GLOBAL CACHE WORKSPACE: ADAPTING FROZEN LANGUAGE MODELS THROUGH THEIR MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Language models are usually adapted by updating their weights, trading broad, out-of-domain competence from large pre-training runs for in-domain gains. We show that adaptation can instead live in a model's memory. We introduce the Global Cache Workspace (GCW), a module inspired by Global Workspace Theory that selects information from specialised heads across the layers of a frozen backbone, integrates it in a shared workspace, and broadcasts an update back into their memory, covering both attention and linear-attention layers. Trained with next-token prediction while every backbone parameter stays fixed, GCW reaches in-domain gains competitive with full fine-tuning and LoRA while forgetting far less: across every adaptation setting we study, it loses the least out-of-domain reasoning accuracy on average, and after mathematical adaptation its loss is less than half that of weight-space methods. Its adaptation is also selective. With thinking enabled, GCW outperforms full fine-tuning in domain while shortening in-domain reasoning by up to 82%, a saving comparable to weight-space methods, yet keeps reasoning on unfamiliar tasks close to its original length; weight-space methods shorten that reasoning too, and lose far more accuracy there. Adapting what a frozen model remembers, rather than its weights, thus adds capability while preserving far more of what the model already knows.

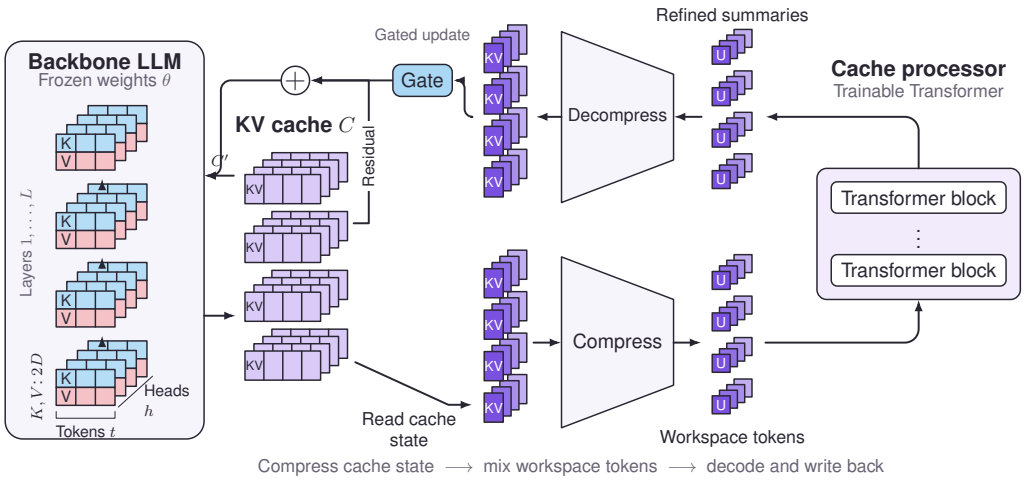


Figure 1: **Global Cache Workspace.** A trainable cache processor revises the memory of a frozen language model. Cache states are compressed into workspace tokens, mixed by a shared Transformer, and decompressed into gated residual updates. The revised cache C' is used when generation resumes. BGT retains positions as workspace tokens; GCW retains heads. Figures 3 and 4 detail their compression and attention mechanisms. Selection and write locations are omitted here. No future tokens are read.

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.

1 INTRODUCTION

Transformers (Vaswani et al., 2017) and the foundation models built on them are usually adapted to new domains by changing their weights. We ask whether adaptation can instead live in the model's memory: whether a small module that reads and rewrites the memory of a frozen model can acquire a new skill while preserving more of what the model could already do.

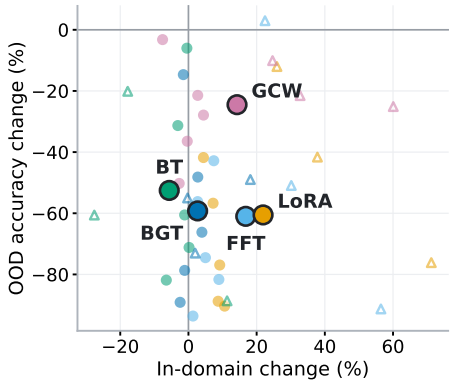


Figure 2: **Adapting memory rather than weights.** Relative change in in-domain and out-of-domain (KOR-Bench) accuracy from the backbone after mathematical adaptation. Large markers: method means; small markers: individual backbones (filled: thinking off; open: thinking on). BT and BGT are position-based cache processors (Section 3); values in Tables 3 and 4.

Problem. Full fine-tuning and low-rank adaptation (LoRA) (Hu et al., 2022) change the weights to raise in-domain accuracy, which can degrade performance outside the adaptation domain, much as sequential learning can interfere with previously learned tasks (McCloskey & Cohen, 1989; Kirkpatrick et al., 2017). In the programming and mathematics settings studied by Biderman et al. (2024), LoRA forgets less than full fine-tuning but also learns less, and in both cases the same learned weight update is applied to every input. Prefix and prompt tuning leave the weights intact but learn prefixes or prompt embeddings that are likewise fixed after training (Li & Liang, 2021; Lester et al., 2021). Chain-of-thought prompting elicits reasoning without updating the weights (Wei et al., 2022), but in standard autoregressive decoding each additional reasoning token requires a further decoding step and cache entry. We therefore pursue a module that attaches to a frozen backbone, adds no visible tokens, and computes its update from the current context, with the aim of adding capability without giving up out-of-domain competence.

Gap. Such a module needs a substrate to compute over, and the model's inference-time memory is the natural candidate. We call this memory the cache: the KV entries of attention heads and the recurrent states of linear-attention heads, which store the model's representations of the tokens processed so far. The cache is fragmented: each layer keeps its own cache, each KV head stores its own slice, and attention reads it separately for each query head. This matters because heads can specialise in distinct computations, such as syntactic tracking and induction-style copying (Voita et al., 2019; Olsson et al., 2022); attention often concentrates on a small subset of cached positions (Zhang et al., 2023); and analyses of head interactions find functionally important synergy alongside redundancy (Urbina-Rodriguez et al., 2026). What one head has resolved is therefore not directly available to another. In a standard pretrained Transformer, heads exchange information through the residual stream within a position and through attention across positions (Elhage et al., 2021), but both paths run from earlier to later layers: a layer reads only cache entries produced at its own depth, so what deeper layers compute never reaches shallower ones, at the current position or any later one. The residual stream also has a fixed width, and superposition offers a mechanism by which features share its dimensions (Elhage et al., 2022), so all components communicate through one limited channel. WhiteMatter routes deep representations to shallow layers through shared KV channels (Zhang & Ren, 2026), but it changes how entries are produced and is trained from scratch. Work on the cache of pretrained models includes eviction (Zhang et al., 2023; Wang et al., 2026) and localised edits such as suppressing a context span, appending corrections, or one-shot steering (Li et al., 2026; Li, 2026; Belitsky et al., 2025). Learned cache processors either append new entries (Liu et al., 2025) or consolidate each layer separately (Oomerjee et al., 2026), so heads and layers still cannot exchange what they hold. We ask whether a small module can let them do so, writing the result into memory that every layer reads at subsequent positions.

