ	$B \times T \times p$	Slot readout for the source query and gate
Source weights a_ℓ	$B \times T \times J_\ell$	Distribution over completed source blocks
Source mixture c_ℓ	$B \times T \times r$	Weighted sum of projected source values
Raw direction W_ℓ	$B \times T \times d$	Receiving-layer projection of c_ℓ
Gate g_ℓ	$B \times T \times 1$	Signed coefficient for calibrated writeback
Intervention R_ℓ	$B \times T \times d$	Additive input to the frozen layer

selection and the gate. For example, layers 4 and 5 see the same bank $\{m_0\}$, but their residuals and controller states can differ, yielding different contexts. With one source its routing weight is one, while the learned gate can still vary with P.

A.2 Parameter allocation and initialization

Table 6 counts scalar parameters by component. Matrices are expressed as output-by-input dimensions. The controller MLP is

$$\mathrm{MLP}(z) = W_2\,\mathrm{SiLU}(W_1 z + b_1) + b_2, \qquad z \in \mathbb{R}^{2p+e},$$

with hidden dimension c and output dimension $2p$. The two output halves provide a shared proposal and an elementwise proposal gate. The MLP has biases; the listed linear projections are otherwise bias free. The receiving gate has a shared vector and a separate bias for each routed layer.

Table 6: **Canonical module dimensions and exact parameter counts.** Counts include the allocated terminal source. The final row counts parameters with a path to the training loss.

Component	Shape or multiplicity	Parameters
Block compressors C_b	$8 \times (r \times d)$	8,388,608
Writeback projections U_ℓ	$28 \times (d \times r)$	29,360,128
Layer and source embeddings	$32e + 8e$	1,280
Initial controller slots	$K \times p$	2,048
Routing query W_q	$r \times (d + p + e)$	1,122,304
Routing key and value	$2 \times [r \times (r + e)]$	147,456
Hidden projection A_h	$p \times d$	1,048,576
Memory projection A_m	$p \times r$	65,536
Controller MLP	$(2p + e) \to c \to 2p$	271,104
Slot selector W_s	$p \times (2p + e)$	139,264
Slot query Q_s	$p \times d$	1,048,576
Slot key and value	$2 \times (p \times p)$	131,072
Dynamic and layer gates	$\mathbb{R}^{d+p}$ vector + 28 scalars	4,380
Allocated total		41,730,332
Terminal source parameters	$rd + e$	1,048,608
Loss-connected total		**40,681,724**

There are 28 receiving projections because the first four layers have no completed source to read. Eight compressors are allocated, one for each block, while the eighth source is produced after the last receiving decision. Removing that terminal compressor and its source embedding changes the allocated count by $rd + e = 1,048,608$ and leaves the output unchanged. We distinguish allocation from loss connectivity so that the parameter count describes both the implemented state and the effective learning interface.

The allocated count is 0.4661% of the 8,953,803,264-parameter backbone; the loss-connected count is 0.4544%. Receiving projections account for 29.36M scalars and source compression accounts for 8.39M. Together they contain approximately 90.46% of the allocation. The remainder implements routing, state update, slot reading, and scalar control. Thus most trainable capacity transforms source content into receiver-specific directions; the controller coordinates those transformations with a comparatively compact shared mechanism.

Table 7: **Initialization of the canonical learned-RMS module.** Initialization assigns content transport and intervention magnitude distinct roles.

Component	Initialization
Source compressors, source/layer embeddings, initial slots	Gaussian with standard deviation 10^{-3}
Routing, controller-input and slot projections	Xavier initialization
Controller MLP output layer	Zero weights and biases
Receiving projections U_ℓ	Orthogonal initialization, gain 1
Shared dynamic gate vector	Zero
Receiving-layer gate bias	$\operatorname{arctanh}(0.02/0.05)$
Gate bound and initial coefficient	$g_{\max} = 0.05$ and $g_{\text{init}} = 0.02$

The zero MLP output makes the initial proposal vanish, so the slots initially follow their decay from the learned starting array. This does not zero the complete module: the source bank and orthogonal receiving projections provide a direction, and the initialized gate applies a calibrated 2% coefficient. As optimization changes the proposal, different layers contribute different updates to the slot history. Appendix B.4 derives that accumulation.

A.3 Layer schedule and source availability

The distinction between a source's creation and a receiver's decision is explicit in Table 8. Before a layer executes, its controller reads the records that already exist. The intervention is applied next. If the layer starts a block, its post-intervention residual becomes the anchor. The frozen layer then runs, and a block-ending layer creates a new source for subsequent receivers.

Table 8: **Source availability over 32 decoder layers.** Layer indices are zero based. Each row covers four receiver decisions; the source appended at the end is available from the next row onward.

Receiving layers	Sources visible before layer	Appended after	New source	Read count
0–3	None	Layer 3	m_0	0
4–7	m_0	Layer 7	m_1	1
8–11	m_0, m_1	Layer 11	m_2	2
12–15	m_0, m_1, m_2	Layer 15	m_3	3
16–19	$m_0, \ldots, m_3$	Layer 19	m_4	4
20–23	$m_0, \ldots, m_4$	Layer 23	m_5	5
24–27	$m_0, \ldots, m_5$	Layer 27	m_6	6
28–31	$m_0, \ldots, m_6$	Layer 31	m_7	7

The total number of source–receiver pairs per position is $4(1 + 2 + \cdots + 7) = 112$. These are attention candidates, not 112 backbone layer executions: the backbone always runs exactly 32 layers. Source selection has one candidate in layers 4–7, so its normalized weight is identically one there. The receiving projection and gate still operate in those layers. Source preference becomes a nontrivial distribution at layer 8, when the second completed block is available.

At layer 4, for example, the controller has already progressed through four earlier updates. It incorporates m_0, reads its slots, and constructs the direction for layer 4 from the sole source. At layer 8, the same sequence of operations includes m_0 and m_1. Their attention weights can differ with the receiving token, hidden state, and accumulated controller context. This schedule separates the growth of the source set from the evolution of the state used to query it.

Algorithm 1 One causal T-Router forward pass

Require: Residual tensor $H_0 \in \mathbb{R}^{B\times T\times d}$, frozen layers $\mathcal{B}_0, \ldots, \mathcal{B}_{L-1}$
Copy the learned initial slots to all current batch/token positions
Set source bank $\mathcal{M} \leftarrow \varnothing$ and block anchor $A \leftarrow \varnothing$
for $\ell = 0, \ldots, L-1$ **do**
    Compute the mean visible memory; use zero when the bank is empty
    Update slots, then read P_ℓ using Equations 3–5
    if $\mathcal{M} \neq \varnothing$ **then**
        Compute source weights and retrieved source mixture using Equation 6
        Project, calibrate and gate the writeback using Equation 7
    else
        $R_\ell \leftarrow 0$
    end if
    $\widetilde{H}_\ell \leftarrow H_\ell + R_\ell$
    if $\ell \bmod s = 0$ **then**
        $A \leftarrow \widetilde{H}_\ell$
    end if
    $H_{\ell+1} \leftarrow \mathcal{B}_\ell(\widetilde{H}_\ell)$
    if $(\ell+1) \bmod s = 0$ **then**
        $b \leftarrow \lfloor \ell/s \rfloor$
        Append $(C_b(H_{\ell+1} - A), e_b^{\mathrm{src}})$ to $\mathcal{M}$
        Clear the completed block anchor
    end if
end for
return H_L

The post-intervention anchor matters for the meaning of a record. The writeback immediately before the block's first layer is already included in the anchor; subsequent writebacks are included in the completed displacement. A record therefore describes the change produced along the block's actual executed path. The decomposition in Appendix B makes this distinction algebraic. It is useful when comparing a stored displacement with a hypothetical sum of frozen-layer outputs evaluated on another trajectory.

A.4 State lifetime, precision, and differentiation

Every forward call starts a fresh bank and copies the learned initial slot array. Runtime state also resets when the first target layer is entered. During teacher forcing, each token position has its own depth-local bank and controller. During cached generation, each new call constructs these objects for the current positions, while the backbone retains its normal causal token cache. Depth recurrence therefore operates inside a forward call; the backbone cache carries context across calls.

Table 9: **Objects with different lifetimes.** The same word "memory" can refer to trainable state, temporary depth context, or the language model's causal cache.

Object	Lifetime	Update rule
Adapter parameters	Shared across examples and forward calls	Optimizer update during training
Initial slot array	Learned parameter	Copied at forward initialization
Controller state	Current forward call and token position	Recurrent update at each layer
Source bank	Current forward call and token position	Append after a completed block
Block anchor	Current block	Replace at block start; clear at block end
Backbone causal cache	Generation context	Managed by the backbone's decoder

The adapter computes its projections, slots, normalization, and gates in fp32. Frozen backbone tensors use bfloat16. The final intervention is cast back to the receiving residual's dtype before addition, keeping the next decoder layer's expected dtype. The target hidden-state RMS is detached; gradients

remain active through the direction normalization, source weights, source content, controller, and gate inputs. This is the implementation of the geometric distinction analyzed in Appendix B.5.

Frozen backbone parameters have no parameter gradients, while their input derivatives connect earlier interventions to the final loss. Compressed records remain in the graph as well. An earlier intervention can therefore influence later losses through both the ordinary residual path and the record subsequently read by another layer. The first routed layer is layer 4; the prefix before that point supplies inputs to trainable auxiliary operations without requiring gradients for its frozen weights.

Activation checkpointing encloses the frozen decoder-layer computation. Source recording and controller updates occur outside the recomputed function. Consequently, recomputation reevaluates a frozen transformation at its saved input and does not append another source or advance the controller a second time. This placement preserves the one-update-per-layer semantics of Algorithm 1. The adapter remains differentiable while the frozen backbone is kept in evaluation mode.

A.5 Training configuration and optimization flow

The final mathematics comparison trains each adaptation method on all 7,473 GSM8K training examples. Evaluation uses the full GSM8K test set, all 500 MATH-500 questions, and both 30-question AIME sets. The component study reports an estimated 3,737 optimizer steps per configuration, with exact trainable allocations and training costs in Table 21. Prompt count, optimizer steps, and generated responses describe different aspects of this training budget.

Table 10: **Final mathematics comparison: training and evaluation scope.** The full-test suite is shared by the methods in Table 1.

Setting	Value
Training data	GSM8K train: 7,473 questions
Evaluation data	GSM8K: 1,319; MATH-500: 500; AIME24/25: 30 each
Evaluation generation	One generated answer per question in each evaluation round
Evaluation aggregate	Equal weight for GSM8K, MATH-500, and mean AIME
T-Router allocation	41,730,332 parameters; 40,681,724 loss connected
Reference policy	Frozen backbone with the adapter disabled

Table 11: **Optimization controls exposed by the implementation.** Values are the implementation's configurable mathematics recipe. Sampling and regularization are specified separately from the benchmark aggregation rule.

Control	Value	Control	Value
Optimizer	AdamW	Weight decay	0.01
Peak learning rate	10^{-4}	Warmup	10% of scheduled updates
Schedule	Linear warmup/decay	Accumulation	2 prompt groups
Prompt batch size	1	Group size	4 completions
Additional retry limit	3 groups	Temperature	0.7
Top-p	0.95	Completion ceiling	512 tokens
KL coefficient	0.02	Allocation coefficient	0.01
Backbone dtype	bfloat16	Adapter arithmetic	fp32
Frozen-layer checkpointing	Enabled	Slot-diversity weight	0

For each prompt, generation uses the current adapted model. A group whose correctness rewards are all equal can be replaced by another group for that same prompt, with at most three replacements. The retained group is the first mixed-correctness group or the final attempt. The procedure then evaluates adapted and adapter-disabled log probabilities on the retained completions, computes group-standardized advantages, and differentiates the policy, KL, and allocation terms. The configurable recipe accumulates two prompt groups before an optimizer update. For N_{opt} scheduled updates, warmup lasts $\max(1, \lfloor N_{\mathrm{opt}}/10 \rfloor)$ updates.

The generation distribution uses temperature and nucleus sampling, whereas the objective uses full-vocabulary model log probabilities. An explicit top-k override is not applied, so generation retains

the loaded model's top-k setting. These are distinct parts of the procedure: sampling determines which responses enter the group, and the differentiable objective assigns gradients to those responses. The exact group selection law and finite response budget are derived in Appendix B.

The adapter-disabled reference retains the same frozen backbone parameters. No separately trained reward model is used; the reward is defined by final-answer correctness. The configurable recipe disables parse-failure reward shaping and the optional length-format and correct-response anchoring losses. A completion that is both truncated and unparseable has zero preference weight. Other incorrect but valid completions remain in the group with reward zero. The KL term applies to the nonpadding completion tokens of the retained responses. The allocation term uses source weights a_ℓ and source mixtures c_ℓ across the current sequence, including prompt positions. The corresponding reductions are specified in Equation 49 and the training analysis.

A.6 Evaluation tasks, repeated rounds, and scoring

Generative evaluations use three rounds with random seeds 42, 43, and 44. For a score x_j in round j, the reported statistics are

$$\bar{x} = \frac{1}{3}\sum_{j=1}^{3} x_j, \qquad s^2 = \frac{1}{2}\sum_{j=1}^{3}(x_j - \bar{x})^2, \qquad s = \sqrt{s^2}. \tag{11}$$

Tables report $\bar{x} \pm s$; s is the sample SD across evaluation rounds. Within each round, AIME mean averages the two annual accuracies and MathAvg averages the three task families. For method comparisons, $d_j = x_j - y_j$ pairs matching round indices, and the same formulas give $\bar{d}$ and its SD. Means and SD are displayed to two decimals, after aggregation; Appendix C.12 reports the complete variances to four decimals. Accuracy SD is in percentage points and its variance in squared percentage points; search scores use their corresponding points and squared points. Parameter allocations and GPU-hours are resource records, separate from these evaluation statistics.

The mathematics suite measures grade-school arithmetic, broader mathematical problem solving, and competition mathematics. Table 12 specifies the full-test denominators. Auxiliary tasks examine narrative reasoning with MuSR (Sprague et al., 2024), domain-question answering with GPQA (Rein et al., 2024), instruction compliance with IFEval (Zhou et al., 2023), and language-model likelihood with WikiText-2 (Merity et al., 2017). Their scores remain in their native units rather than entering MathAvg. MATH-500 uses the held-out subset released with PRM800K (Lightman et al., 2024), accessed through the Hugging Face interface in Table 13.

Table 12: **Full-test mathematics evaluation.** Each evaluation round covers the listed questions. AIME's two annual scores are averaged before the three-family MathAvg is computed.

Task	Questions	Scoring object
GSM8K	1,319	Normalized final answer
MATH-500	500	Mathematical equivalence of the final answer
AIME 2024	30	Mathematical equivalence of the final answer
AIME 2025	30	Mathematical equivalence of the final answer

The multiple-choice evaluator scores option likelihoods. MuSR provides narrative text, a question, and its explicit option list; the GPQA-D input contains the question and options. The evaluator tokenizes each space-prefixed option letter, requires it to be one token, and selects the highest next-token log probability. This decision rule makes the predicted option independent of the length of a generated explanation.

A.7 Agentic search data and evaluation

Search training uses ASearcher data (Gao et al., 2025) and the `search-agent-rl` implementation (siqi654321, 2026), which combines `verl` training, SGLang trajectory generation, and a retrieval-and-summary tool. The specifications below describe the public pipeline's data interfaces and default execution protocol. Dataset sizes count converted records before long-prompt filtering; retrieval and serving settings are repository defaults.

Table 13: **Dataset interfaces in the evaluation implementation.** These identifiers specify the task content accessed by the corresponding loaders.