Motivation. Global Workspace Theory (GWT), developed in cognitive science by Baars (1997) and subsequently given a neural formulation reviewed by Mashour et al. (2020), describes conscious access in terms of information selected from specialised, parallel systems, integrated in a capacity-

limited workspace, and broadcast to multiple downstream systems. Machine learning has since adopted this structure as an architectural prior for modularity and communication (Bengio, 2017; Goyal et al., 2022). It maps onto the gap above: heads are the specialised systems, competition determines which of them contribute to a small shared processor that acts as the workspace, and every head reads the result. We use this architectural analogy to motivate selection, shared processing, and broadcast, without treating the resulting mechanism as a model of cognition.

Proposal. We introduce the Global Cache Workspace (GCW), a GWT-inspired cache processor that periodically selects a sparse set of source heads from the cache of a frozen decoder, integrates them in a shared workspace, and writes a gated update into each head's latest cache entry (Figure 1). Writing the update into the cache is what makes the exchange global: every layer that later reads that memory sees it, so heads separated by depth can communicate without retraining the backbone or routing information through the residual stream at a single position. Because workspace tokens are indexed by head rather than by position, one processor reads the KV entries of attention heads and the recurrent states of Gated DeltaNet heads (Yang et al., 2025), and can write to both, as in the hybrid Qwen3.5 backbones we use (Qwen Team, 2026). We compare GCW with full fine-tuning, LoRA, and position-based cache processors across mathematical, broad, and sequential adaptation of Llama 3.2 and Qwen3.5 backbones.

Contributions.

- **Architecture.** A brain-inspired architecture that adapts a frozen language model through its memory rather than its weights: a head-indexed global workspace over both softmax and linear attention caches (Section 3).
- **Adaptation with retention.** Evidence that adapting memory preserves out-of-domain competence: GCW reaches in-domain gains competitive with weight-space methods, with shorter in-domain reasoning, while forgetting substantially less (Section 4.1).
- **Validation of the workspace design.** A sparse workspace using about 10% of heads as sources recovers near-full accuracy, echoing proposals for sparse biological workspaces. Increasing processor size brings no consistent improvement, suggesting that our core mechanism is the key contributor to performance, rather than additional capacity (Section 4.2).

2 Background and Related Work

Where adaptation lives. Sequential learning can cause catastrophic interference with previously learned tasks (McCloskey & Cohen, 1989; Kirkpatrick et al., 2017). In language-model adaptation, LoRA learns low-rank weight updates while freezing the original parameters (Hu et al., 2022). In the programming and mathematics settings studied by Biderman et al. (2024), standard low-rank configurations learn less than full fine-tuning but better preserve performance outside the target domain. Both approaches deploy the same learned weight update across inputs. Prefix and prompt tuning leave the weights untouched and learn cache entries or input embeddings instead (Li & Liang, 2021; Lester et al., 2021), and Cartridges trains such a cache offline for a given corpus, finding next-token prediction alone insufficient without self-study and distillation (Eyuboglu et al., 2026). These learned prefixes or caches are fixed after training. GCW also freezes the backbone and acts only on its memory, but computes its update from the current context.

Latent reasoning. Continuous thoughts and pause tokens move computation away from visible answers (Hao et al., 2024; Goyal et al., 2024), recurrent-depth models loop a weight-tied block and extrapolate depth at inference (Geiping et al., 2025), and test-time training writes the context into the weights (Tandon et al., 2025). Each acquires the extra computation by training the model's own weights, whereas we leave them fixed and act on the model's memory.

Cache processing. Cache-eviction methods reduce storage by retaining recent and high-attention entries (Zhang et al., 2023) or by randomly evicting reasoning-trace entries (Wang et al., 2026). Other work edits the cache locally: suppressing a context span (Li et al., 2026), appending errata that override stale conclusions (Li, 2026), or adding steering vectors (Belitsky et al., 2025). Liu et al. (2025) keep the decoder frozen and train a coprocessor that appends latent embeddings to its cache,

lowering perplexity on subsequent tokens and improving reasoning without task-specific training. Like prefix tuning, this adds entries rather than revising those the model has already written, so it does not isolate the question we study: whether communication across heads and layers, written back into existing memory, supports adaptation.

Bottlenecked Transformers. Closest to our work, BT (Oomerjee et al., 2026) rewrites recent and attention-selected past KV entries at newline-delimited reasoning boundaries. A layer-local, non-causal Transformer processor mixes the selected positions within each layer and adds a gated residual to them (Section 3), without shortening the cache; unselected entries remain unchanged. The processor trains through next-step prediction with a frozen, fine-tuned backbone, without an explicit information-bottleneck loss. GCW keeps this gated update of existing cache entries but replaces the per-layer processors with one workspace shared across layers and heads. BT is therefore our main architectural baseline, and BGT, which combines layers before BT-style processing of positions, separates the effect of sharing across layers from the choice of heads as specialists.