Task	Dataset interface
GSM8K	`openai/gsm8k`
MATH-500	`HuggingFaceH4/MATH-500`
AIME 2024	`HuggingFaceH4/aime_2024`
AIME 2025	`yentinglin/aime_2025`
MuSR	`TAUR-Lab/MuSR`
GPQA-D	`fingertap/GPQA-Diamond`
IFEval	`google/IFEval`
WikiText-2	`Salesforce/wikitext`

Data partitions and evaluation roles. The conversion reads aidenjhwu/ASearcher_en_no-math_Qwen3-8B-reject-sample. Its 13,985-record snapshot is randomly split with seed 42 and a 2% holdout, producing 13,706 training records and 279 held-out records. The latter file is named `test.parquet`, but the training entry point passes it to `data.val_files` and validates every 50 optimizer steps. Accordingly, the column headed *ASearch Test* in Tables 2 and 19 refers to *ASearcher held-out validation*, rather than an additional independent test partition.

BrowseComp Plus (Chen et al., 2025) uses the 830 original questions in `Tevatron/browsecomp-plus`; the converter neither subsamples nor randomly repartitions them. Its evaluation entry point passes the same file to both `train_files` and `val_files`, with `val_before_train=True` and `val_only=True`. The corresponding control flow returns after validation, without training updates. The retrieval collection, `Tevatron/browsecomp-plus-corpus`, contains 100,195 documents in the public snapshot. These are source sizes; prompt filtering determines the effective evaluation count, and the local index determines the available retrieval collection.

Exact-match split diagnostic. The ASearcher converter splits records without first deduplicating question text. Reconstructing the default split from the fixed public snapshot identifies eight held-out records whose question text and reference answer exactly match a training record: $8/279 = 2.87\%$ of the holdout. The records have different IDs, so ID-disjointness alone does not reveal these matches. This diagnostic concerns exact question–answer matches in the reconstructed, pre-filter split; it does not quantify overlap in a run-specific filtered file or semantic near-duplicates. The overlap count is a property of this validation partition and does not, by itself, attribute a difference between methods to duplication.

Answer extraction and token-F1. Both conversion entry points select the `token_f1` scorer. It extracts the last complete `<answer>...</answer>` span and returns zero when no complete span exists. With multiple reference answers, it takes the highest answer-level F1. Whitespace is first collapsed; strings containing a space are split on spaces, and strings without spaces are split into characters, including a single English word. Case and punctuation are preserved. For token-multiset overlap o and answer lengths $n_{\text{pred}}, n_{\text{ref}}$, the nonempty-answer form is

$$\text{Prec} = \frac{o}{n_{\text{pred}}}, \qquad \text{Rec} = \frac{o}{n_{\text{ref}}}, \qquad \text{F1} = \frac{2\,\text{Prec}\,\text{Rec}}{\text{Prec} + \text{Rec}}, \tag{12}$$

with zero F1 when the overlap is zero. Scoring is string-based and does not use a language-model judge; the original BrowseComp-Plus benchmark's judge-based accuracy is a distinct metric.

Default returned score. The scorer's default reward applies repetition, answer-length, and answer-tag penalties to token-F1. Thus a table obtained directly from that reward has the interpretation *penalized token-F1* $\times 100$:

$$\text{Score} = \frac{100}{N} \sum_{i=1}^{N} \text{F1}_i \;\rho_{\text{rep},i}\,\rho_{\text{len},i}\,\rho_{\text{tag},i}.$$

Here ρ_{rep} is the scorer's repetition penalty. The length factor multiplies the score by 0.85 above 3,000 scoring tokens, and by a further 0.7 above 6,000. The tag factor is 0.25 if either the opening-

or closing-answer-tag count exceeds ten, and one otherwise. Scoring tokens follow the string rule above and are distinct from model tokens. A 0–100 score is a scaled mean reward, not the percentage of questions answered correctly. An unpenalized token-F1 value instead requires aggregation of the F1 component before these factors; the default returned reward must not be interpreted as that component alone.

Retrieval and summarization. Each `local_search(query)` call retrieves the top ten documents and uses a separate model to summarize them. Table 14 records the default settings. Both summarizers generate at most 256 model tokens with temperature 0.2 and top-p 0.9. BrowseComp Plus shortens the document and retries on summary failure, with an excerpt fallback. Its launcher uses the same `BASE_MODEL_PATH` variable for the actor base and summary service; overriding this variable can therefore change both roles. The listed models describe the repository defaults, not an invariant of the launcher under overrides.

Table 14: **Default search-tool settings.** Character limits apply to the initial document excerpts; summary limits use model tokens.

Setting	ASearcher training/validation	BrowseComp Plus evaluation
Retrieval collection	Local `wiki-18.jsonl`	BrowseComp-Plus corpus
Retriever	`e5-base-v2`	`Qwen3-Embedding-8B`
Summary model	`Qwen3-1.7B`	`Qwen3-8B`; three services
Retrieved documents	Top 10 per call	Top 10 per call
Initial excerpt	2,000 characters/document	10,000 characters/document
Summary ceiling	256 model tokens	256 model tokens
Tool-return truncation	2,048 characters	2,048 characters

Trajectory budgets and evaluation sampling. The default entry points cap the prompt at 4,096 model tokens, the response or multi-turn trajectory at 35,000 token positions, and total model length at 40,000. The trajectory limit includes tool feedback. At most 100 assistant-generation rounds are permitted, with at most one tool call per round; the termination order allows at most 99 executed tool-call rounds, and length limits can stop a trajectory earlier. Tool outputs are truncated by string slicing at 2,048 characters. Training samples eight trajectories per question. Validation uses one trajectory per question, with temperature 0.7, top-p 0.8, and top-k 20; the training setting `rollout.n=8` does not imply best-of-eight evaluation. These are execution ceilings and sampling settings, rather than measured equal token or tool-use costs across methods.

Reading the aggregate comparisons. Tables 2 and 19 report three-round score means and SDs; their paired differences use corresponding evaluation-round indices. Those SDs describe variation across the reported rounds. The ASearcher column characterizes performance on the validation partition identified above, while BrowseComp Plus supplies the separate benchmark evaluation. Differences are descriptive score comparisons; evaluation-round SDs do not themselves establish item-level statistical significance.

A.8 Prompt construction and final-answer extraction

Prompt templates define the answer interface for each task. They are kept fixed within a comparison. GSM8K training and evaluation use a direct-answer text prompt. Mathematics tasks use a problem-solving prompt whose final answer is requested in a box. Multiple-choice tasks request the option letter directly. The task text is inserted at the indicated symbolic field below; these fields describe the template, rather than individual generated examples.

```
Solve the following grade-school math problem.
Return only the final answer.
Question: question text
Answer:
```

```
Solve the following math problem carefully.  End with the final
answer in \boxed{...}.
Problem:  problem text
Solution:
```

```
question and option text
Answer immediately with only the option letter.  Do not explain or
show reasoning.
Answer:
```

GSM8K reference answers are normalized from the final-answer field. The prediction parser first recognizes an explicit final-answer delimiter, then considers the region after a completed thinking segment and the output as a whole. Within a candidate region, it prioritizes an explicit final-answer line, followed by a final numeric candidate from non-step lines. Normalization removes commas, currency markers and whitespace and strips a trailing period. A mathematical answer parser selects the final expression and checks that expression against the reference answer.

For mathematical equivalence, the scorer first uses the parsed expression with the mathematical verifier when both sides parse. Its fallback normalizes basic LaTeX formatting and compares expressions symbolically; ordered tuples are compared component by component. The selected final answer is the object being judged. An intermediate occurrence of the reference expression elsewhere in the reasoning trace does not itself satisfy the final-answer criterion.

Answer generation stops at the first end-of-sequence token or at the task's completion ceiling. A response at the ceiling is marked truncated. For a truncated response that remains inside an unfinished thinking segment without an explicit final answer, the score is incorrect. An explicit final-answer region, a box, or the designated final-answer markers allow the parser to evaluate the answer that was actually emitted. This rule treats termination and answer correctness as separate observables.

A.9 Auxiliary metric definitions

IFEval evaluates each instruction constraint using its task-specific checker. If prompt i has n_i constraints with binary outcomes z_{ij}, its strict prompt score is

$$I_i = \prod_{j=1}^{n_i} z_{ij}, \qquad \text{Strict@50} = \frac{100}{50} \sum_{i=1}^{50} I_i. \tag{13}$$

Every prompt therefore carries equal weight, independent of how many constraints it contains. The instruction-level average is a different quantity because it weights individual constraints. The reported strict@50 metric makes satisfying the entire prompt the unit of evaluation.

WikiText-2 perplexity evaluates language-model likelihood. The evaluator joins nonempty paragraphs, tokenizes the text, and scores overlapping windows while masking context positions covered by preceding windows. Window length, stride, and token limit are evaluation configuration parameters. If a window ends at token e_j after a previous end e_{j-1}, its new span has length $t_j = e_j - e_{j-1}$. With model-returned masked causal loss ℓ_j, the implementation uses

$$\text{PPL} = \exp\left(\frac{\sum_j w_j \ell_j}{\sum_j w_j}\right), \qquad w_j = \max(t_j - 1, 1). \tag{14}$$

The first window supplies the initial context; each later window reuses context and scores its newly exposed suffix. Perplexity is retained on its own lower-is-better scale in Table 25.

Final WB is a whole-tensor ratio at the latest routed layer,

$$\text{FinalWB} = 100 \frac{\|g_\ell \widehat{W}_\ell\|_F}{\|H_\ell\|_F}, \qquad \ell = 31. \tag{15}$$

The numerator is computed before the intervention is cast to the backbone dtype. Layer-wise execution overwrites the latest writeback statistic, which makes this diagnostic a description of the final receiving location. When all positions use active calibration, $(\text{FinalWB}/100)^2$ is a hidden-energy-weighted mean of squared token gates. It is consequently distinct from the arithmetic mean of gate coefficients. Appendix B.5 gives the corresponding identity and its interpretation.

A.10 Analytical storage and projection work

The auxiliary live state has three principal components: completed compressed sources, current controller slots, and the active block anchor. Before the final receiving layers, their scalar counts per batch/token position are $7r$, Kp, and d, respectively. For the canonical dimensions this is $1792 + 2048 + 4096 = 7936$ fp32 scalars, or 31 KiB. This count describes those named tensors; temporary projection operands, backbone state, and differentiation history have their own allocations.

Table 15: **Analytical storage of source bank, current slots, and anchor.** Values use fp32 and 2^{20} bytes per MiB. BT is the number of positions processed concurrently in one forward call.

BT	Source bank (MiB)	Slots (MiB)	Anchor (MiB)	Sum (MiB)
1	0.0068	0.0078	0.0156	0.0303
128	0.875	1.000	2.000	3.875
512	3.500	4.000	8.000	15.500
2048	14.000	16.000	32.000	62.000
8192	56.000	64.000	128.000	248.000

Under reverse-mode differentiation, earlier controller states and auxiliary operands participate in the graph. Keeping the controller history over L updates has a leading $O(BTLKp)$ scalar term. Repeated source projections additionally involve $O(BTr\sum_\ell J_\ell)$ operands. At inference, those gradient-history objects are absent, while current source and controller state remain. This distinction follows from the execution graph rather than from the allocated parameter count.

The two source-transport projection families have a particularly simple operation count. A dense d-to-r or r-to-d multiplication uses dr scalar multiply–accumulates per position. Eight block compressions and 28 writebacks therefore contribute

$$8dr + 28dr = 36dr = 37,748,736 \tag{16}$$

multiply–accumulates per batch/token position in the allocated execution. Omitting the terminal compression gives $35dr = 36,700,160$. Routing queries, source key/value projections, controller updates, and slot attention add their respective matrix and reduction operations. These formulas expose which dimensions control the auxiliary work: r scales transport width, Kp scales live controller state, and s sets the number and spacing of source records.

Unlike a fixed linear weight update, the intervention depends on the receiving state, visible records, controller history, and calibrated direction. Its computation remains part of the forward pass after training. The storage and operation counts above therefore characterize the resources of an explicit communication interface: a small trainable footprint with transient per-position state and context-dependent computation.

A.11 How architectural dimensions allocate capacity

The parameter count can be evaluated for other source ranks and block sizes without training a new model. This calculation separates architectural capacity from measured task quality. Let $M = L/s$ for a block size that divides L. With all layers targeted and the same dense controller structure, the allocated count decomposes as

$$\begin{aligned} P_{\text{transport}} &= (M + L - s)dr, \\ P_{\text{route}} &= r(d + p + e) + 2r(r + e), \\ P_{\text{controller}} &= 2pd + pr + c(2p + e) + c + 2pc + 2p \\ &\quad + p(2p + e) + 2p^2 + Kp, \\ P_{\text{identity}} &= e(L + M), \qquad P_{\text{gate}} = d + p + L - s. \end{aligned} \tag{17}$$

Their sum gives 41,730,332 at the canonical dimensions. The receiving and compression terms grow linearly with rank, while the source key/value term also has a quadratic component. Controller width is held fixed in Figure 5, which isolates the allocation changes induced by source rank and block size.

A smaller block size creates more sources and enables earlier writeback, increasing transport allocation and source–receiver pairs, $sM(M-1)/2$. A larger block compresses a longer interval into each record, reducing the resolution of the addressable history.

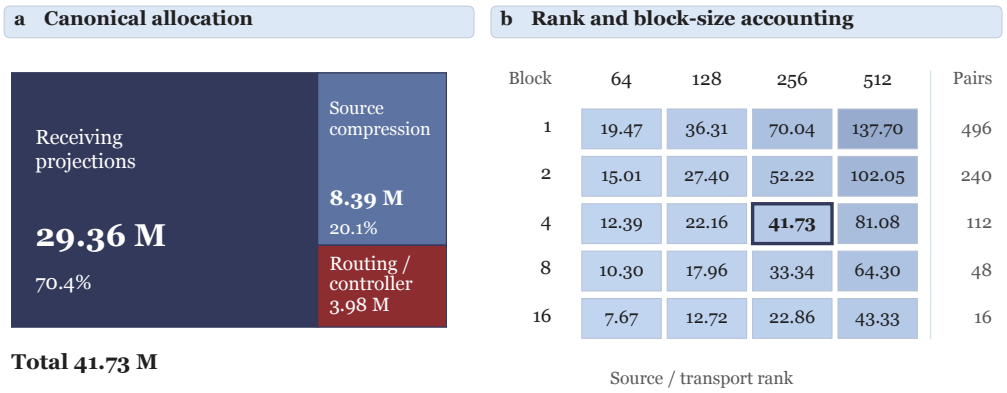


Figure 5: **Analytical allocation of adaptation capacity.** Left: area-proportional component allocation in the canonical module. Right: Equation 17 over source ranks and block sizes, with $L = 32$, $d = 4096$, $K = 8$, $p = c = 256$, and $e = 32$. Cells give parameters in millions; the outlined cell is canonical. The last column counts source–receiver pairs.

Memory rank sets the width of each stored change and the maximal receiving subspace dimension. Controller width sets the capacity used to select among records. Their equal canonical width, $r = p = 256$, does not make these roles interchangeable; Figure 5 quantifies the resulting allocation.

B FORMAL ANALYSIS OF CROSS-LAYER COMMUNICATION

This section analyzes the interface through which T-Router reuses completed computations. We first establish its event ordering and the content of its block records. We then characterize the receiving subspace, the history represented by controller slots, the geometry of calibrated interventions, and their propagation through frozen layers. The final part derives the differentiated training objective and the distribution induced by bounded informative resampling. These results concern the specified computation in exact arithmetic; casts to the backbone dtype occur after the calibrated intervention is formed.

B.1 NOTATION AND COMPLETED-BLOCK EVENT ORDERING

We use sequence tensors $H_\ell \in \mathbb{R}^{B\times T\times d}$ and token vectors $h_{\ell,t} \in \mathbb{R}^d$, suppressing the batch index unless a reduction requires it. The norm of a sequence tensor is its Frobenius norm. Each $\mathcal{B}_\ell$ is a complete frozen decoder layer, including causal token mixing and its internal residual connections. Auxiliary projections and attentions are tokenwise. The K controller slots for one token form the rows of $\mathbf{S}_{\ell,t} \in \mathbb{R}^{K\times p}$. Layer indices start at zero; $\mathbf{S}_{-1,t} = \mathbf{S}^0$ denotes the learned initial state.