Global Workspace. Global Workspace Theory (GWT) describes conscious access through competition among parallel, specialised processes for a capacity-limited workspace whose contents are widely broadcast (Baars, 1997; Mashour et al., 2020). We translate these principles into a shared state distinct from local processing, competitive contributions, and access to the resulting broadcast. In a standard Transformer, heads and MLPs add outputs to a shared residual stream without an explicit competition for write access; within a forward pass, later layers read earlier outputs (Elhage et al., 2021). Its fixed width constrains communication, and features can share representational dimensions through superposition (Elhage et al., 2022). Earlier GWT-inspired systems include game-avatar control using spiking neurons (Fountas et al., 2011) and Transformer-based multimodal fusion (Bao et al., 2020). The consciousness prior proposes a low-dimensional bottleneck that selects and broadcasts task-relevant content (Bengio, 2017). The Shared Workspace of Goyal et al. (2022) implements competitive writes to shared slots and broadcasts the result back to specialists, represented by sequence positions in its Transformer variant. The Perceiver uses a latent array to query inputs (Jaegle et al., 2021), an architecture argued to satisfy several functional workspace criteria (Juliani et al., 2022). Other approaches use translation between module-specific latent spaces (VanRullen & Kanai, 2021) or learn routing through a gated workspace on controlled modulo-10 addition tasks (Chateau-Laurent & VanRullen, 2025). In pretrained language models, Gurnee et al. (2026) identify representations poised for verbalisation that carry coherent content in intermediate layers and are broadcast more widely than other representations, suggesting an implicit workspace. We add an explicit workspace to a pretrained backbone, with its heads as specialists.

Cross-Layer Communication. Cross-layer connections combine representations through depth-weighted averaging (Pagliardini et al., 2024), multiple residual streams (Zhu et al., 2025), or input-dependent mixing (Heddes et al., 2025; Xiao et al., 2025), including attention over a frozen backbone's layer representations (ElNokrashy et al., 2024). Cross-layer cache methods let layers read, mix, or reconstruct other layers' key-value states, often shrinking the cache while preserving or improving quality (Zhang & Ren, 2026; Lin et al., 2026; Filippova et al., 2026). The former aggregate representations within the forward pass and the latter change how the backbone produces its cache; neither writes later-layer information back into earlier layers' existing cache entries, which our workspace does while keeping the backbone fixed.

3 METHODS

Cache rewriting. We freeze the decoder and train a processor to revise its memory without adding tokens. Following BT, θ and ω denote the real-valued backbone and processor parameter vectors; only ω is trained. The processor runs between causal windows of P tokens, in training, prefill, and decoding alike, so it never reads a token the backbone has not processed. At each invocation, it reads the P most recent cached positions and the k older positions of largest salience, $m = P + k$ in total (Appendix B.1). All variants encode memory into a matrix of workspace tokens $\mathbf{U}$, mix these tokens with non-causal attention, and decode gated updates back into memory. They differ in which memory dimension becomes the token axis of $\mathbf{U}$, and in whether heads compete for access (Table 2). Figures 3 and 4 expand the compression and cache-processor blocks of Figure 1, isolating this choice of axis and the resulting attention pattern.

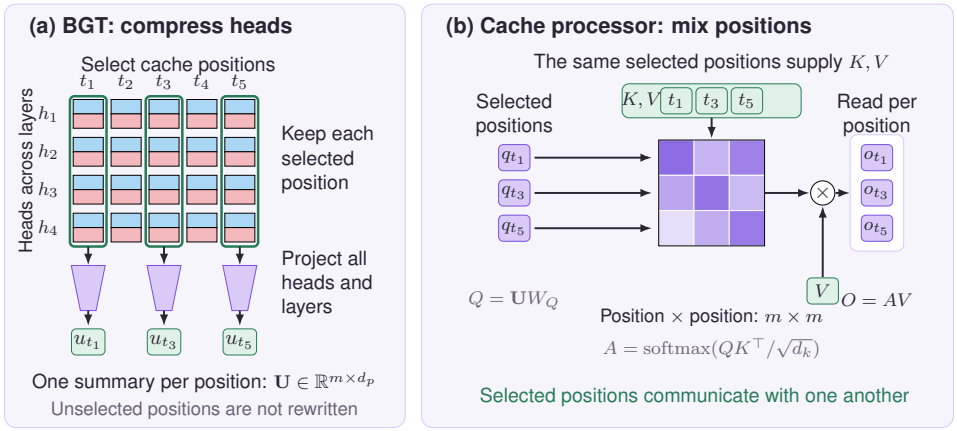


Figure 3: **BGT: positions as communicating components.** **(a)** After position selection, KV features from all heads and layers at each retained position are projected into one workspace token; positions remain separate. Green outlines mark selected positions. **(b)** These m tokens supply queries, keys, and values, giving an $m \times m$ attention matrix across positions. The output at each position is subsequently decoded across heads and layers at that position. Selection thus acts on temporal slices of the cache, consistent with the position-as-specialist interpretation of Goyal et al. (2022). Colours in the attention matrix illustrate query-dependent weights.

BT and BGT. BT and BGT retain selected sequence positions, shared by all layers and heads, as workspace tokens, compressing KV features rather than pooling positions. BT does this independently within each layer; BGT combines layers before encoding (Figure 3a). Let L, m, D, d_p be positive integers denoting layers, selected positions, KV features per layer, and processor width. For layer $\ell \in \{1, \dots, L\}$, each row of $\mathbf{X}^{(\ell)} \in \mathbb{R}^{m \times D}$ concatenates its KV heads at one selected position. BGT uses shared projections $W_{\text{in}} \in \mathbb{R}^{LD \times d_p}$ and $W_{\text{out}} \in \mathbb{R}^{d_p \times LD}$, a shape-preserving Transformer processor $\mathcal{T}_\omega^{\text{proc}}$, and a gate logit $g \in \mathbb{R}$. With logistic function σ, its position-indexed cache $\mathbf{X} \in \mathbb{R}^{m \times LD}$ and workspace $\mathbf{U} \in \mathbb{R}^{m \times d_p}$ give the rewrite

$$\begin{aligned} \mathbf{X} &= [\mathbf{X}^{(1)} \mid \cdots \mid \mathbf{X}^{(L)}], \qquad \mathbf{U} = \mathbf{X}W_{\text{in}}, \\ \mathbf{X}' &= \mathbf{X} + \sigma(g)\, \mathcal{T}_\omega^{\text{proc}}(\mathbf{U})W_{\text{out}}. \end{aligned} \tag{1}$$

The input projection combines layers, attention mixes positions (Figure 3b), and the output projection distributes updates across layers and heads. Keeping positions separate permits position-specific rewrites and follows the position-as-specialist interpretation of Goyal et al. (2022). Every selected position is rewritten, so older entries can be revised; unselected positions remain unchanged. Biases and normalisation are omitted from the equation.

GCW. GCW instead treats heads in sequence-mixing layers as the specialised modules of the global workspace. The functionally important, synergistic attention-head interactions reported by Urbina-Rodriguez et al. (2026) motivate communication at this level; our selector does not estimate synergy. Where BT/BGT compress features while retaining positions, GCW compresses histories while retaining heads (Figure 4a).

Each KV head c selects its own m positions by salience and forms a history $\mathbf{X}_c \in \mathbb{R}^{m \times 2d_h}$ of concatenated keys and values, with rotary embeddings removed so that its summary depends on content rather than absolute position. An attention-pooling encoder $\mathcal{E}_\omega$, queried from the latest position, compresses each history into one workspace token, giving $\mathbf{U} \in \mathbb{R}^{N \times d_p}$ together with the linear-attention tokens described below (Appendix B.2). Heads then compete for access. With head scores $\mathbf{s} \in \mathbb{R}^N$, given by the salience of each retained history, and source budget $b \in \{1, \dots, N\}$, the selected indices $\mathcal{J} \subseteq \{1, \dots, N\}$ and processor outputs $\mathbf{O} \in \mathbb{R}^{N \times d_p}$ satisfy

$$\begin{aligned} \mathbf{U} &= \mathcal{E}_\omega(\mathbf{X}_1, \dots, \mathbf{X}_N), \\ \mathcal{J} &= \mathrm{TopK}(\mathbf{s}; b), \\ \mathbf{O} &= \mathcal{T}_\omega^{\text{proc}}\big(\underbrace{\mathbf{U}}_{\text{all heads}}, \underbrace{\mathbf{U}_{\mathcal{J}}}_{\text{selected sources}} \big). \end{aligned} \tag{2}$$

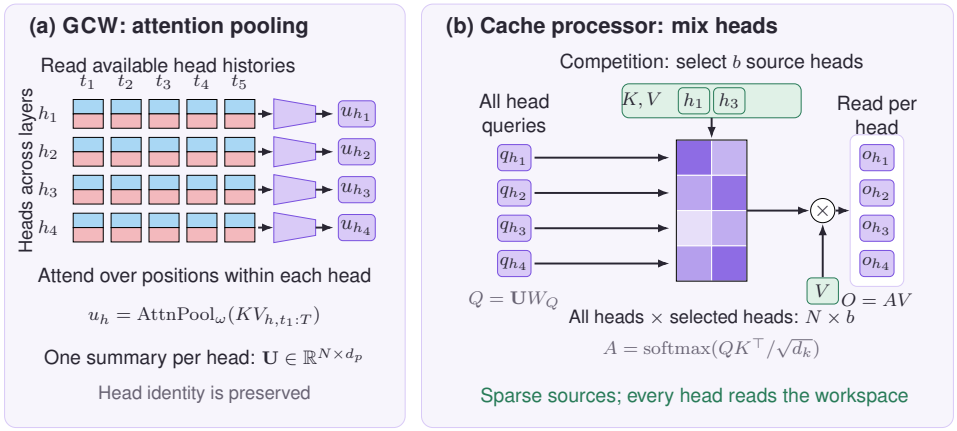


Figure 4: **GCW: heads as communicating components.** **(a)** Attention pooling over time compresses each head's available KV history into one workspace token, retaining head identity across layers. **(b)** Competition selects b source heads (green) to supply keys and values, while all N heads supply queries. The $N \times b$ attention matrix lets each head read a different weighted combination of the same shared sources. This implements sparse contribution to a workspace with broadcast access for every head; decoded updates reach each head's latest cache position. Colours illustrate query-dependent weights. Only compression and selective attention are shown; the remaining architecture follows Figure 1.

All heads supply queries; the b selected rows supply keys and values (Figure 4b). Selection is a hard top-k, so gradients train the selected tokens rather than the choice (Appendix B.3). An output projection $W_{\text{out}} \in \mathbb{R}^{d_p \times 2d_h}$, shared by all KV heads, decodes each head's output $\mathbf{o}_c \in \mathbb{R}^{d_p}$ separately into a gated residual update of that head's latest cache entry $\mathbf{x}_{c,m} \in \mathbb{R}^{2d_h}$:

$$\mathbf{x}'_{c,m} = \mathbf{x}_{c,m} + \sigma(g)\, \mathbf{o}_c W_{\text{out}}. \tag{3}$$

Every head, selected or not, receives this update, with rotary embeddings reapplied to its key. Older entries are read-only: GCW broadcasts to the cache frontier rather than revising history (Appendix B.4).

Workspace interpretation. BGT, like the Shared Workspace of Goyal et al. (2022), treats positions as specialists. In the account of Baars (1997), unconscious specialised processors operate in parallel, their outputs compete for a limited-capacity workspace, and its contents are broadcast globally; GCW assigns the specialist role to heads, whose selected members contribute shared sources and all of which read the result with their own queries.

Hybrid memory. Linear attention stores a state matrix $\mathbf{S}$ rather than one KV entry per past token, so it has no position-indexed $\mathbf{X}$. Indexing workspace tokens by head removes the need for a common sequence axis. For each linear-attention head, an adapter forms a unit-norm query $\mathbf{q}$ from the workspace tokens of the next attention layer, reads $\mathbf{r} = \mathbf{S}^\top \mathbf{q}$, and adds $[\mathbf{q}; \mathbf{r}]$ to $\mathbf{U}$ as a workspace token, scored for selection by $\|\mathbf{r}\|_2^2$. Its output is decoded into a unit-norm key direction $\hat{\mathbf{k}}$ and a value $\Delta\mathbf{v}$, written back as the gated rank-one update $\mathbf{S}' = \mathbf{S} + \sigma(g)\, \hat{\mathbf{k}}\Delta\mathbf{v}^\top$ with the gate of Equation 3 (Appendix B.5). BT and BGT rewrite only KV entries.

Training. All variants minimise next-token cross-entropy on supervised answer tokens. Let $\mathcal{V}$ be the vocabulary, $x_t \in \mathcal{V}$ a token, $x_{<t}$ its preceding prefix, and $\mathcal{S} \subset \mathbb{N}$ the finite set of supervised positions. With $p_{\theta,\omega}$ denoting the decoder's token distribution under cache interventions, the loss is

$$\mathcal{L}(\omega) = -\sum_{t \in \mathcal{S}} \log p_{\theta,\omega}(x_t \mid x_{<t}). \tag{4}$$

Gradients pass through the frozen decoder into the processor. There is no cache-reconstruction target.