For block size s, the events at depth ℓ occur in the order

$$\begin{gathered}\text{read completed records} \;\longrightarrow\; \text{update and read controller} \;\longrightarrow\; R_\ell,\\ \widetilde{H}_\ell = H_\ell + R_\ell \;\longrightarrow\; H_{\ell+1} = \mathcal{B}_\ell(\widetilde{H}_\ell) \;\longrightarrow\; \text{append if the block ends.}\end{gathered} \tag{18}$$

An anchor $\widetilde{H}_{sb}$ is associated with the first actual input of block b. Its compressed record is appended after layer $s(b+1)-1$ finishes. Thus the readable set before layer ℓ is

$$\mathcal{J}_\ell = \{0,\dots,\lfloor \ell/s\rfloor - 1\}, \qquad n_\ell := |\mathcal{J}_\ell| = \lfloor \ell/s\rfloor, \tag{19}$$

with the set empty when $\ell < s$. This formula assumes the full-history configuration and $s \mid L$. It distinguishes storing a record from using it: block b has exactly $L - s(b+1)$ receiving layers, when this quantity is positive.

Proposition 1 (Causal source availability)**.** *Assume each frozen layer is causal over token positions. Before layer ℓ, the residual, controller slots, and visible records at position t depend only on input positions at most t. Routing reads only completed blocks and does not depend on the output of its receiving layer.*

Proof. At the first layer the input embeddings are causal, the slots are copied from an input-independent parameter array, and the bank is empty. Suppose the claim holds before layer ℓ. Every auxiliary operation at position t uses $h_{\ell,t}$, slots at that position, and compressed records at that position. Linear maps, nonlinearities, slot attention, source attention, and RMS calibration therefore preserve the same token support. Applying the causal map $\mathcal{B}_\ell$ also preserves it. A new record, if created, is a tokenwise linear map of the difference between this output and an earlier causal anchor. Equation 18 places that append after the receiving decision, so it is available only to later layers. Induction establishes both token causality and depth causality. □

The argument also explains the state lifetime. A new forward pass initializes fresh slots and an empty bank. In cached decoding, earlier tokens influence the current token through the backbone's causal cache; the auxiliary state evolves along the current pass's depth. Under exact arithmetic, identical position/mask semantics, and cache-equivalent frozen layers, teacher-forced and cached evaluations construct the same token-local auxiliary state for the same prefix. This follows by applying the induction to the current token at each depth. It does not require persistent controller slots across generated tokens.

For $L/s = m$ complete blocks, the total number of source–receiver pairs is

$$\sum_{\ell=0}^{L-1} n_\ell = s\sum_{j=0}^{m-1} j = \frac{sm(m-1)}{2}. \tag{20}$$

The canonical configuration has 112 pairs. The terminal record has no consumer; deleting its append leaves H_L and the training loss unchanged. This is an execution identity rather than an approximation.

B.2 What a block change contains

Write $\mathcal{B}_j(X) = X + F_j(X)$, where F_j includes all computations within the complete layer beyond its outer identity term. This is an algebraic decomposition of the frozen layer, not a replacement for its internal architecture. The actual residual transition is

$$H_{j+1} - H_j = R_j + F_j(\widetilde{H}_j). \tag{21}$$

Summing across block b and subtracting the post-intervention anchor gives

$$\begin{aligned} D_b &= H_{s(b+1)} - \widetilde{H}_{sb} \\ &= \sum_{j=sb}^{s(b+1)-1} F_j(\widetilde{H}_j) + \sum_{j=sb+1}^{s(b+1)-1} R_j. \end{aligned} \tag{22}$$

Indeed, the telescoping sum first includes R_{sb}, and the subtraction of $\widetilde{H}_{sb} = H_{sb} + R_{sb}$ removes precisely that term. All later injections within the block remain. For a two-layer block, for example, $D_b = F_{2b}(\widetilde{H}_{2b}) + R_{2b+1} + F_{2b+1}(\widetilde{H}_{2b+1})$.

The first injection can still influence every F_j through its input. Equation 22 excludes its *explicit additive term*, not its downstream computational effect. The record therefore describes the change realized by the intervened block. It is neither a separately evaluated unmodified-backbone change nor a causal attribution to an isolated layer. This distinction makes the record usable during one forward pass: subsequent routing operates on computations the model has actually completed.

Two immediate consequences are useful. First, compression yields

$$\|m_b\|_F \le \|C_b\|_2 \left(\sum_{j=sb}^{s(b+1)-1} \|F_j(\widetilde{H}_j)\|_F + \sum_{j=sb+1}^{s(b+1)-1} \|R_j\|_F \right). \tag{23}$$

Second, both endpoints remain differentiable. If $G_b = \partial \mathcal{L} / \partial m_b$ denotes the adjoint from all later uses, the direct contribution to the compressor gradient is $\nabla_{C_b} \mathcal{L} = \sum_{n,t} G_{b,n,t} D_{b,n,t}^\top$, and the adjoint entering D_b is $C_b^\top G_b$. The endpoint difference distributes that adjoint with opposite signs to output and anchor, after which ordinary reverse-mode differentiation follows their shared earlier computation. Shared ancestors are handled by summing these paths, not by treating the two endpoints as independent model executions.

B.3 Source transport and the receiving subspace

Partition the routing value projection into content and identity terms, $W_v = [W_v^m \; W_v^e]$. For one receiving layer and token, the raw direction is

$$w_{\ell,t} = \sum_{b \in \mathcal{J}_\ell} a_{\ell,t,b} \left[\underbrace{U_\ell W_v^m C_b}_{T_{\ell b}} D_{b,t} + U_\ell W_v^e e_b^{\mathrm{src}} \right], \quad \operatorname{rank}(T_{\ell b}) \le r. \tag{24}$$

Conditioning on the attention weights, this is affine in the collection of block changes. The content transports $T_{\ell b}$ can differ across sources through C_b, while all terms share the receiving projection U_ℓ. In particular, the concatenated content transport $[T_{\ell 0} \; T_{\ell 1} \; \cdots]$ also has rank at most r, rather than the sum of the individual rank bounds.

Proposition 2 (Aggregate receiving subspace)**.** *At fixed parameters and receiving layer ℓ, every intervention lies in* $\operatorname{col}(U_\ell)$*:*

$$\operatorname{span}\{R_{\ell,t} : \textit{inputs and positions } t\} \subseteq \operatorname{col}(U_\ell), \qquad \dim \operatorname{col}(U_\ell) \le r. \tag{25}$$

This holds for both branches of RMS calibration and for signed scalar gates.

Proof. Source attention first produces $c_{\ell,t} \in \mathbb{R}^r$. Calibration and gating multiply $U_\ell c_{\ell,t}$ by a scalar $\alpha_{\ell,t}$, with $\alpha = g \operatorname{RMS}(h) / \operatorname{RMS}(w)$ on the active branch and $\alpha = g$ on the fallback branch. Thus $R_{\ell,t} = U_\ell(\alpha_{\ell,t} c_{\ell,t})$. Every such vector belongs to the same column space, including zero vectors and source-identity contributions. Taking a span proves the result. □

There is a corresponding differential statement. Let x collect any continuous upstream inputs to a token's routing decision, and write $f_\ell(x) = \alpha_\ell(x)c_\ell(x)$. At any differentiable point away from a calibration boundary,

$$\frac{\partial R_\ell}{\partial x} = U_\ell \frac{\partial f_\ell}{\partial x}, \qquad \operatorname{rank}\left(\frac{\partial R_\ell}{\partial x}\right) \le r. \tag{26}$$

Adaptive attention changes the coefficient derivative but not its receiving column space. Holding the auxiliary history fixed and differentiating the layer-input interface gives $\partial \widetilde{h}_\ell / \partial h_\ell = I + U_\ell \partial f_\ell / \partial h_\ell$. This locates the low-dimensional modification at the interface itself. The subsequent nonlinear $\mathcal{B}_\ell$ can transform it, and comparing whole-layer Jacobians at two different inputs introduces the change in the backbone Jacobian as well.

Addressable records and coefficient choice. For fixed visible values $v_b = W_v[m_b; e_b^{\mathrm{src}}]$, dense softmax gives $c \in \operatorname{conv}\{v_b\}$; with finite logits all weights are positive. At the raw interface w lies in the corresponding convex hull of $U_\ell v_b$. Calibration preserves the retrieved direction, while the signed gate sets the final magnitude and sign. Thus dense attention permits a state-dependent mixture without implying sparse execution or omission of a frozen layer. When there is one record, $a_0 = 1$ and the attention-logit derivative is zero, although the content/value path is still trainable. With additional records, the controller and current state can change the mixture within the same receiving subspace.

The interface is generally nonlinear in its input: attention weights, controller history, normalization, and gating all depend on that input. A fixed rank-r transport and an input-dependent rank-r interface share a dimensional constraint but need not implement the same function. For example, with one nonzero value and a scalar gate depending on h, $R(h) = g(h)\widehat{w}(h)$ already varies nonlinearly. It cannot in general be absorbed into one constant additive weight matrix.

B.4 Controller history, rank, and state bounds

Suppressing the token index, define $\nu_\ell = \sigma(v_\ell) \odot \tanh(u_\ell) \in (-1, 1)^p$ and let ω_ℓ be the slot-selection probability vector. We take $0 \le \gamma < 1$, with $\gamma = 0.9$ in the canonical configuration. The row-stacked update is

$$\begin{aligned} \mathbf{S}_\ell &= \gamma \mathbf{S}_{\ell-1} + (1-\gamma)\omega_\ell \nu_\ell^\top \\ &= \gamma^{\ell+1}\mathbf{S}^0 + (1-\gamma)\sum_{j=0}^{\ell} \gamma^{\ell-j}\omega_j \nu_j^\top . \end{aligned} \tag{27}$$

The second equality follows by substitution, starting from $\mathbf{S}_{-1} = \mathbf{S}^0$. It is valid even though ω_j and ν_j depend on the realized preceding states. Unrolling a trajectory is an algebraic operation; it does not assume independent updates.

Rank and distinct history mixtures. Each added matrix has rank at most one, so subadditivity gives

$$\operatorname{rank}(\mathbf{S}_\ell) \le \min\{K, p, \operatorname{rank}(\mathbf{S}^0) + \ell + 1\}. \tag{28}$$

To see why the state need not remain rank one, take $K = p = 2$, $\mathbf{S}^0 = (1,2)^\top e_1^\top$, scaled selector $W_s z/\sqrt{p} = e_1$, and a nonzero proposal proportional to e_2. The selector produces $\omega = \operatorname{softmax}(1,2)$, whose entries are not in the ratio $1:2$. Therefore the two columns of $\gamma(1,2)^\top e_1^\top + (1-\gamma)\omega\nu^\top$ are linearly independent for $0 < \gamma < 1$. A rank-one initial state becomes rank two in one valid shared-proposal update. Both proposal coordinates and selector scale can be chosen within the specified parameterization.

There are also exact symmetry cases. If all initial rows are equal, their slot logits are equal, so $\omega_k = 1/K$. The shared proposal preserves equal rows; by induction, the slots remain identical. If ω_j is a fixed vector at every depth and the initial state has the same row factor, the complete history also has that row factor and rank at most one. These cases show what produces multiple slot mixtures: distinct initial rows and depth-dependent slot allocation, together with different proposal directions. The learned initialization allows this asymmetry without assigning a prescribed semantic role to any slot.

For a fixed trajectory, substituting Equation 27 into the attention readout (Equation 5) expresses the controller context in terms of layer-wise proposals:

$$P_\ell = \gamma^{\ell+1} \sum_k \xi_{\ell,k} V_s S_k^0 + (1-\gamma) \sum_{j=0}^{\ell} \gamma^{\ell-j} \underbrace{(\xi_\ell^\top \omega_j)}_{\text{read–write overlap}} V_s \nu_j. \tag{29}$$

The scalar overlap is between zero and one. Each past proposal contributes according to both its earlier slot assignment and the current slot read. Consequently, a decay average of the proposals alone does not generally determine P_ℓ: the history of slot assignments also matters. At the same time, the slots are finite-dimensional mixtures of that history, not an unbounded list of separately addressable depth records.

Forward bounds and conditional sensitivity. The bounded proposal and probability selector imply

$$\begin{aligned} \|\mathbf{S}_\ell\|_F &\le \gamma^{\ell+1}\|\mathbf{S}^0\|_F + (1-\gamma^{\ell+1})\sqrt{p}, \\ \sum_k |S_{\ell,kj}| &\le \gamma^{\ell+1} \sum_k |S^0_{kj}| + 1 - \gamma^{\ell+1}. \end{aligned} \tag{30}$$

For the first line, $\|\omega\nu^\top\|_F = \|\omega\|_2\|\nu\|_2 \le \sqrt{p}$; apply the triangle inequality to the unrolled history. For the second, sum absolute values over slots and use $\sum_k \omega_k = 1$. Since readout attention is a probability vector, $\|P_\ell\|_2 \le \|V_s\|_2 \max_k \|S_{\ell,k}\|_2$. These bounds apply to forward values for any inputs and selector logits.

Sensitivity involves the selector as well as γ. Hold z_ℓ fixed and write $a = W_s z_\ell/\sqrt{p}$, $\omega = \mathrm{softmax}(\mathbf{S}a)$. The differential of one update with respect to the previous slots is

$$\Delta\mathbf{S}^+ = \gamma\Delta\mathbf{S} + (1-\gamma)(\mathrm{diag}\,\omega - \omega\omega^\top)(\Delta\mathbf{S}a)\nu^\top. \tag{31}$$

The softmax Jacobian has operator norm at most $1/2$: its quadratic form is the variance of the entries of a vector under ω, at most one quarter of their squared range, and that squared range is at most twice the vector's squared norm. Hence

$$\|\Delta\mathbf{S}^+\|_F \le \left[\gamma + \frac{1-\gamma}{2}\|a\|_2\|\nu\|_2\right] \|\Delta\mathbf{S}\|_F. \tag{32}$$

This is a contraction bound when $\|a\|_2\|\nu\|_2 < 2$ with fixed z_ℓ. In the full model z_ℓ also depends on the current residual and visible records, so its derivative supplies additional paths. The decay factor controls historical coefficients; it is not by itself the Jacobian norm of the complete controller.

B.5 RMS GEOMETRY, GRADIENTS, AND THRESHOLD BEHAVIOR

For this subsection suppress layer and token indices, put $q = \mathrm{RMS}(w)$ and $\kappa = \mathrm{RMS}(h)$, and define the calibrated map $f_\kappa(w) = \widehat{w}$ from Equation 7. On the active branch $q > \epsilon$, $\mathrm{RMS}(x) = \|x\|_2/\sqrt{d}$ gives

$$f_\kappa(w) = \|h\|_2 \frac{w}{\|w\|_2}, \qquad \|f_\kappa(w)\|_2 = \|h\|_2. \tag{33}$$

Calibration maps the raw direction to the sphere whose radius is the receiving state's norm. Positive rescaling of w cancels while both points stay on this branch; negative rescaling reverses the direction. Multiplication by the signed gate gives $\|R\|_2/\|h\|_2 = |g|$ for $h \ne 0$. This explains how gate magnitude can be interpreted independently of the scale of the compression and projection matrices.

Tangential and radial derivatives. Holding the reference scale fixed, differentiation gives

$$J_w := \frac{\partial f_\kappa(w)}{\partial w} = \frac{\kappa}{q}\left(I - \frac{ww^\top}{\|w\|_2^2}\right), \quad J_w w = 0, \quad \|J_w\|_2 = \frac{\kappa}{q} \quad (d > 1). \tag{34}$$

To derive it, $\mathrm{d}q = w^\top \mathrm{d}w/(d\,q)$, where d is the vector dimension. Applying the product rule to $\kappa w/q$ yields $\mathrm{d}f = (\kappa/q)\mathrm{d}w - \kappa w(w^\top \mathrm{d}w)/(d\,q^3)$; since $d\,q^2 = \|w\|_2^2$, Equation 34 follows. The matrix in parentheses is the orthogonal projector onto the tangent space perpendicular to w. Its radial eigenvalue is zero and its $d-1$ tangential eigenvalues are one. Thus calibration learns direction

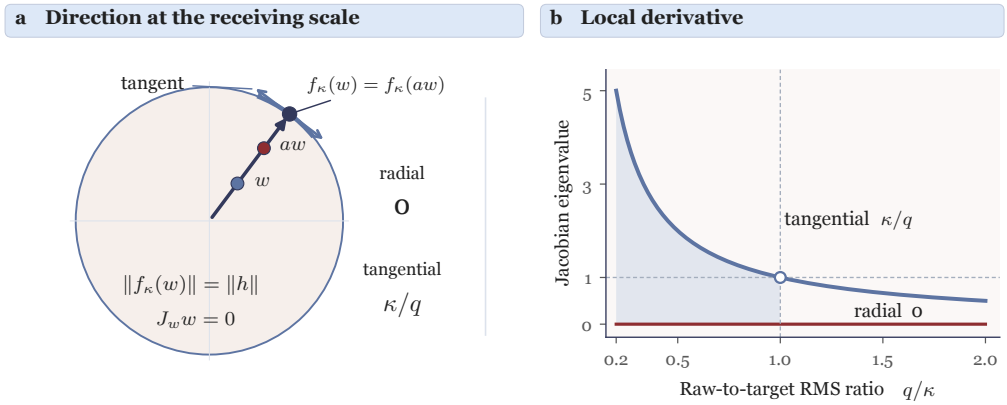


Figure 6: **Analytical geometry of RMS calibration.** Left: in two dimensions with $\|h\| = 1$, positive multiples on the same ray map to one point on the target sphere; the double arrow marks a tangent. The local basis separates zero radial response from tangential gain κ/q. Right: the active-branch Jacobian eigenvalues from Equation 34. The gate subsequently sets signed amplitude.

through tangential gradients; the signed gate supplies a separate radial degree of freedom in the final intervention.

Figure 6 separates three geometric objects: the ray selected by the raw projection, the sphere set by the receiving residual, and the local derivative along or across that ray. A positive radial change of the raw vector leaves its calibrated output fixed. A tangential change rotates the direction, with sensitivity determined by κ/q. The gate's derivative supplies the remaining scalar control of the final intervention. This geometry connects the parameterization to the separate source, transport, and strength coordinates discussed in Appendix D.

Let $a = \partial\mathcal{L}/\partial R$ be an incoming adjoint and temporarily hold routing inputs fixed. Then

$$\frac{\partial \mathcal{L}}{\partial w} = g J_w a, \qquad \frac{\partial \mathcal{L}}{\partial g} = a^\top f_\kappa(w). \tag{35}$$

The normalization derivative can have gain greater than one when the raw RMS is small relative to the reference RMS. Meanwhile, for gate logit z, $\partial g/\partial z = g_{\max}\,\mathrm{sech}^2 z \le g_{\max}$. These two derivatives control different quantities: the first changes the direction, while the second changes its signed coefficient.

Table 16: **Calibration regimes for fixed κ.** Values and derivatives refer to the piecewise mathematical map before the final dtype cast.

Region	Returned vector	Output norm	Derivative in w
$q > \epsilon$	$(\kappa/q)w$	$\sqrt{d}\kappa$	Tangential projector $\times \kappa/q$
$q < \epsilon$	w	$\sqrt{d}q$	I
$q = \epsilon$	w	$\sqrt{d}\epsilon$	Boundary; see one-sided limits

The threshold and a branch-aware bound. For a unit vector v, the fallback value at $w = \sqrt{d}\epsilon v$ is $\sqrt{d}\epsilon v$, while the active-side limit is $\sqrt{d}\kappa v$. Their difference has norm $\sqrt{d}|\kappa - \epsilon|$. The forward map is continuous at this boundary only if $\kappa = \epsilon$; even then the radial one-sided derivatives differ. Inside the fallback region the map is the identity. These properties follow from the explicit branch rule, which returns the unnormalized vector below the threshold rather than dividing it by ϵ.

For example, an *analytical illustration* with $\kappa = 1$ and $\epsilon = 10^{-6}$ has active-side tangential gain approaching 10^6, while its active output RMS remains one. The two numbers describe derivative gain and forward scale respectively. For a complete sequence, let $\mathcal{A}$ and $\mathcal{F}$ be the active and fallback

token positions. Since $|g| < g_{\max}$,

$$\|R_\ell\|_F^2 \le g_{\max}^2 \left(\sum_{(n,t)\in\mathcal{A}} \|h_{\ell,n,t}\|_2^2 + |\mathcal{F}| d\epsilon^2 \right). \tag{36}$$

The fallback has a small absolute bound; its relative bound also depends on the receiving norm. When every position is active and $\|H_\ell\|_F > 0$, the squared norm ratio satisfies

$$\frac{\|R_\ell\|_F^2}{\|H_\ell\|_F^2} = \sum_{n,t} \frac{\|h_{\ell,n,t}\|_2^2}{\|H_\ell\|_F^2} g_{\ell,n,t}^2. \tag{37}$$

Thus the norm ratio is the square root of a hidden-energy-weighted mean of squared gates.

B.6 Stop-gradient and scale-invariant optimization

The reference RMS is detached when forming f_κ. Its forward value still tracks h, but its reverse-mode derivative is zero. On the active branch $q > \epsilon$, for $h \neq 0$ and with w held fixed, the mathematical forward map without detachment would contribute the additional Jacobian

$$\frac{\partial f_{\mathrm{RMS}(h)}(w)}{\partial h} = \frac{w}{q} \frac{h^\top}{d\kappa}. \tag{38}$$

Detachment removes this reference-scale edge. It leaves the identity path through $\widetilde{h} = h + R$, the dependence of w on h and past computations, and the dependence of g on h and P intact. On the active branch the differentiated writeback is therefore $dR = f_\kappa(w)\, dg + g J_w\, dw$, with no $d\kappa$ term.

Memory construction is not detached. A later loss can reach an earlier intervention through the ordinary residual chain, the compressed source record, and the controller that reads that record. Freezing a layer's parameters preserves its input vector–Jacobian product:

$$\frac{\partial \mathcal{L}}{\partial \widetilde{H}_\ell} = J_{\mathcal{B}_\ell}(\widetilde{H}_\ell)^\top \frac{\partial \mathcal{L}}{\partial H_{\ell+1}} \quad \text{for the backbone edge.} \tag{39}$$

Other outgoing edges add their adjoints to this expression. This is also why training earlier LoRA modules differentiates through subsequent frozen operators. Parameter freezing removes weight-gradient and optimizer-state requirements; it does not replace reverse-mode propagation by a local update at each insertion point.

Positive scaling of a receiving projection. Suppose all uses of U_ℓ remain on the active branch in a neighborhood of a positive rescaling, and the loss has no explicit norm penalty on U_ℓ. Replacing U_ℓ by aU_ℓ, $a > 0$, scales the raw w by a and leaves every calibrated intervention unchanged. Downstream residuals, records, and losses are consequently unchanged as well. Differentiating this invariance at $a = 1$ gives

$$\langle \nabla_{U_\ell} \mathcal{L}, U_\ell \rangle_F = 0. \tag{40}$$

This concerns the Euclidean gradient of the differentiated loss. AdamW adds momentum, coordinatewise preconditioning, and decoupled decay. If V is its preconditioned moment direction, η its learning rate, and $a = 1 - \eta\lambda_{\mathrm{wd}}$, the actual norm update is

$$U^+ = aU - \eta V, \qquad \|U^+\|_F^2 = a^2\|U\|_F^2 - 2a\eta\langle U, V\rangle_F + \eta^2\|V\|_F^2. \tag{41}$$

Raw-gradient orthogonality does not force $\langle U, V\rangle_F$ to vanish. Even when it vanishes, the last term is positive. For a concrete two-coordinate illustration, $U = (1, 2)$ and raw gradient $(2, -1)$ have zero inner product; a first-step Adam direction close to $(1, -1)$ has inner product -1 with U. Reversing the raw gradient reverses that inner product. The projection norm can therefore increase or decrease under the full update. Decay contracts its own component; it does not establish a monotone trajectory for the total norm.

B.7 Conditional propagation through the frozen decoder

Local calibration supplies an intervention budget at each receiving layer. Its effect at a later depth depends on the frozen computation between those layers. To make this dependence explicit, compare

an intervened trajectory H_ℓ with the adapter-disabled trajectory H_ℓ^0, starting from the same H_0. Assume $\mathcal{B}_\ell$ is Λ_ℓ-Lipschitz on a domain containing the two actual layer inputs. Put $e_\ell = \|H_\ell - H_\ell^0\|_F$. Then

$$\begin{aligned} e_{\ell+1} &\le \Lambda_\ell \|H_\ell + R_\ell - H_\ell^0\|_F \\ &\le \Lambda_\ell (e_\ell + \|R_\ell\|_F), \\ e_L &\le \sum_{j=0}^{L-1} \left(\prod_{k=j}^{L-1} \Lambda_k \right) \|R_j\|_F. \end{aligned} \tag{42}$$

The final inequality follows by repeatedly substituting the preceding one and using $e_0 = 0$. Crucially, it treats each R_j as the intervention realized on the actual trajectory. Routing can depend on the entire available auxiliary history; no assumption that R_j is a function of H_j alone is needed.

When all positions use active calibration and $|g_{j,n,t}| \le \rho_j$, we have $\|R_j\|_F \le \rho_j \|H_j\|_F$. One can either substitute this directly into Equation 42, or eliminate the intervened norm using $\|H_j\|_F \le \|H_j^0\|_F + e_j$. The latter gives

$$e_L \le \sum_{j=0}^{L-1} \Lambda_j \rho_j \|H_j^0\|_F \prod_{k=j+1}^{L-1} \Lambda_k (1 + \rho_k). \tag{43}$$

The empty product is one. This follows from the scalar recurrence $e_{j+1} \le \Lambda_j(1+\rho_j)e_j + \Lambda_j \rho_j \|H_j^0\|_F$. Fallback positions can instead be handled with the absolute budget in Equation 36.

For an analytical illustration, if the realized intervention norm is at most δ at each of n successive layers and $\Lambda_j \le \Lambda$, Equation 42 becomes $\delta \sum_{k=1}^{n} \Lambda^k$. It equals $n\delta$ when $\Lambda = 1$, is at most $\delta\Lambda/(1-\Lambda)$ when $\Lambda < 1$, and grows geometrically with n when $\Lambda > 1$. Thus the same local budget can have different downstream effects depending on the intervening maps. These conditional bounds explain why local relative scale and final-output change are separate quantities.

There is an analogous first-order description. Introduce a scalar perturbation amplitude τ with $R_j(\tau) = \tau V_j + o(\tau)$, $R_j(0) = 0$, and differentiable frozen layers at the unperturbed trajectory. Differentiating $H_{j+1}(\tau) = \mathcal{B}_j(H_j(\tau) + R_j(\tau))$ gives

$$\dot{H}_L = \sum_{j=0}^{L-1} J_{L-1} J_{L-2} \cdots J_j V_j, \qquad J_j = J_{\mathcal{B}_j}(H_j^0). \tag{44}$$

Each interface direction is transported by the frozen Jacobians that follow it. The product acts on a low-dimensional injected vector, while different layers can inject through different receiving subspaces. This is the precise sense in which changing cross-layer communication can alter later computation without changing the pretrained operators themselves.

B.8 The differentiated group-relative objective

For a retained group of $G = 4$ completions, let $\mathcal{T}_i$ contain its scored completion positions and $T_i = \max\{1, |\mathcal{T}_i|\}$. Prompt and padding positions are excluded from $\mathcal{T}_i$. Let V omit completions that are simultaneously truncated and unparseable; other incorrect completions are valid and receive binary reward zero. For $m = |V| > 0$, write

$$\bar{r} = \frac{1}{m} \sum_{i \in V} r_i, \quad \sigma^2 = \frac{1}{m} \sum_{i \in V} (r_i - \bar{r})^2, \quad A_i = \begin{cases} (r_i - \bar{r})/(\sigma + 10^{-6}), & i \in V,\ \sigma > 10^{-6}, \\ r_i - \bar{r}, & i \in V,\ \sigma \le 10^{-6}, \\ 0, & i \notin V. \end{cases} \tag{45}$$

For $V = \varnothing$, all advantages and the policy term are zero. Centering gives $\sum_{i \in V} A_i = 0$. For mixed binary rewards with c correct among m valid completions, $\sigma^2 = (c/m)(1 - c/m)$. Ignoring the displayed additive 10^{-6} only for this interpretation, correct and incorrect advantages are $\sqrt{(m-c)/c}$ and $-\sqrt{c/(m-c)}$ respectively. With the additive term retained their magnitudes are slightly smaller, and $|A_i| \le \sqrt{m-1} \le \sqrt{3}$. Uniform valid rewards produce zero advantages.

Put $\ell_{i,t} = \log \pi_\theta(y_{i,t} \mid x, y_{i,<t})$ and let $\ell^0_{i,t}$ be the corresponding adapter-disabled log probability. The differentiated surrogate uses a detached copy from the current evaluation,

$$\begin{aligned} q_{i,t} &= \exp(\ell_{i,t} - \mathrm{sg}(\ell_{i,t}))\,, \\ \mathcal{L}_{\mathrm{policy}} &= -\frac{1}{m}\sum_{i\in V}\frac{1}{T_i}\sum_{t\in\mathcal{T}_i}\min(q_{i,t}A_i, \mathrm{clip}(q_{i,t}, 0.8, 1.2)A_i)\,, \\ \mathcal{L}_{\mathrm{KL}} &= \frac{1}{\sum_i |\mathcal{T}_i|}\sum_i\sum_{t\in\mathcal{T}_i}[\exp(-\delta_{i,t}) + \delta_{i,t} - 1]\,, \quad \delta_{i,t} = \ell_{i,t} - \ell^0_{i,t}. \end{aligned} \tag{46}$$

The token-level KL reduction is defined for a group with at least one scored completion token. Every retained group receives one differentiable policy evaluation; accumulating gradients from multiple groups does not change the denominator construction.

Gradient equivalence to sequence-normalized REINFORCE. In the forward evaluation $q_{i,t} = 1$, while its derivative is $\nabla_\theta q_{i,t} = \nabla_\theta \ell_{i,t}$. The value one lies strictly inside the clipping interval. In a neighborhood with the denominator held fixed, both arguments of the minimum therefore coincide and have the same derivative, irrespective of the sign of A_i. It follows that

$$\nabla_\theta \mathcal{L}_{\mathrm{policy}} = \nabla_\theta\left[-\frac{1}{m}\sum_{i\in V}\frac{\mathrm{sg}(A_i)}{T_i}\sum_{t\in\mathcal{T}_i}\ell_{i,t}\right]. \tag{47}$$

This is equality of gradients for the retained observations and advantages. It is not equality of the two scalar loss values. When every valid completion has at least one scored token, the implemented policy value is $-m^{-1}\sum_{i\in V} A_i = 0$ even when its gradient is nonzero. Its computational graph carries the gradient through the numerator despite cancellation of the evaluated values. A finite-difference check of this derivative must freeze the denominator at the expansion point; rebuilding a detached denominator separately at each perturbed point instead differentiates a constant-valued forward expression.

Sequence normalization gives each valid response one outer weight and averages its token scores before that weighting. The KL term uses a different reduction: each scored token has the same weight across all retained completions, including those excluded from V. Group standardization, validity filtering, sampling, and sequence normalization are therefore all part of the update estimator; Equation 47 does not replace them by an unnormalized expected-reward objective.