3.1 Experimental Setup

Adaptation and evaluation. We adapt pretrained instruct backbones from the Llama 3.2 (Meta, 2024) and Qwen3.5 (Qwen Team, 2026) families in three settings: mathematical adaptation on OpenMathInstruct-2 (Toshniwal et al., 2025), broad adaptation on 128,000 Reasoning Gym (Stojanovski et al., 2025) examples, and sequential adaptation, which continues each method's mathematics-adapted checkpoint on Reasoning Gym. Each stage uses 2,000 optimisation steps. We evaluate zero-shot on mathematical benchmarks that match the training format (GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), and GSM-Hard (Gao et al., 2023)), on benchmarks beyond it (TheoremQA (Chen et al., 2023), Gaokao MathQA (Zhong et al., 2023), and LogiQA (Liu et al., 2020)), and on KOR-Bench (Ma et al., 2024), which supplies unseen rule systems in context and measures out-of-domain retention throughout (Appendix A.1).

Baselines and controls. We compare against full fine-tuning (FFT) and LoRA. All cache processors add 1.5% of the backbone's non-embedding parameters, counting every added parameter, and LoRA's rank matches this budget up to a cap of 32 (Appendix A.1). Within each setting, data, training order, prompts, and decoding are matched. Backbones with a thinking mode are evaluated with thinking disabled and, after mathematical adaptation, also enabled.

4 Results

Throughout, we measure change relative to the matched backbone, $100(x/x_0 - 1)$; per-backbone accuracies, token counts, and truncation rates are in Appendix C.

4.1 Adapting memory rather than weights

Retention. Adapting memory instead of weights preserves out-of-domain competence at little in-domain cost. In Figure 2, full fine-tuning and LoRA gain the most in domain but sit at the bottom of the plot: on average they keep less than half of their KOR-Bench accuracy. GCW lies close to them horizontally and far above them vertically, keeping about three-quarters (24.5% mean loss, against 61.0% and 60.5%) and, per question, more of the backbone's correct answers than both in seven of eight observations (Table 4, Appendix D). This separation is not a side effect of weaker in-domain learning: on the two larger Qwen3.5 backbones with thinking disabled, where GCW matches one weight-space baseline in domain, it scores 10 to 21 points higher on KOR-Bench (Tables 3 and 4). The same ordering holds after broad and sequential adaptation, where GCW again has the smallest mean KOR-Bench loss and shortens responses relative to the backbone (Figure 5).

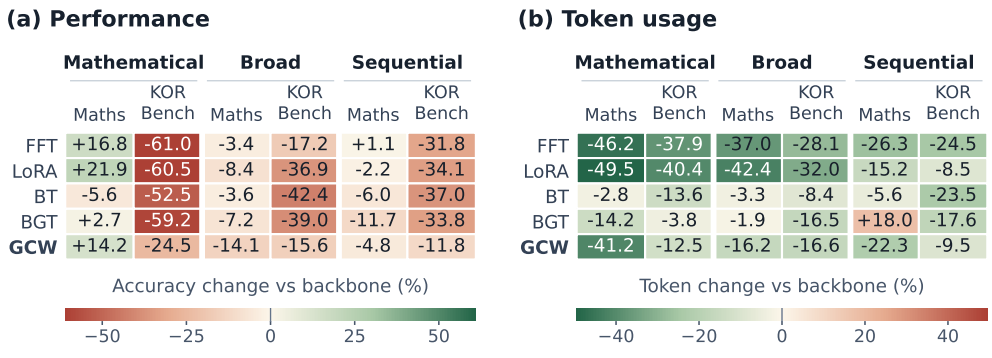


Figure 5: **Accuracy and token changes across adaptation settings.** Mean $100(x/x_0-1)$ relative to the matched backbone, with equal weight per backbone and thinking-mode observation (mathematical adaptation: five backbones with thinking disabled and three Qwen3.5 backbones with thinking enabled; other settings: four backbones with thinking disabled). Maths accuracy averages six tasks after mathematical adaptation and five, excluding TheoremQA, otherwise; token changes pool the generation tasks. Green denotes higher accuracy or fewer tokens; panels use independent colour scales.

Selective reasoning. With thinking enabled, GCW learns to reason concisely in its training domain and leaves reasoning elsewhere largely intact. In domain, full fine-tuning, LoRA, and GCW all shorten reasoning sharply, mostly by resolving truncation: the backbones exhaust the 4,096-token budget on 39% to 84% of in-domain responses. GCW cuts in-domain tokens by 67% to 82% and exceeds full fine-tuning's in-domain accuracy on every backbone, approaching LoRA's. Out of domain, the methods diverge. The weight-space methods carry the shortening over to KOR-Bench and lose far more accuracy there, whereas GCW shortens KOR-Bench responses by at most 19% and keeps truncation close to the backbone's (Tables 3 and 4). A weight update changes behaviour on every input; GCW computes its update from the current context, and its effect is correspondingly concentrated in domain.

Heads as specialists. Only the head-indexed workspace escapes the trade-off. BT and BGT share GCW's frozen backbone and gated cache updates but treat positions, not heads, as the communicating units. After mathematical adaptation, both stay near the backbone in domain (Figure 2), and in every setting they lose KOR-Bench accuracy on the scale of the weight-space methods (Figure 5). With thinking enabled, neither resolves in-domain truncation on the smaller Qwen3.5 backbones. With the backbone interface held fixed, these differences are consistent with heads, rather than positions, acting as the specialists that communicate through the workspace.

4.2 A SPARSE WORKSPACE SUFFICES

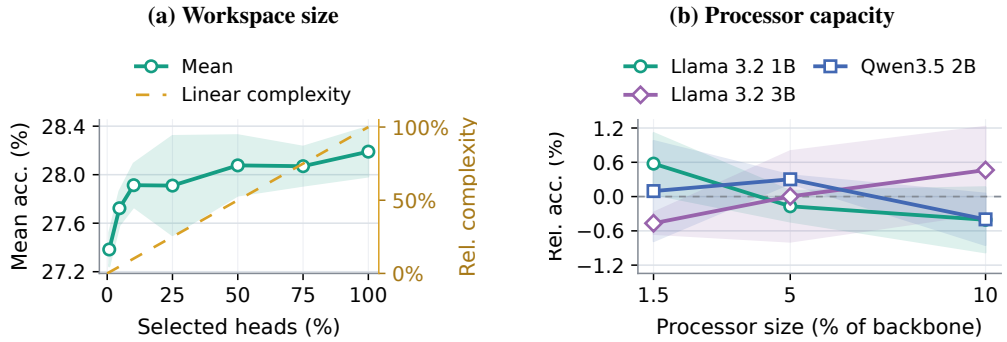


Figure 6: **Scaling ablations.** Mean accuracy over GSM8K, SVAMP, GSM-Hard, LogiQA, and Gaokao MathQA, averaged over three seeds, with earlier new-token caps (512 for GSM8K and GSM-Hard, 256 for SVAMP). **(a)** Llama 3.2 1B accuracy against the percentage of its 128 heads selected (shading: one standard error). The dashed line shows the linear relation between selected heads and relative complexity, not measured runtime. **(b)** Relative accuracy $100(a/\bar{a}-1)$ after 2,000 steps against target processor parameters as a percentage of the backbone, where $\bar{a}$ is each backbone's mean over all seeds and processor sizes (shading: one standard deviation). The workspace selects 32 heads for Llama 3.2 1B and Qwen3.5 2B, and 56 for Llama 3.2 3B.

Workspace size. A capacity-limited workspace is central to our motivation, and the data support it. On Llama 3.2 1B, accuracy varies by less than one point between a single selected head per processing step and all 128, and most of that range is covered by the first 10% of heads, which come within 0.3 points of the full workspace (Figure 6a). The brain's workspace is similarly sparse: across seven tasks and rest, it comprises only 8 of 80 brain regions, or 10% (Deco et al., 2021), and global integration relies on a small set of densely interconnected hub regions (van den Heuvel & Sporns, 2011). In Transformers, likewise, a small subset of attention heads carries most of the function: a translation model loses only 0.15 BLEU when 38 of its 48 encoder heads are pruned (Voita et al., 2019), and a large proportion of heads can be removed at test time without a significant loss (Michel et al., 2019). Because the workspace's attention scales with the number of selected source heads, a sparse workspace reaches near-optimal accuracy at a fraction of the cost of a full one.

Processor capacity. Capacity, by contrast, is not the bottleneck. Increasing the processor from 1.5% to 10% of backbone parameters changes accuracy by less than 0.4 points on each backbone, with no consistent direction (Figure 6b). What the processor is given to communicate matters more

than its size. This also bears on the comparison with BT and BGT: since added parameters do not improve GCW, its advantage over them is unlikely to be a matter of capacity and is consistent with the head-level selection that distinguishes it.

5 Discussion

Where adaptation lives. Our results suggest that where an adaptation is stored shapes what it costs. Full fine-tuning and LoRA apply one learned change to every input, so what they learn in domain, including shorter reasoning, carries over to unrelated tasks, where accuracy falls. GCW leaves the weights fixed and computes its update from the current context, and its effects are concentrated where it was trained. This is a candidate mechanism for its retention rather than a demonstrated one; comparing the magnitude of its cache updates in and out of domain would test it directly. Retention here also means preservation rather than transfer: most of GCW's KOR-Bench successes are questions the backbone already solved, and its advantage over weight-space methods narrows as backbones grow and their own retention improves (Appendix D). Memory-based adaptation may therefore matter most where weight updates forget the most.

A workspace, not a larger processor. Two results favour structure over capacity. Position-based processors with the same frozen interface forget about as much as weight-space methods, and enlarging GCW's processor brings no consistent gain, whereas a workspace of about 10% of heads suffices. Together they support the premise taken from Global Workspace Theory: what matters is selective communication between specialised components, and the useful capacity of GCW lies in which heads it selects rather than in its parameters.

Limitations. In-domain gains are noisy, particularly on smaller backbones: GCW does not improve in-domain on the Llama 3.2 backbones or on Qwen3.5-2B with thinking disabled (Table 3), and after broad adaptation it loses more mathematical accuracy than the alternatives. Its retention advantage holds on average but not everywhere. A weight-space method scores higher on KOR-Bench in 2 of 16 backbone, setting, and thinking-mode observations (Qwen3.5-27B with thinking enabled, and Qwen3.5-2B after broad adaptation), and position-based processors do so in five, most notably on Llama 3.2-3B after mathematical adaptation, where BT and BGT score 12 to 15 points higher. Thinking-mode results are available only after mathematical adaptation, due to thinking-mode's evaluation cost. The backbones also often exhaust the token budget, so part of every method's thinking-mode gain reflects finishing within budget, an efficiency gain rather than purely better reasoning. Finally, GCW and the position-based processors also differ in write location, positional encoding, and, on Qwen3.5, linear-state updates. While these differences are all inherent to their respective architectures, they should be taken into account as potentially contributing to the overall improvements, alongside the core mechanism.

Future work. Natural extensions include broader adaptation settings, such as continued pretraining, multi-task learning, and domains beyond mathematics and abstract reasoning; ablations that isolate cross-head communication and the roles of attention and linear-attention heads; combining GCW with weight-space methods, which act on complementary parts of the model; and dynamic sparsity, in which the number of selected heads adapts to the context or task.

6 Conclusion

We introduced the Global Cache Workspace, a module inspired by Global Workspace Theory that adapts a frozen language model by selecting a sparse set of heads, integrating them in a shared workspace, and broadcasting an update back into their memory. It reaches in-domain gains competitive with full fine-tuning and LoRA, loses the least out-of-domain reasoning accuracy on average across mathematical, broad, and sequential adaptation, and with thinking enabled it shortens reasoning mainly where it was trained. A workspace of about 10% of heads suffices, and performance is insensitive to processor size, consistent with the premise that selective communication between specialised components, rather than added capacity, supports adaptation. These results suggest that a model's memory, and not only its weights, is a viable place for it to learn, opening a path towards continual adaptation with frozen backbones.

AI USE STATEMENT

Generative AI tools were used for editorial assistance, including wording, grammar, syntax, and LaTeX presentation, as well as literature and reference checking and assistance with code implementation. They also provided feedback on experimental comparisons, interpretation of results, and the scope of claims. Reference metadata and cited claims were checked against original sources. The authors take responsibility for the final manuscript, implementation, and all scientific claims.

REPRODUCIBILITY STATEMENT

To support reproducibility, Section 3 describes the architecture and training objective, with cache-processing details in Appendix B. Training and evaluation protocols are described in Section 3.1 and Appendix A.1. Appendix A.2 specifies the Reasoning Gym data generation, filtering, solution construction, seeds, and splits. Appendix C reports per-backbone benchmark accuracies, generated-token counts, and truncation rates, while Appendix D defines the additional retention measures.