Sampled divergence geometry. For $k(\delta) = e^{-\delta} + \delta - 1$, $k \geq 0$, $k'(\delta) = 1 - e^{-\delta}$, and $k''(\delta) = e^{-\delta} > 0$. The minimum is at $\delta = 0$, with expansion $k(\delta) = \delta^2/2 + O(\delta^3)$. For a fixed prefix and full-support distributions p, p_0,

$$\mathbb{E}_{y\sim p} k\left(\log\frac{p(y)}{p_0(y)}\right) = \sum_y p(y)\log\frac{p(y)}{p_0(y)} = D_{\mathrm{KL}}(p\|p_0). \tag{48}$$

The exponential term integrates to $\sum_y p_0(y) = 1$. The optimizer, however, differentiates the sampled expression with the observed token fixed. Under a general sampling distribution q, that conditional expected derivative is $\sum_y q(y)(1 - p_0(y)/p(y))\nabla_\theta \log p(y)$. This specifies the differentiated penalty independently of the value identity. Temperature, nucleus filtering, and group selection determine the actual sampling distribution, while the evaluated p uses full-vocabulary, untempered logits.

B.9 Routing regularization and its reduction

For the canonical 28 receiving layers, the complete auxiliary objective is $\mathcal{L} = \mathcal{L}_{\mathrm{policy}} + 0.02\mathcal{L}_{\mathrm{KL}} + 0.01\Omega$, with

$$\Omega = \frac{1}{28}\sum_{\ell=4}^{31}\left[\mathrm{mean}_{n,t,b}\left(a_{\ell,n,t,b} - \frac{1}{n_\ell}\right)^2 + 0.05\,\mathrm{mean}_{n,t,k}\, c^2_{\ell,n,t,k}\right]. \tag{49}$$

The first mean includes the n_ℓ visible sources; the second includes r source-mixture coordinates. Both range over all BT tensor positions. For one position with n sources, define $\psi(a) = n^{-1}\sum_b (a_b - 1/n)^2$. Since $\sum_b a_b = 1$,

$$\psi(a) = \frac{1}{n}\left(\|a\|_2^2 - \frac{1}{n}\right), \qquad 0 \leq \psi(a) \leq \frac{n-1}{n^2}. \tag{50}$$

The minimum is uniform allocation; the upper bound is reached at a simplex vertex and approached by dense softmax as logits separate. For $n = 1$, the function and its logit gradient are identically zero. Thus at layers 4–7 the source-mixture magnitude term contributes while the routing-deviation term does not. This follows from source availability, not from a special early-layer regularization mask.

The source-coordinate mean also means that the maximum per-position deviation penalty depends on n: it is $1/4$ for two sources and $6/49$ for seven. Its gradient with respect to logits z is $\nabla_z \psi = \frac{2}{n}(\operatorname{diag} a - aa^\top)(a - \mathbf{1}/n)$. The penalty acts on selection coefficients, while the source-mixture term acts on the mixture before U_ℓ and RMS calibration. Neither directly penalizes the norm of the final calibrated writeback projection.

To make the position reduction explicit, partition the BT positions into prompt, completion, and padding sets $\mathcal{P}, \mathcal{C}, \mathcal{Z}$. For either scalar positionwise penalty f,

$$\frac{1}{BT}\sum_{n,t} f_{n,t} = \sum_{Q\in\{\mathcal{P},\mathcal{C},\mathcal{Z}\}} \frac{|Q|}{BT}\,\mathrm{mean}_{(n,t)\in Q}\, f_{n,t}, \tag{51}$$

where empty-set contributions are zero. This identity describes the implemented full-tensor objective: each position set contributes in proportion to its size. It also separates this reduction from the completion masks used by the two language-model loss terms.

B.10 Bounded informative resampling and generation budgets

For a fixed prompt and fixed model parameters during sampling, let Y denote a group drawn from the generation procedure, and let $I(Y)$ indicate that at least one completion is parsed correct and at least one is not. Write $q = \Pr(I = 1)$ and let A be the maximum number of group attempts. The procedure retains the first mixed-correctness group; if no earlier attempt is mixed, it retains attempt A regardless of its outcome. The configured values are $G = 4$ and $A = 4$ (three additional attempts).

Assume attempts are independent and identically distributed conditional on the prompt. With N the number of attempts, the tail probabilities and expectation are

$$\Pr(N \ge j) = (1-q)^{j-1}, \quad 1 \le j \le A, \qquad \mathbb{E}N = \sum_{j=0}^{A-1}(1-q)^j = \frac{1-(1-q)^A}{q}. \tag{52}$$

The quotient has continuous limit A at $q = 0$. This follows because reaching attempt j requires exactly that the preceding $j-1$ attempts fail the mixed-correctness test. The terminal probability is $\Pr(N = A) = (1-q)^{A-1}$, which includes both success and failure on the final attempt.

Let μ denote the original group law and μ_{ret} the retained law. For any group event E and $0 < q < 1$,

$$\begin{aligned} \mu_{\mathrm{ret}}(E) &= \frac{1-(1-q)^A}{q}\,\mu(E\cap I) + (1-q)^{A-1}\mu(E\cap I^c), \\ \mu_{\mathrm{ret}} &= [1-(1-q)^A]\,\mu(\cdot \mid I) + (1-q)^A\,\mu(\cdot \mid I^c). \end{aligned} \tag{53}$$

To prove the first line, sum $(1-q)^{j-1}\mu(E\cap I)$ over successful retention at each $j = 1, \dots, A$, then add $(1-q)^{A-1}\mu(E\cap I^c)$ for a failed final attempt. The second line groups these terms into normalized conditional distributions. Bounded retry increases the mass of mixed groups while preserving a nonzero failed-group mass $(1-q)^A$ whenever $q < 1$.

The retry event precedes preference-validity filtering. Let J denote a group whose *valid* completions include both reward values. Then $J \subseteq I$, but equality need not hold: a group containing correct answers and truncated-unparseable failures is mixed before filtering and can become uniform afterwards. If $r = \Pr(J)$, Equation 53 gives

$$\Pr_{\mathrm{ret}}(J) = r\frac{1-(1-q)^A}{q}. \tag{54}$$

For binary rewards, this is the probability of a retained group with nonzero advantages. It distinguishes the retry acceptance event from producing a nonconstant set of preference coefficients; the resulting parameter gradient also depends on the token-score derivatives.

In the independent Bernoulli illustration, $q = 1 - p^G - (1-p)^G$. It is symmetric under $p \mapsto 1-p$: both nearly always-correct and nearly always-incorrect prompts can consume additional attempts. The procedure therefore selects correctness diversity within groups, rather than preferring high accuracy by itself.

Table 17: **Analytical illustration of bounded retry.** Four independent Bernoulli-correctness completions per attempt, all preference-valid, with four attempts allowed. Values are computed from $q = 1 - p^4 - (1 - p)^4$.

Per-completion correctness p	Mixed q	Retained mixed	Expected attempts	Expected responses
0.1	0.3438	0.8146	2.3694	9.4774
0.5	0.8750	0.9998	1.1426	4.5703
0.9	0.3438	0.8146	2.3694	9.4774

Responses, tokens, and retained updates. Every prompt yields one retained group but incurs between G and GA generated responses, with expectation $G\mathbb{E}N$. Let C_j be the completion-token cost of attempt j, identically distributed jointly with its outcome. Reaching attempt j depends only on earlier outcomes and is independent of C_j. Therefore

$$\mathbb{E}\left[\sum_{j=1}^{N} C_j\right] = \sum_{j=1}^{A} \Pr(N \geq j)\mathbb{E}C_1 = \mathbb{E}N\,\mathbb{E}C_1. \tag{55}$$

This equality does not require the cost of an attempt to be independent of its own correctness outcome. The expected cost of discarded attempts, for $0 < q < 1$, is

$$\mathbb{E}C_{\text{discarded}} = \mathbb{E}[C_1 \mid I^c](1 - q) \sum_{j=0}^{A-2} (1 - q)^j. \tag{56}$$

Only failed nonterminal attempts are discarded. Retained mixed and failed groups can have different length distributions, as made explicit by Equation 53.

For D prompts and a cap of M new tokens per completion, the deterministic bounds are $DG \leq N_{\text{responses}} \leq DGA$ and $N_{\text{generated tokens}} \leq DGAM$. For one pass over $D = 7{,}473$ prompts with $G = A = 4$, the response budget is 29,892–119,568 completions and the token bound is $119{,}568M$. The number of differentiated retained groups remains 7,473. These count generation separately from teacher-forced scoring and from gradient accumulation. They make precise why prompt count, optimizer updates, and generated-token budget characterize different aspects of the same RL procedure.

C COMPLETE EMPIRICAL RESULTS AND DESIGN ANALYSIS

This appendix develops the evidence for parameter-efficient computation reuse. Method comparisons pair reasoning quality with trainable allocation; component comparisons examine how that capacity is organized; writeback diagnostics and configuration profiles characterize the resulting task responses. Generative scores are means ± sample standard deviations (SD) across three evaluation seeds, 42, 43, and 44. Bold/underlined means identify the best/second-best values within each column, including ties; they describe the ordering of means. Lower values are preferred for resource costs and perplexity. Historical configurations retain their † estimate attribute, and ‡ marks original non-generative point references. The complete sample variances appear in Appendix C.12.

C.1 METHOD COMPARISONS ACROSS MATHEMATICS AND SEARCH

For each evaluation seed, AIME mean averages the two annual scores and MathAvg gives one-third weight to GSM8K, MATH-500, and AIME mean. The reported aggregate SD is computed across those seed-level aggregates. Differences likewise pair the same evaluation-seed indices before taking their mean and SD. Table 18 completes the annual and aggregate profiles in Table 1; Table 19 retains the complete search comparison.

Table 18: **Annual mathematics results and aggregate contrasts.** Accuracy and MathAvg are percentages; SD and Δ are percentage points. Δ is the paired MathAvg difference from full-parameter GRPO. GSM8K, MATH-500, and parameter allocations appear in Table 1.

Method	AIME24	AIME25	AIME mean	MathAvg	Δ vs GRPO
Frozen base	30.00 ±5.77	33.33 ±3.33	31.67 ±3.33	65.33 ±0.89	−8.46 ±0.95
Full-parameter GRPO	44.44 ±6.94	44.44 ±3.85	44.44 ±5.36	73.79 ±1.83	0.00 ±0.00
RFT	<u>51.11</u> ±10.18	41.11 ±13.47	46.11 ±2.55	73.64 ±1.33	−0.15 ±2.54
LoRA-r2 + GRPO	43.33 ±17.32	41.11 ±10.72	42.22 ±12.73	73.42 ±4.28	−0.37 ±2.67
LoRA-r16 + GRPO	50.00 ±5.77	41.11 ±6.94	45.56 ±6.31	74.83 ±1.41	+1.04 ±2.69
LoRA-MoE + RO-GRPO	41.11 ±5.09	47.78 ±5.09	44.44 ±2.55	75.74 ±1.47	+1.95 ±0.41
LoRA + S-GRPO	<u>51.11</u> ±19.53	46.67 ±3.33	<u>48.89</u> ±8.22	75.95 ±2.43	+2.16 ±2.30
LoRA-r16, matched retries	45.56 ±7.70	<u>51.11</u> ±12.62	48.33 ±6.01	<u>77.28</u> ±1.95	<u>+3.49</u> ±3.76
T-Router + GRPO	**63.33** ±6.67	**57.78** ±1.92	**60.56** ±3.47	**83.64** ±1.16	**+9.85** ±2.93

Table 19: **Complete agentic search comparison.** BrowseComp Plus uses token-F1 multiplied by 100; ASearch Test denotes the ASearcher held-out validation score (0–100). Appendix A.7 details the scorer and default penalty factors. Δ is the paired difference from full-parameter GRPO; all entries are mean ± SD in the corresponding metric's points.

Method	BrowseComp F1	Δ vs GRPO	ASearch	Δ vs GRPO
Frozen base	5.11 ±0.30	−28.12 ±1.25	59.76 ±0.41	−10.19 ±0.56
Full-parameter GRPO	33.23 ±1.54	0.00 ±0.00	69.94 ±0.53	0.00 ±0.00
RFT	31.20 ±0.75	−2.03 ±2.07	65.81 ±0.92	−4.14 ±1.18
LoRA-r2 + GRPO	32.98 ±0.56	−0.25 ±1.25	69.19 ±0.59	−0.75 ±1.01
LoRA-r16 + GRPO	36.06 ±0.56	+2.83 ±1.80	70.59 ±0.18	+0.64 ±0.64
LoRA-MoE + RO-GRPO	**37.00** ±1.17	**+3.77** ±2.56	71.46 ±0.55	+1.52 ±0.65
LoRA + S-GRPO	35.50 ±0.57	+2.27 ±1.58	70.67 ±0.19	+0.73 ±0.34
LoRA-r16, matched retries	36.90 ±1.13	+3.67 ±1.32	<u>72.80</u> ±0.75	<u>+2.86</u> ±0.63
T-Router + GRPO	<u>36.98</u> ±1.16	<u>+3.75</u> ±0.91	**73.98** ±0.74	**+4.04** ±1.27

Against full-parameter GRPO, T-Router improves MathAvg by 9.85 ± 2.93 points with a trainable allocation equal to 0.466% of the backbone. At a similar adapter budget, matched-retry LoRA records 77.28 ± 1.95 versus T-Router's 83.64 ± 1.16, a 6.36-point difference in means. The corresponding search means are 36.90 versus 36.98 on BrowseComp Plus and 72.80 versus 73.98 on ASearch. LoRA-MoE records 37.00 BrowseComp F1 with 173.112M trainable parameters; T-Router records 36.98 with 41.730M. Together these profiles establish a favorable quality–allocation tradeoff and the strongest ASearch mean in the comparison.

C.2 COMPLETE COMPONENT ABLATIONS

Table 20 retains all four task scores and the paired aggregate contrast for each component change. The no-state alternative uses a capacity-matched MLP, allowing the controller's role to be assessed alongside its allocation.

Table 20: **Complete task-level component ablations.** Entries are mean ± SD across evaluation seeds. Δ is each variant minus full T-Router in MathAvg percentage points.

Configuration	GSM8K	MATH	AIME24	AIME25	MathAvg	Δ
Full T-Router	**97.62** ±0.79	**92.73** ±0.70	**63.33** ±6.67	**57.78** ±1.92	**83.64** ±1.16	**0.00** ±0.00
No cross-layer retrieval	96.26 ±0.59	79.40 ±1.56	14.44 ±3.85	51.11 ±15.03	69.48 ±2.36	−14.16 ±1.78
Hidden-state memory	96.79 ±0.53	77.80 ±1.31	37.78 ±10.72	17.78 ±5.09	67.46 ±2.19	−16.18 ±2.96
Learned static routing	96.01 ±0.68	77.27 ±0.81	26.67 ±5.77	16.67 ±6.67	64.98 ±1.71	−18.66 ±2.78
No state (matched MLP)	95.02 ±0.75	83.93 ±1.92	50.00 ±17.64	28.89 ±10.18	72.80 ±4.10	−10.84 ±3.09
No layer/block identity	94.29 ±0.42	66.13 ±1.80	11.11 ±10.72	16.67 ±3.33	58.10 ±1.06	−25.53 ±0.56
No RMS alignment	94.57 ±0.64	67.93 ±3.80	3.33 ±3.33	10.00 ±3.33	56.39 ±0.41	−27.25 ±1.24
No token-conditioned gate	96.26 ±0.42	66.73 ±1.81	34.44 ±8.39	22.22 ±11.71	63.78 ±2.61	−19.86 ±1.47
No informative retries	94.39 ±0.55	74.87 ±2.55	14.44 ±1.92	22.22 ±3.85	62.53 ±1.20	−21.11 ±2.26

Computation content and addressed retrieval. Full T-Router reaches 83.64 ± 1.16 MathAvg. Removing cross-layer retrieval gives 69.48 ± 2.36, and replacing completed block changes with hidden states gives 67.46 ± 2.19. These changes distinguish access to earlier computations from the representation stored in memory. The corresponding MATH-500 means are 92.73, 79.40, and 77.80. Source/layer identity identifies the origin and destination of each exchange; its removal has a paired MathAvg difference of -25.53 ± 0.56 points.

History, selection, and relative influence. The matched MLP records 72.80 ± 4.10 MathAvg and learned static routing records 64.98 ± 1.71, supporting the organization of capacity into recurrent depth history and query-dependent selection. Removing RMS alignment gives a paired difference of -27.25 ± 1.24 points, while removing the token-conditioned gate gives -19.86 ± 1.47. Calibration sets the relative intervention scale; the signed gate adjusts that influence to the current token. The complete interface learns which computations to reuse and how strongly to incorporate them.

C.3 PARAMETER AND TRAINING-COST PROFILES OF THE ABLATIONS

Table 21 pairs the same configurations with exact allocated parameter counts and their original GPU-hour records. The common estimated optimizer-step count separates update count from response generation and the computation performed within an update.

Table 21: **Resources for the component study.** Parameter counts are exact allocations; steps retain their estimated status. GPU-hours retain their original training-cost records, separate from the evaluation-seed statistics.

Configuration	Parameters	Steps (est.)	GPU-hours
Full T-Router	41,730,332	3,737	121.23
No cross-layer retrieval	41,648,924	3,737	110.97
Hidden-state memory	41,730,332	3,737	122.42
Learned static routing	**40,535,324**	3,737	114.90
No state (matched MLP)	41,731,978	3,737	69.25
No layer/block identity	41,729,052	3,737	77.34
No RMS alignment	41,730,332	3,737	117.70
No token-conditioned gate	41,725,980	3,737	123.89
No informative retries	41,730,332	3,737	**37.92**

The MLP allocates 41,731,978 parameters, 1,646 more than full T-Router's 41,730,332. Its paired MathAvg difference is -10.84 ± 3.09 points. Hidden-state memory and the no-RMS variant retain exactly the full allocation; removing the token-conditioned gate changes it by 4,352 parameters. These comparisons link accuracy to the organization of trainable capacity.

Informative retries address a complementary training dimension. The no-retry configuration records 37.92 GPU-hours and 62.53 ± 1.20 MathAvg, while full T-Router records 121.23 GPU-hours and 83.64 ± 1.16. Their paired MathAvg difference is 21.11 ± 2.26 points in favor of the full configuration. Retry generation changes the completions available to each prompt's update while retaining the same estimated optimizer-step count, connecting the empirical comparison to the correctness diversity in Appendix B.

C.4 Configuration landscape and annual task profiles

The historical exploration varies writeback rules, controller structure, and optimization settings. Table 22 preserves every annual score from its initialization and structural branch, while Table 23 records the complete writeback sweep. The historical † attribute applies to GSM8K, MATH-500, and their aggregates; evaluation-seed dispersion does not change that provenance. Full T-Router is the final-model reference.

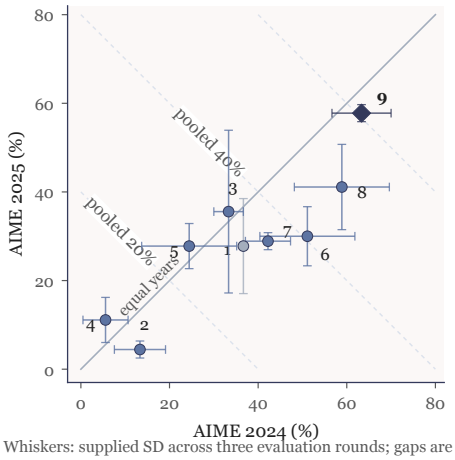
Table 22: **Complete historical task profiles and final reference.** Generative entries are mean ± SD; † retains the historical IID-estimate attribute. The initialization and local/controller variants use the standard-gate branch. Full T-Router is the final configuration from Table 1.

Configuration	GSM8K	MATH-500	AIME24	AIME25	MathAvg
Standard gate†	87.57 ±0.87	84.87 ±3.29	36.67 ±0.00	27.78 ±10.72	68.22 ±2.83
Standard gate, no KL†	89.03 ±0.58	71.67 ±4.02	13.33 ±5.77	4.44 ±1.92	56.53 ±1.72
Local16†	90.75 ±1.12	82.00 ±1.00	33.33 ±3.33	35.56 ±18.36	69.07 ±2.74
Local64†	89.54 ±0.57	71.07 ±2.57	5.56 ±5.09	11.11 ±5.09	56.31 ±2.48
Wide512†	87.57 ±1.18	84.60 ±1.93	24.44 ±10.72	27.78 ±5.09	66.09 ±2.85
Zero initialization†	89.03 ±0.35	83.87 ±1.81	51.11 ±10.72	30.00 ±6.67	71.15 ±2.29
Orthogonal initialization†	90.83 ±0.08	82.93 ±1.72	42.22 ±5.09	28.89 ±1.92	69.77 ±0.38
Top2 composite†	96.64 ±0.29	87.20 ±1.06	58.89 ±10.72	41.11 ±9.62	77.95 ±2.12
Full T-Router	**97.62** ±0.79	**92.73** ±0.70	**63.33** ±6.67	**57.78** ±1.92	**83.64** ±1.16

Full T-Router has the largest mean on all four tasks in this panel. Relative to Top2, the approximately 5.69-point MathAvg difference comprises 0.33 point from GSM8K, 1.84 from MATH-500, and 3.52 from pooled AIME. The annual means are 63.33 and 57.78 for full T-Router, compared with 58.89 and 41.11 for Top2. Zero initialization has the largest annual mean gap in this panel, 21.11 points. These annual gaps compare two task sets; their seed-paired SDs are not among the reported statistics.

Zero and orthogonal initialization modify the standard-gate branch; the orthogonal gain is 0.064, whereas RMS configurations use gain 1. Local16 and Local64 change the additional token-local branch rank; Wide512 changes controller dimensions; Top2 denotes the composite sparse-routing configuration specified in Appendix C.8.

a Annual accuracy ± SD



Whiskers: supplied SD across three evaluation rounds; gaps are mean differences.

b Annual means (%)

		2024	2025	Gap
1	Standard gate†	36.67	27.78	8.89
2	Standard, no KL†	13.33	4.44	8.89
3	Local16†	33.33	35.56	-2.23
4	Local64†	5.56	11.11	-5.55
5	Wide512†	24.44	27.78	-3.34
6	Zero init.†	51.11	30.00	21.11
7	Orthogonal init.†	42.22	28.89	13.33
8	Top2 composite†	58.89	41.11	17.78
9	**T-Router, init. 2%**	63.33	57.78	5.55

Figure 7: **Annual competition-mathematics profiles.** The two annual means locate each configuration; whiskers show supplied SDs, and numerical summaries give annual means and their gaps. Pooled means average AIME24 and AIME25, and annual gaps are differences of those means. Each annual set contains 30 questions per evaluation seed.

C.5 Benchmark contributions to aggregate differences

MathAvg admits an additive decomposition. For mean family score $s_j(c)$ and standard-gate reference c_0,

$$\Delta\mathrm{MathAvg}(c, c_0) = \frac{s_{\mathrm{G}}(c) - s_{\mathrm{G}}(c_0)}{3} + \frac{s_{\mathrm{M}}(c) - s_{\mathrm{M}}(c_0)}{3} + \frac{s_{\mathrm{A}}(c) - s_{\mathrm{A}}(c_0)}{3}. \tag{57}$$

Figure 8 applies this identity to the displayed mean profiles. These decompositions and the later reweighting analyses use supplied rounded means; small rounding differences from separately reported aggregates can therefore occur.

Full T-Router's approximately 15.42-point difference from the standard gate comprises 3.35 points from GSM8K, 2.62 from MATH-500, and 9.45 from pooled AIME. Top2's approximately 9.73-point difference comprises 3.02, 0.78, and 5.93 points. Both distribute gains across all three mathematical families, with the full design adding a stronger MATH-500 and AIME profile.

Local16 contributes approximately $(+1.06, -0.96, +0.74)$ points, while Local64 contributes $(+0.66, -4.60, -7.96)$. The no-KL branch contributes $(+0.49, -4.40, -7.78)$. Zero and orthogonal initialization produce net mean differences of approximately $+2.93$ and $+1.55$, respectively. These task-level views characterize the effect of complete settings alongside their aggregate order.

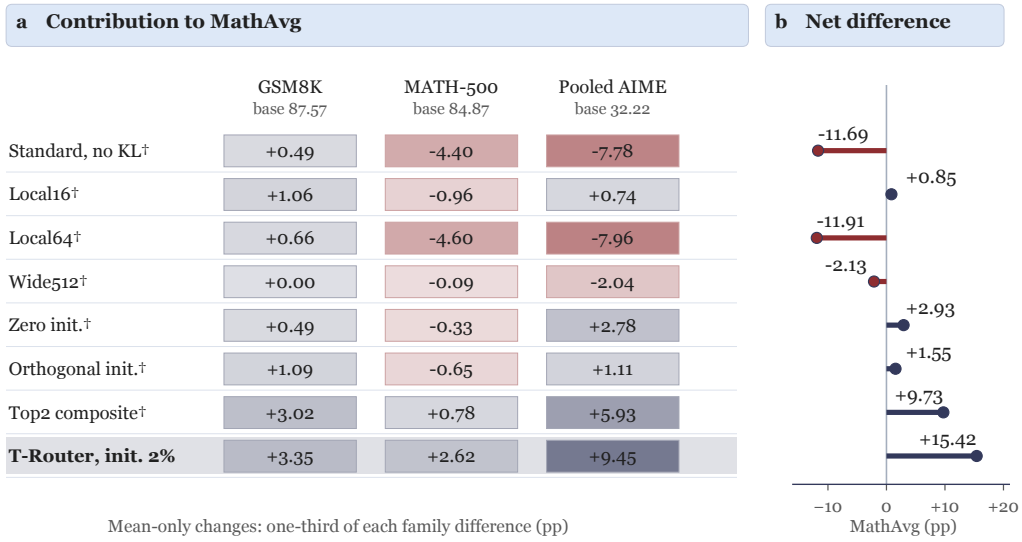


Figure 8: **Where aggregate mean differences occur.** Each family contributes one-third of its signed difference from the standard gate. Navy/dark red indicate positive/negative contributions; the net column sums those mean contrasts. These are analytical decompositions of the same reported evaluations.

C.6 Writeback configurations and task response

The writeback study connects relative intervention scale to the quality of the reused computation. The standard gate learns a coefficient without RMS alignment; fixed and learned RMS configurations express writeback on a common relative-scale basis. Table 23 contains every reported family score, aggregate, and endpoint diagnostic.

Table 23: **Complete writeback exploration and final-model reference.** Scores are mean ± SD; † retains historical estimates. Final WB and tail median are original percentage-valued endpoint diagnostics. A dash denotes an unavailable trailing median.

Configuration	GSM8K	MATH	AIME mean	MathAvg	Final WB	Tail
Standard gate†	87.57 ±0.87	84.87 ±3.29	32.22 ±5.36	68.22 ±2.83	0.1254	—
Fixed RMS 0.5%†	87.47 ±0.99	76.20 ±1.59	35.56 ±3.47	66.41 ±1.69	0.5005	—
Fixed RMS 1%†	89.31 ±0.67	79.67 ±3.11	40.00 ±5.00	69.66 ±2.34	1.0010	—
Fixed RMS 2%†	96.99 ±0.56	87.60 ±2.11	47.22 ±2.55	77.27 ±1.16	2.0020	—
Learned RMS, init 0.5%†	96.99 ±0.29	84.67 ±0.99	55.00 ±8.33	78.89 ±2.39	3.4987	3.7596
Learned RMS, init 1%†	**97.83** ±0.46	84.07 ±1.67	55.00 ±12.02	78.96 ±4.40	3.6535	3.8762
Full T-Router	97.62 ±0.79	**92.73** ±0.70	**60.56** ±3.47	**83.64** ±1.16	4.0211	4.2065

Full T-Router records 83.64 ± 1.16 MathAvg at 4.0211% final writeback, combining a 92.73 MATH-500 mean with 60.56 pooled AIME. Learned RMS initialized at 0.5% and 1% records MathAvg means of 78.89 and 78.96, with pooled AIME means of 55.00 for both. These task profiles connect writeback scale to the computations delivered to the receiving layer.

C.7 Writeback coefficients and endpoint diagnostics

The standard gate has a 20% bound and 5% initial coefficient, with small-normal writeback projections. Fixed RMS uses gain-1 orthogonal projections and constant coefficients of 0.5%, 1%, or 2%. Learned RMS uses the same gain-1 projection initialization, a 5% gate bound, and initial coefficients of 0.5%, 1%, or 2%. A fixed coefficient retains input-dependent source routing.

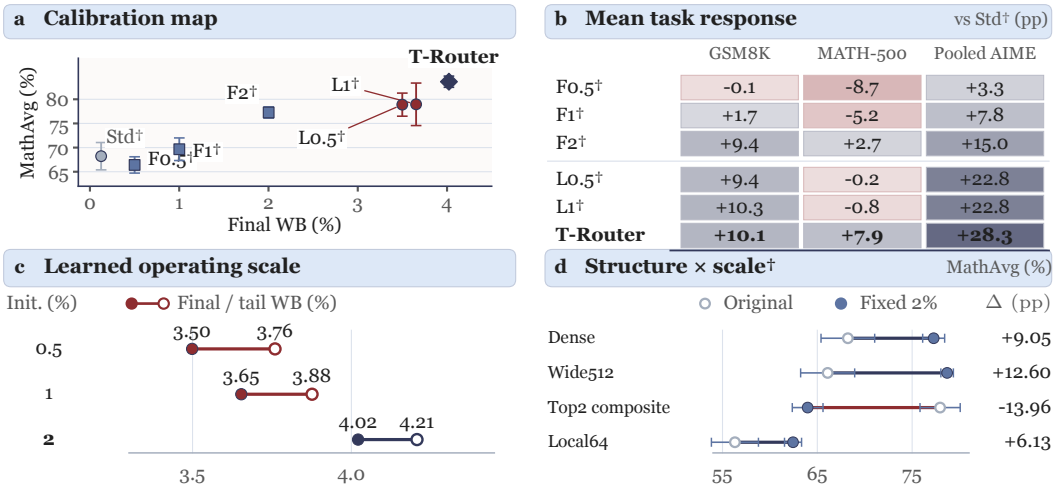


Figure 9: **Writeback scale, task response, and structure.** (a) MathAvg versus final relative writeback. (b) Mean family-score differences from the standard gate. F/L denote fixed/learned RMS, with coefficients or initializations in percent. (c) Original final and trailing-median diagnostics. (d) Structure-dependent mean changes under fixed 2% RMS. Whiskers in (a,d) show evaluation-seed SD. Tables 23 and 24 give the complete statistics.

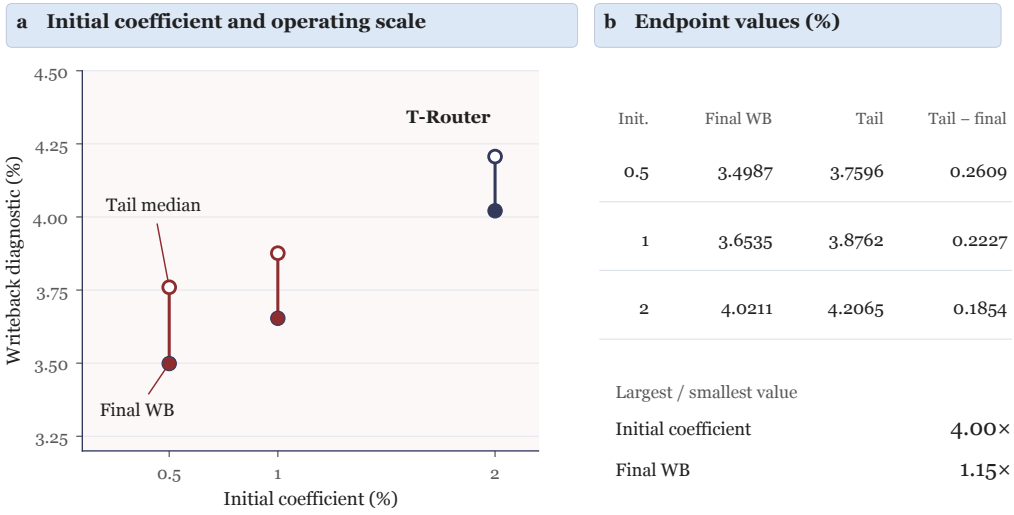


Figure 10: **Writeback endpoint summaries.** Initial coefficients are paired with Final WB (filled circles) and trailing-median WB (open circles). The table gives the original scalar diagnostics, their differences, and the largest-to-smallest initial and final ratios. Final WB is the latest routed layer's whole-tensor writeback-to-residual norm ratio.