REFERENCES

Bernard J Baars. *In the theater of consciousness: The workspace of the mind*. Oxford University Press, 1997. 2, 4, 6

Cong Bao, Zafeirios Fountas, Temitayo Olugbade, and Nadia Bianchi-Berthouze. Multimodal data fusion based on the Global Workspace Theory. In *Proceedings of the 2020 International Conference on Multimodal Interaction*, pp. 414–422. Association for Computing Machinery, 2020. doi: 10.1145/3382507.3418849. URL https://doi.org/10.1145/3382507.3418849. 4

Max Belitsky, Dawid J. Kopiczko, Michael Dorkenwald, M. Jehanzeb Mirza, James R. Glass, Cees G. M. Snoek, and Yuki M. Asano. KV cache steering for controlling frozen LLMs. *arXiv preprint arXiv:2507.08799*, 2025. 2, 3

Yoshua Bengio. The consciousness prior. *arXiv preprint arXiv:1709.08568*, 2017. 3, 4

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P Cunningham. LoRA learns less and forgets less. *Transactions on Machine Learning Research*, 2024. 2, 3

Hugo Chateau-Laurent and Rufin VanRullen. Learning to chain operations by routing information through a global workspace. *arXiv preprint arXiv:2503.01906*, 2025. 4

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. TheoremQA: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://arxiv.org/abs/2305.12524. 7

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168. 7

Gustavo Deco, Diego Vidaurre, and Morten L. Kringelbach. Revisiting the global workspace orchestrating the hierarchical organization of the human brain. *Nature Human Behaviour*, 5:497–511, 2021. 8

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html. 2, 4

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/toy_model/index.html. 2, 4

Muhammad ElNokrashy, Badr AlKhamissi, and Mona Diab. Depth-wise attention (DWAtt): A layer fusion method for data-efficient classification. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 4665–4674, 2024. 4

Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Atri Rudra, James Y Zou, Azalia Mirhoseini, and Christopher Ré. Cartridges: Lightweight and general-purpose long context representations via self-study. In *International Conference on Learning Representations*, volume 2026, pp. 42642–42687, 2026. 3

Anastasiia Filippova, David Grangier, Marco Cuturi, and João Monteiro. Stochastic KV routing: Enabling adaptive depth-wise cache sharing. *arXiv preprint arXiv:2604.22782*, 2026. 4

Zafeirios Fountas, David Gamez, and Andreas K. Fidjeland. A neuronal global workspace for human-like control of a computer game character. In *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*, pp. 350–357. IEEE, 2011. doi: 10.1109/CIG.2011.6032027. URL https://doi.org/10.1109/CIG.2011.6032027. 4

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *International Conference on Machine Learning*, 2023. URL https://arxiv.org/abs/2211.10435. 7

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *Advances in Neural Information Processing Systems*, 38: 41340–41391, 2025. 3

Anirudh Goyal, Aniket Didolkar, Alex Lamb, Kartikeya Badola, Nan Rosemary Ke, Nasim Rahaman, Jonathan Binas, Charles Blundell, Michael Mozer, and Yoshua Bengio. Coordination among neural modules through a shared global workspace. In *International Conference on Learning Representations*, 2022. 3, 4, 5, 6

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *International Conference on Learning Representations*, volume 2024, pp. 27896–27923, 2024. 3

Wes Gurnee, Nicholas Sofroniew, Adam Pearce, Mateusz Piotrowski, Isaac Kauvar, Runjin Chen, Anna Soligo, Paul Bogdan, Euan Ong, Rowan Wang, Ben Thompson, David Abrahams, Subhash Kantamneni, Emmanuel Ameisen, Joshua Batson, and Jack Lindsey. Verbalizable representations form a global workspace in language models. *arXiv preprint arXiv:2607.15495*, 2026. 4

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. arXiv preprint arXiv:2412.06769, 2024. URL https://arxiv.org/abs/2412.06769. 3

Mike Heddes, Adel Javanmard, Kyriakos Axiotis, Gang Fu, MohammadHossein Bateni, and Vahab Mirrokni. DeepCrossAttention: Supercharging transformer residual connections. In *International Conference on Machine Learning*, 2025. 4

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. 2, 3

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning*. PMLR, 2021. 4

Arthur Juliani, Ryota Kanai, and Shuntaro Sasai. The perceiver architecture is a functional global workspace. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 44, 2022. 4

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 2, 3

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021. 2, 3

Bojie Li. Models take notes at prefill: KV cache can be editable and composable. *arXiv preprint arXiv:2606.17107*, 2026. 2, 3

Mufei Li, Shikun Liu, Dongqi Fu, Haoyu Wang, Yinglong Xia, Hong Li, Hong Yan, and Pan Li. KVEraser: Learning to steer KV cache for efficient localized context erasing. *arXiv preprint arXiv:2606.17034*, 2026. 2, 3

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2021. 2, 3

Hongzhan Lin, Zhiqi Bai, Xinmiao Zhang, Xiang Li, Siran Yang, Yunlong Xu, Jiaheng Liu, Yongchi Zhao, Jiamang Wang, Yuchi Xu, Wenbo Su, and Bo Zheng. Reconstructing KV caches with cross-layer fusion for enhanced transformers. In *International Conference on Learning Representations*, volume 2026, pp. 87101–87124, 2026. 4

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. LogiQA: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 3622–3628, 2020. URL https://arxiv.org/abs/2007.08124. 7

Luyang Liu, Jonas Pfeiffer, Jiaxing Wu, Jun Xie, and Arthur Szlam. Deliberation in latent space via differentiable cache augmentation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 39261–39274. PMLR, 2025. 2, 3

Kaijing Ma, Xinrun Du, Yunran Wang, Haoran Zhang, Zhoufutu Wen, Xingwei Qu, Jian Yang, Jiaheng Liu, Minghao Liu, Xiang Yue, Wenhao Huang, and Ge Zhang. KOR-Bench: Benchmarking language models on knowledge-orthogonal reasoning tasks, 2024. URL https://arxiv.org/abs/2410.06526. 7

George A Mashour, Pieter Roelfsema, Jean-Pierre Changeux, and Stanislas Dehaene. Conscious processing and the global neuronal workspace hypothesis. *Neuron*, 105(5):776–798, 2020. 2, 4