For the 0.5%, 1%, and 2% learned initializations, tail medians are 3.7596%, 3.8762%, and 4.2065%, exceeding final diagnostics by 0.2609, 0.2227, and 0.1854 points. The final scales lie closer together than the initial coefficients. Across the broader comparison, final scale alone does not order accuracy: fixed RMS 0.5% has a larger final diagnostic than the standard gate, while their MathAvg means are 66.41 and 68.22. Retrieval direction, the receiving representation, and scaling jointly determine the resulting computation.

C.8 STRUCTURE–WRITEBACK COMBINATIONS

The interaction study pairs four structures with their original writeback and fixed 2% RMS. Table 24 preserves the mean and SD of each paired original-to-fixed change, retaining each structure's routing and state computation.

Table 24: **Structure–writeback combinations.** MathAvg is in percent; SD and paired changes are in percentage points; † retains the historical estimate attribute. Change is fixed RMS minus original writeback, paired by evaluation-seed index.

Structure	Original writeback	Fixed RMS 2%	Paired change
Dense, standard gate†	68.22 ±2.83	77.27 ±1.16	+9.05 ±2.26
Wide512†	66.09 ±2.85	**78.69** ±0.66	**+12.60** ±3.41
Top2 composite†	**77.95** ±2.12	63.98 ±1.62	−13.96 ±1.95
Local64†	56.31 ±2.48	62.44 ±0.90	+6.13 ±1.62

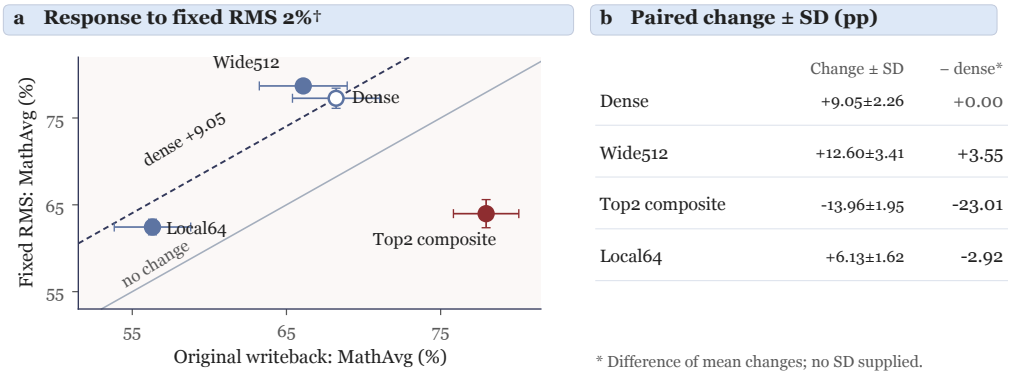


Figure 11: **Structure-dependent response to fixed RMS writeback.** Original and fixed-RMS scores in (a) show means and evaluation SDs. The dashed reference adds the dense row's 9.05-point change. The paired changes use supplied SDs; differences between those changes are descriptive contrasts of means.

Dense routing and Wide512 change by $+9.05 \pm 2.26$ and $+12.60 \pm 3.41$ points; the difference between these means is $+3.55$. Local64 changes by $+6.13 \pm 1.62$, or -2.92 relative to the dense response. Top2 changes by -13.96 ± 1.95, giving a -23.01 mean contrast relative to dense routing. The signs show that a configuration's original quality and its response to fixed calibration describe distinct aspects of the design space.

Meaning of the structure names. Local16 and Local64 add a token-local branch of rank 16 or 64, a controller-conditioned sigmoid gate initialized at 0.2, and a branch learning-rate multiplier of two; cross-layer memory rank remains 256. Wide512 sets slot and controller hidden dimensions to 512, with eight slots and memory rank 256. Top2 restricts source and slot selection to two entries, RMS-normalizes stored deltas, sets routing temperature to 0.7, and uses routing-sharpness, balance, and slot-diversity terms. Its result therefore characterizes that complete configuration.

Wide512 with fixed RMS records 78.69 ± 0.66 MathAvg with 45.546M allocated parameters, versus 83.64 ± 1.16 and 41.730M for the selected learned-RMS design. The 3.816M allocation difference accompanies a 4.95-point difference in mean MathAvg. Allocation remains distinct from active memory and arithmetic during execution.

C.9 AUXILIARY TASK PROFILES

The auxiliary tasks describe narrative reasoning (MuSR), demanding multiple-choice questions (GPQA-D), instruction following (IFEval), and language-model likelihood (WikiText-2). IFEval

strict@50 uses 50 prompts per evaluation seed and requires satisfying every specified constraint for a prompt.

Table 25: **Complete auxiliary task profiles.** IFEval is mean ± sample SD over evaluation seeds. ‡ marks original non-generative point references for MuSR, GPQA-D, and perplexity; no evaluation-seed dispersion is assigned to those entries. Lower perplexity is preferred; dashes denote unavailable entries.

Configuration	MuSR	GPQA-D	IFEval strict@50	WikiText-2 PPL
Frozen base	—	—	—	—
Full-parameter GRPO	**63.00**‡	40.00‡	—	—
Standard gate	**63.00**‡	**41.00**‡	31.33 ±3.06	**7.0391**‡
Fixed RMS 2%	61.00‡	39.00‡	36.00 ±4.00	7.0890‡
Learned RMS, init 1%	61.00‡	40.00‡	34.00 ±7.21	7.0758‡
Full T-Router	61.00‡	38.00‡	**38.67** ±4.16	7.1185‡

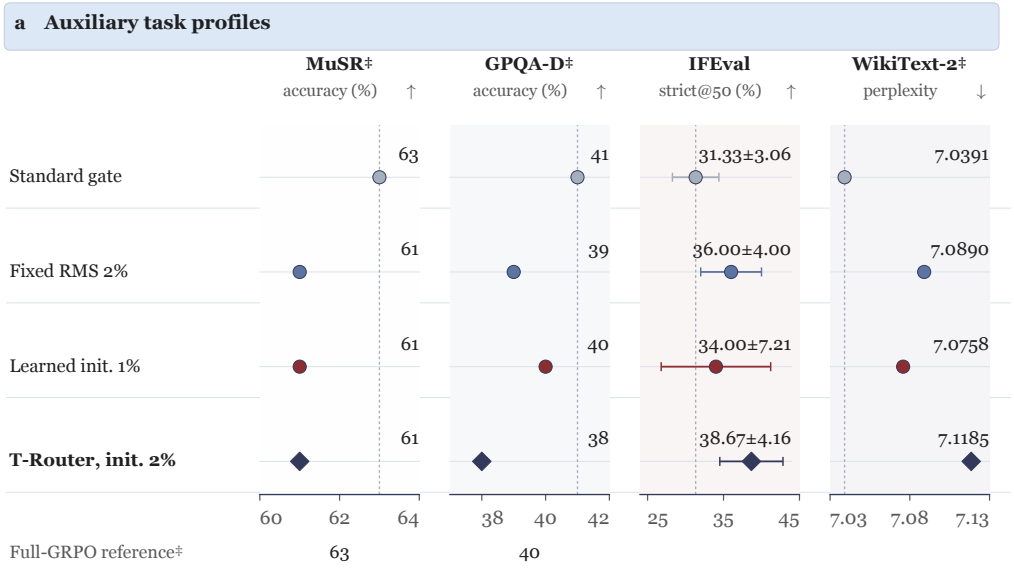


Figure 12: **Task-specific diagnostic scales.** Each task retains its native units. IFEval means carry evaluation-seed SDs; ‡ point references retain their original non-generative scoring status. The full-GRPO reference is available for the two multiple-choice tasks.

Full T-Router records the highest IFEval mean, 38.67 ± 4.16, compared with 36.00 ± 4.00 for fixed RMS 2%, 34.00 ± 7.21 for learned initialization at 1%, and 31.33 ± 3.06 for the standard gate. The three RMS settings share the MuSR point reference of 61.00, with GPQA-D references between 38.00 and 40.00. Their WikiText-2 perplexities are 7.0890, 7.0758, and 7.1185 for fixed 2%, learned 1%, and full T-Router, respectively. These values retain the separate meanings of accuracy, constraint satisfaction, and likelihood.

C.10 Sensitivity to the choice of family weights

MathAvg assigns equal weight to the three mathematical families. To characterize alternative priorities, reaggregate the same mean profiles with $w_{\mathrm{G}}, w_{\mathrm{M}}, w_{\mathrm{A}} \geq 0$ and $w_{\mathrm{G}} + w_{\mathrm{M}} + w_{\mathrm{A}} = 1$:

$$S_w(c) = w_{\mathrm{G}} s_{\mathrm{G}}(c) + w_{\mathrm{M}} s_{\mathrm{M}}(c) + w_{\mathrm{A}} s_{\mathrm{A}}(c). \tag{58}$$

Full T-Router has mean profile $(97.62, 92.73, 60.56)$, versus $(97.83, 84.07, 55.00)$ for learned RMS initialized at 1%. Therefore,

$$S_w(\text{init } 2\%) - S_w(\text{init } 1\%) = -0.21 w_{\mathrm{G}} + 8.66 w_{\mathrm{M}} + 5.56 w_{\mathrm{A}}. \tag{59}$$

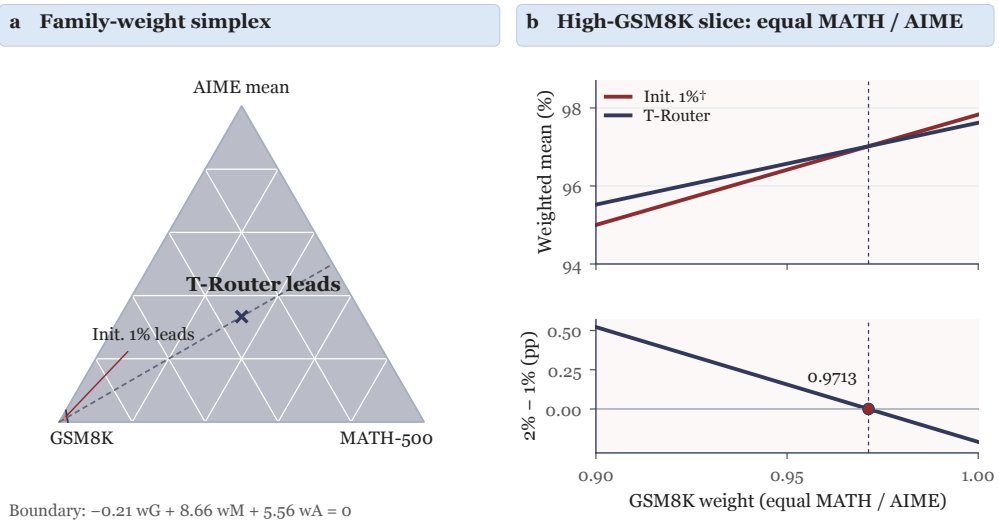


Figure 13: **Analytical sensitivity to benchmark-family weights.** The simplex shows the largest weighted mean among 14 complete profiles. The one-dimensional slice gives equal weight to MATH-500 and AIME and varies GSM8K weight; the crossover is near $w_{\mathrm{G}} = 0.9713$. Scores are analytical reaggregations of the supplied means using Equation 58.

These two profiles define the upper envelope: full T-Router strictly exceeds every other historical profile in all three family means. The 1% initialization has the largest GSM8K mean, while full T-Router has the largest MATH-500 and AIME means. Along $w_{\mathrm{M}} = w_{\mathrm{A}} = (1 - w_{\mathrm{G}})/2$, full T-Router leads up to $w_{\mathrm{G}} = 7.11/7.32 \approx 0.9713$. Along the equal-GSM8K/MATH-500 slice, its advantage is $4.225 + 1.335 w_{\mathrm{A}}$ and remains positive throughout. Equal family weighting lies in the full model's region. This geometry describes priorities over the same mean score profiles.

C.11 Question resolution and alternative aggregation

The mathematics evaluation contains 1,319 GSM8K questions, 500 MATH-500 questions, and 60 pooled AIME questions per seed. For that seed's correct-answer counts $n_{\mathrm{G}}, n_{\mathrm{M}}, n_{\mathrm{A}}$,

$$\mathrm{MathAvg} = \frac{100}{3}\left(\frac{n_{\mathrm{G}}}{1319} + \frac{n_{\mathrm{M}}}{500} + \frac{n_{\mathrm{A}}}{60}\right). \tag{60}$$

One extra correct answer in a single evaluation changes that seed's MathAvg by approximately 0.0253, 0.0667, or 0.5556 points for GSM8K, MATH-500, or pooled AIME. The three-seed mean spreads a change in one seed over three evaluations. Equal-family weighting prevents the largest test set from determining the aggregate solely through its question count.

An equal-question summary instead uses

$$S_{\mathrm{question}} = \frac{1319 s_{\mathrm{G}} + 500 s_{\mathrm{M}} + 60 s_{\mathrm{A}}}{1879}, \tag{61}$$

with weights approximately $(0.7020, 0.2661, 0.0319)$. One correct answer changes this per-seed summary by $100/1879 \approx 0.0532$ points. Reaggregating full T-Router's mean profile gives approximately 95.14, compared with its reported equal-family MathAvg of 83.64. Full T-Router has the highest mean in all three mathematical families among the nine main methods, so it remains first under every nonnegative family weighting summing to one. Both summaries express the same evaluated profile with different emphasis.

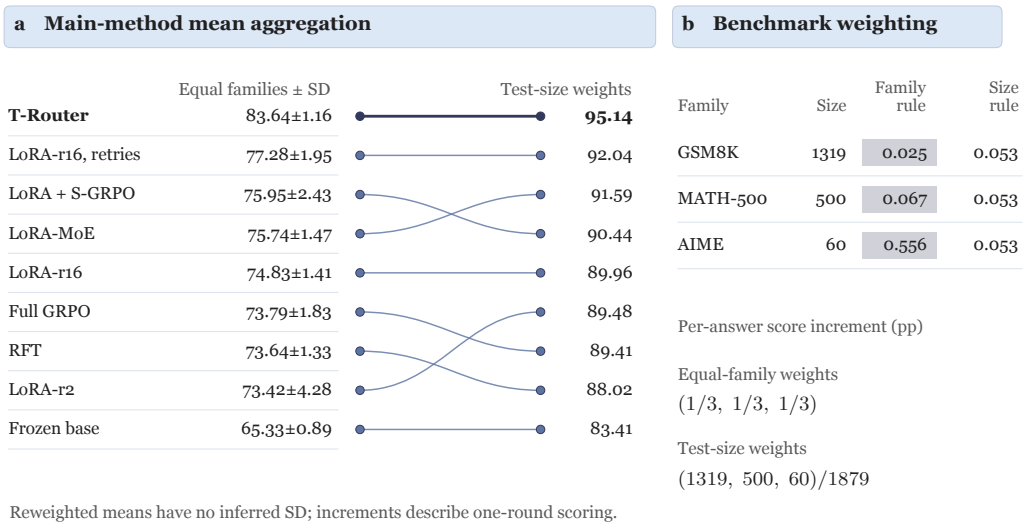


Figure 14: **Aggregation at full-test sizes.** The same nine method mean profiles are summarized by equal-family MathAvg and equal-question reaggregation. The size/resolution panel reports per-evaluation answer increments and family weights. The alternative weighted means are calculated from reported mean scores.

C.12 Evaluation-seed dispersion and complete variance record

The source evaluations use seeds 42, 43, and 44. For a reported score or paired difference x_r, the sample variance and SD are

$$\bar{x} = \frac{1}{3}\sum_{r=1}^{3} x_r, \qquad s^2 = \frac{1}{2}\sum_{r=1}^{3}(x_r - \bar{x})^2, \qquad s = \sqrt{s^2}. \tag{62}$$

Tables 26–28 preserve all seven supplied sample-variance panels at four decimals. Mathematical accuracy, MathAvg, and their differences use pp^2; search scores use the corresponding points^2. Aggregates and paired differences are formed within each evaluation seed before their variances are computed. Their supplied variances therefore retain the joint task/seed behavior; averaging component variances would not produce the same statistic. SDs describe evaluation variability for the reported configurations. Resource measurements and original ‡ references keep their separate measurement status.

Table 26: **Sample variances: mathematics and search.** All values are supplied s^2 over three evaluation seeds. Zero is an exact zero sample variance; Δ pairs each method with full-parameter GRPO at the same seed index.

(a) Mathematics

Method	GSM8K	MATH	AIME24	AIME25	AIME mean	MathAvg	Δ
Frozen base	0.8986	5.0800	33.3333	11.1111	11.1111	0.7874	0.8933
Full-parameter GRPO	0.0690	0.4933	48.1481	14.8148	28.7037	3.3394	0.0000
RFT	1.3412	6.0933	103.7037	181.4815	6.4815	1.7760	6.4687
LoRA-r2 + GRPO	0.0536	0.6533	300.0000	114.8148	162.0370	18.2979	7.1083
LoRA-r16 + GRPO	0.5882	6.4133	33.3333	48.1481	39.8148	2.0014	7.2478
LoRA-MoE + RO-GRPO	0.3679	2.0933	25.9259	25.9259	6.4815	2.1701	0.1681
LoRA + S-GRPO	0.2146	3.4533	381.4815	11.1111	67.5926	5.9142	5.2820
LoRA-r16, matched retries	0.2548	0.0133	59.2593	159.2593	36.1111	3.8180	14.1538
T-Router + GRPO	0.6227	0.4933	44.4444	3.7037	12.0370	1.3391	8.6068

(b) Agentic search

Method	BrowseComp	Δ F1	ASearch	Δ score
Frozen base	0.0925	1.5637	0.1699	0.3186
Full-parameter GRPO	2.3867	0.0000	0.2777	0.0000
RFT	0.5584	4.2884	0.8434	1.3926
LoRA-r2 + GRPO	0.3094	1.5502	0.3459	1.0129
LoRA-r16 + GRPO	0.3189	3.2539	0.0317	0.4089
LoRA-MoE + RO-GRPO	1.3643	6.5709	0.3075	0.4258
LoRA + S-GRPO	0.3256	2.4924	0.0371	0.1147
LoRA-r16, matched retries	1.2793	1.7436	0.5555	0.3915
T-Router + GRPO	1.3487	0.8305	0.5536	1.6154

Table 27: **Sample variances: components and writeback.** The component Δ is each variant minus full T-Router. Historical writeback settings retain their † estimate attribute.

(c) Component ablations

Configuration	GSM8K	MATH	AIME24	AIME25	MathAvg	Δ
Full T-Router	0.6227	0.4933	44.4444	3.7037	1.3391	0.0000
No cross-layer retrieval	0.3468	2.4400	14.8148	225.9259	5.5468	3.1758
Hidden-state memory	0.2836	1.7200	114.8148	25.9259	4.7817	8.7683
Learned static routing	0.4675	0.6533	33.3333	44.4444	2.9100	7.7297
No state (matched MLP)	0.5595	3.6933	311.1111	103.7037	16.7764	9.5347
No layer/block identity	0.1744	3.2533	114.8148	11.1111	1.1156	0.3102
No RMS alignment	0.4158	14.4533	11.1111	11.1111	0.1699	1.5405
No token-conditioned gate	0.1744	3.2933	70.3704	137.0370	6.7967	2.1560
No informative retries	0.2989	6.4933	3.7037	14.8148	1.4390	5.1010

(d) Writeback configurations

Configuration	GSM8K	MATH	AIME mean	MathAvg
Standard gate†	0.7645	10.8133	28.7037	7.9975
Fixed RMS 0.5%†	0.9791	2.5200	12.0370	2.8583
Fixed RMS 1%†	0.4541	9.6533	25.0000	5.4889
Fixed RMS 2%†	0.3123	4.4400	6.4815	1.3420
Learned RMS, init 0.5%†	0.0824	0.9733	69.4444	5.7185
Learned RMS, init 1%†	0.2088	2.7733	144.4444	19.3792
Full T-Router	0.6227	0.4933	12.0370	1.3391

Table 28: **Sample variances: historical profiles and auxiliary scores.** Changes in (e) are paired fixed-RMS-minus-original differences. Dashes in (g) preserve unavailable variances for original non-generative references or missing scores.

(e) Structure–writeback combinations

Structure	Original	Fixed RMS 2%	Change
Dense, standard gate†	7.9975	1.3420	5.1199
Wide512†	8.1360	0.4337	11.6171
Top2 composite†	4.5150	2.6144	3.8024
Local64†	6.1259	0.8179	2.6340

(f) Other historical configurations

Configuration	GSM8K	MATH	AIME24	AIME25	MathAvg
Standard gate†	0.7645	10.8133	0.0000	114.8148	7.9975
Standard gate, no KL†	0.3353	16.1733	33.3333	3.7037	2.9447
Local16†	1.2473	1.0000	11.1111	337.0370	7.5142
Local64†	0.3276	6.6133	25.9259	25.9259	6.1259
Wide512†	1.3852	3.7200	114.8148	25.9259	8.1360
Zero initialization†	0.1226	3.2933	114.8148	44.4444	5.2464
Orthogonal initialization†	0.0057	2.9733	25.9259	3.7037	0.1421
Top2 composite†	0.0824	1.1200	114.8148	92.5926	4.5150

(g) Auxiliary task profiles

Configuration	MuSR	GPQA-D	IFEval strict@50	WikiText-2 PPL
Frozen base	—	—	—	—
Full-parameter GRPO	—	—	—	—
Standard gate	—	—	9.3333	—
Fixed RMS 2%	—	—	16.0000	—
Learned RMS, init 1%	—	—	52.0000	—
Full T-Router	—	—	17.3333	—

D DESIGN INTERPRETATION AND FUNCTIONAL CORRESPONDENCE

T-Router pursues parameter-efficient reinforcement learning by concentrating adaptation in a communication interface around a frozen backbone. Completed block changes become explicitly accessible to later receivers, which learn their selection and relative influence. Separating transported content, recurrent depth state, and the current control signal gives each part of the trainable budget a concrete role. This is the computational form of the thalamic routing principle.

D.1 COMPUTATION AND COORDINATION AS SEPARATE ROLES

The relevant thalamic function is context-dependent regulation of cortical communication. The pulvinar coordinates information transmission between cortical areas through attention-dependent synchronization (Saalmann et al., 2012). Mediodorsal thalamic input amplifies functional prefrontal connectivity, sustaining rule representations without itself relaying their categorical content (Schmitt et al., 2017). Together, these findings motivate a routing interface that uses task context to regulate how distributed computations influence a receiver.

In T-Router, pretrained decoder weights provide the computational substrate, while the complete auxiliary module regulates communication. Compressors define accessible source content, the depth controller conditions routing, and calibrated writeback changes the inputs processed by the frozen layers. Green source storage, purple controller state, and gold residual writeback in Figure 2 belong to this one coordinating system. The cortical partitions in Figure 1 depict distributed backbone computation; the thalamic correspondence covers the full auxiliary pathway.

Table 29: **Functional correspondence and its computational realization.** Each row specifies an operation in the design.

Organizing principle	T-Router realization	Computational consequence
Distributed processing	Frozen decoder blocks	Every block produces its original nonlinear transformation on the received residual
Selective inter-area communication	Attention over completed block records	Later computations select from origin-indexed intermediate changes
Context-dependent coordination	Token-specific slots recurrent over depth	Retrieval uses an auxiliary history of preceding computational states
Modulation of influence	Signed RMS-calibrated writeback	Direction and relative magnitude are controlled separately
Coordinating system around existing processing	Full source–controller–writeback pathway	Adaptation capacity is concentrated in communication between frozen blocks

This mapping determines concrete design choices. A receiving block retains its full transformation instead of becoming an expert that may be skipped. Each source has an explicit depth origin rather than becoming an anonymous part of an accumulated hidden state. The same coordinate system is used throughout the paper: block index identifies an available computation, slot index identifies a component of the routing context, and token index identifies the causal position at which the decision occurs. The routing correspondence is operational: source weights determine which completed changes contribute to the mixture, and the gate controls their signed influence on the receiving block.

D.2 WHY THE SOURCE BANK AND CONTROLLER CARRY DIFFERENT INFORMATION

The source bank preserves completed displacements, each compressed using a source-specific map. Its size grows with completed depth blocks, and an appended record remains fixed for subsequent reads in that forward call. The controller state S rewrites a fixed number of slots at each layer, accumulating update proposals through decay and slot allocation. The context P is a query-dependent readout of the updated slots, recomputed for each receiving layer.

These objects have complementary roles. The bank supplies content associated with earlier blocks; S summarizes the trajectory of residual states, source summaries, and layer identities. The current residual queries this state to obtain P. Routing and gating then combine P with the current residual

to select source content and regulate its influence. The transported vector c comes from the source bank. This separation lets accessible sources grow without increasing the number of controller slots.

Table 30: **Source storage and controller state are complementary interfaces.** Their different update rules organize the computation even when their widths coincide.

Property	Source bank	Depth controller
Index meaning	Completed block identity	Slot identity
Cardinality	Grows as blocks finish	Fixed at $K = 8$
Content width	$r = 256$	$p = 256$ per slot
Write frequency	Once per completed block	Once per decoder layer
Write rule	Append compressed block change	Decay plus allocated shared proposal
Read function	Source attention produces c_ℓ	Slot attention produces P_ℓ
Direct contribution	Content projected into the residual	Context for source selection and gate input
Lifetime	One forward call	One forward call

Equal dimensions do not make the two representations interchangeable. A bank of seven records at layer 31 retains seven block identities. Eight controller slots at that layer can each contain a different mixture of many earlier proposals. The controller's rank-one increment per layer is compatible with a higher-rank accumulated state, as shown in Appendix B.4. The two representations also have different sensitivities to intervention: changing a source record alters both retrieved content and later controller input, whereas changing only a controller state changes how existing source content is selected and scaled.

D.3 Three coordinates of a residual intervention

The raw direction can be written as $w_\ell = U_\ell c_\ell$. This expression contains source selection through the attention-weighted source mixture and transport through the receiving projection. On the active branch, calibration maps the raw direction onto a sphere whose radius is the receiving residual norm. Finally, the signed gate chooses the intervention's relative magnitude and orientation along that calibrated direction.

The three coordinates have distinct mathematical roles. Source attention is a distribution over candidate origins; it does not specify the norm of their projected sum. The projection supplies a receiver-specific low-dimensional subspace; a scalar gate does not change that subspace. Calibration changes the norm of a nonzero direction while preserving its ray, and the gate then selects a point along the line spanned by that direction. On the active branch, $|g_\ell|$ has the direct interpretation $\|R_\ell\|/\|h_\ell\|$.

Table 31: **Coordinates of controlled reuse.** Distinct parameter groups control the origin, direction, and relative strength of the intervention.

Coordinate	Governing objects	Quantity it controls
Source preference	Query, source keys, depth context	Relative attention across completed computations
Content transport	C_b, source value projection, U_ℓ	Source-to-receiver transformation within $\mathrm{col}(U_\ell)$
Relative influence	Target-RMS calibration and signed gate	Norm relative to the receiving residual and sign along the direction

This factorization explains why writeback calibration and routing structure interact. Sparsifying source selection changes the mixture delivered to the receiving projection. Normalizing stored deltas changes the geometry on which keys and values are built. Changing controller width changes the context available to the query and gate. A common fixed writeback coefficient can therefore act on different learned directions in different designs. The paired results in Appendix C.8 reveal this interaction at the configuration level. The matched component study in Table 20 separately tests the operations of the full interface.

The learned gate adds another degree of freedom to calibrated transport. Its input contains both the current residual and controller context, so the model can modulate an already selected direction differently across tokens and receiving layers. A fixed gate retains token-dependent directions through source routing, but uses a common scalar coefficient. Thus "fixed" and "learned" refer to the strength coordinate; they do not divide the models into static and dynamic source-selection systems.

D.4 Relation to alternative adaptation interfaces

A local low-rank weight update changes a layer matrix by $\Delta W = AB$ and acts on the current input to that matrix (Hu et al., 2022). A bottleneck adapter applies a learned local transformation around a layer (Houlsby et al., 2019). T-Router instead retains explicit earlier sources and chooses a combination of them at a receiving layer. Its transport matrices are also low rank, but their operands are completed block changes selected through an input-dependent depth route.

DenseNet exposes earlier layer features through fixed feature concatenation (Huang et al., 2017). T-Router reads compressed block changes through learned attention, projects the mixture into the receiving residual space, and calibrates its influence. This gives a fixed-width frozen backbone explicit access to historical computation, with reuse conditioned on the current residual and depth context.

The distinction can be expressed at a fixed receiver. With fixed source weights and no recurrent dependence, the change-dependent part of the raw writeback is a weighted sum of $T_{\ell b} D_b$ terms, with $\mathrm{rank}(T_{\ell b}) \leq r$. With the full module, the weights, records, controller context, calibration factor, and gate all vary with the input. A low-dimensional intervention space therefore coexists with a nonlinear input-dependent adaptation function. This is different from treating low rank as a statement that the entire adapted network is a fixed linear perturbation.

Delta Attention Residuals uses sublayer or block changes as routed content (Luo et al., 2026a); mHC-based finetuning learns read/write routing over multiple residual streams around frozen branches (Oldenburg et al., 2026). T-Router's coupled interface assigns separate roles to compressed origin-indexed records, recurrent context, and relative-scale writeback. The bank preserves completed changes, while controller slots accumulate depth history. A current-state query reads those slots to obtain P, which jointly conditions source attention and the signed gate. This organization makes content reuse and its control trainable within one compact RL interface.

Reuse operates during the causal forward pass: a receiver draws on completed records to shape the input to its frozen transformation. During RL, the task objective assigns credit through the resulting computation, differentiating the source, controller, routing, and writeback parameters. Frozen layers retain their input derivatives, connecting these communication choices to the output loss. Training therefore learns a reuse rule that executes directly during generation.

D.5 Connecting the empirical comparisons to the interface

The component study connects the organization of adaptation capacity to reasoning quality. Block-change memory reaches 83.64 MathAvg versus 67.46 for hidden-state memory at the same allocation; removing retrieval gives 69.48. The full controller also exceeds a capacity-matched MLP, which has 1,646 more parameters and scores 72.80. These comparisons favor accessible changes and recurrent control within the same compact-budget regime. S accumulates depth history, while the current hidden state queries S to obtain P as the context for reuse.

Learned static routing gives 64.98, and removing the token-conditioned gate gives 63.78. The two variants address distinct receiving-layer decisions: which earlier changes contribute and how strongly their mixture enters the residual. All backbone layers continue to execute in order. The supplementary width, initialization, local-branch, and sparse-composite studies explore coupled design choices around this interface; their interaction patterns complement the component comparisons.

The resulting optimization target is a rule for composing existing intermediate contributions. The bank defines available content, depth context conditions its selection, and writeback regulates its influence on the next frozen transformation. Training these choices together yields 83.64 MathAvg with a 41.73M-parameter allocation, compared with 73.79 for full-parameter GRPO and 77.28 for matched-retry LoRA. Computation reuse thus connects the efficiency objective to both an explicit mechanism and its measured reasoning performance.