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989. 2, 3

Meta. Llama 3.2 model card. Hugging Face, 2024. URL https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct. 7

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32, 2019. 8

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html. 2

Adnan Oomerjee, Zafeirios Fountas, Haitham Bou Ammar, and Jun Wang. Bottlenecked transformers: Periodic KV cache consolidation for generalised reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 43350–43368, 2026. 2, 4, 17

Matteo Pagliardini, Amirkeivan Mohtashami, François Fleuret, and Martin Jaggi. DenseFormer: Enhancing information flow in transformers via depth weighted averaging. In *Advances in Neural Information Processing Systems*, volume 37, 2024. 4

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 2080–2094, 2021. URL https://arxiv.org/abs/2103.07191. 7

Qwen Team. Qwen3.5: Towards native multimodal agents, 2026. URL https://qwen.ai/blog?id=qwen3.5. 3, 7

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. REASONING GYM: Reasoning environments for reinforcement learning with verifiable rewards, 2025. URL https://arxiv.org/abs/2505.24760. 7, 15

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, Carlos Guestrin, Jed McCaleb, Yejin Choi, and Yu Sun. End-to-end test-time training for long context. *arXiv preprint arXiv:2512.23675*, 2025. 3

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. OpenMathInstruct-2: Accelerating AI for math with massive open-source instruction data. In *International Conference on Learning Representations*, volume 2025, pp. 19243–19275, 2025. 7, 15

Pedro Urbina-Rodriguez, Zafeirios Fountas, Fernando E Rosas, Jun Wang, Andrea I Luppi, Haitham Bou-Ammar, Murray Shanahan, and Pedro A. M. Mediano. A brain-like synergistic core in LLMs drives behaviour and learning. *arXiv preprint arXiv:2601.06851*, 2026. 2, 5

Martijn P. van den Heuvel and Olaf Sporns. Rich-club organization of the human connectome. *Journal of Neuroscience*, 31(44):15775–15786, 2011. 8

Rufin VanRullen and Ryota Kanai. Deep learning and the global workspace theory. *Trends in Neurosciences*, 44(9):692–704, 2021. 4

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. 2

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5797–5808, 2019. 2, 8

Heng Wang, Jielin Qiu, Wenting Zhao, Cheng Qian, Liangwei Yang, Jiawei Han, Heng Ji, Silvio Savarese, Shelby Heinecke, and Huan Wang. Random attention: Rethinking KV cache eviction for efficient reasoning. *arXiv preprint arXiv:2609.03430*, 2026. 2, 3

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://papers.nips.cc/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html. 2

Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. MUDDFormer: Breaking residual bottlenecks in transformers via multiway dynamic dense connections. In *International Conference on Machine Learning*, 2025. 4

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, 2025. 3, 18

Wenbo Zhang and Xiang Ren. WhiteMatter: All-to-all cross-layer connections via KV mixing. arXiv preprint arXiv:2608.18486, 2026. URL https://arxiv.org/abs/2608.18486. 2, 4

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in neural information processing systems*, 36:34661–34710, 2023. 2, 3

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. AGIEval: A human-centric benchmark for evaluating foundation models, 2023. URL https://arxiv.org/abs/2304.06364. 7

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *International Conference on Learning Representations*, 2025. 4

A Further Experimental Details

A.1 Training and evaluation

Training. Mathematical adaptation trains on OpenMathInstruct-2 (Toshniwal et al., 2025) for 2,000 steps with a batch size of 128. Broad adaptation trains on 128,000 generated Reasoning Gym examples for 2,000 steps, without prior mathematical adaptation (Appendix A.2); sequential adaptation applies the same Reasoning Gym stage to each method's mathematics-adapted checkpoint. Within each setting, the backbone, data order, training tokens, checkpoints, prompts, and decoding settings are matched across methods. Cache processors are sized to 1.5% of the backbone's non-embedding parameters (Appendix B.4). LoRA's rank is set to the same budget but capped at 32 to keep the update low-rank; the cap binds on only one backbone, where the budget would allow rank 34.

Benchmarks. The benchmarks form three groups, defined relative to mathematical adaptation: tasks matching the training format (GSM8K, SVAMP, and GSM-Hard, labelled GSM8K-Hard in our tables); tasks beyond that format (TheoremQA for theorem-based reasoning across mathematics, science, and finance, Gaokao MathQA for examination mathematics, and LogiQA for logical reading comprehension); and KOR-Bench, which supplies rule descriptions in context across five categories. After Reasoning Gym adaptation, KOR-Bench assesses transfer to out-of-distribution abstract reasoning, and the mathematical benchmarks assess retention outside the adaptation domain.

Decoding and scoring. After mathematical adaptation, backbones with a thinking mode are evaluated with thinking disabled and enabled, under new-token caps of 2,048 and 4,096, respectively; Reasoning Gym evaluations use thinking disabled with a 2,048-token cap. Generated tasks are scored by exact match on the extracted answer, and multiple-choice tasks by length-normalised option likelihood. Alongside accuracy, we report generated tokens and the truncation rate, the fraction of responses that reach the token cap.

A.2 Reasoning-Gym

Task mixture. We construct a fixed supervised dataset from six procedural Reasoning Gym (Stojanovski et al., 2025) generators, covering string rewriting, arithmetic, spatial tracking, and matrix transformations.[1] Table 1 specifies the task settings and solution construction. The mixture is balanced across tasks: rewriting and arithmetic each contribute one third of the examples, while spatial tracking and matrix transformations each contribute one sixth. This is a selected subset of Reasoning Gym; base conversion and greatest common divisor (GCD) share quotient–remainder arithmetic.

Sampling and admission. We retain the upstream sampling rules except for the GCD distribution and a 256-transition bound on string manipulation. For GCD, we sample the desired divisor $g = 1$ with probability 0.25, and otherwise sample g uniformly from $\{2, \ldots, 100\}$. We draw two cofactors independently and uniformly from $[\lceil 10/g \rceil, \lfloor 100{,}000/g \rfloor]$, resampling until they are coprime, and multiply both by g. Thus both branches produce inputs within the same numeric bounds. All examples require a nonempty final answer and at least two productive computational steps; GCD requires at least four Euclidean divisions. Matrix operations are sampled with equal weights across the ten upstream operation types, including rotations, reflections, value transformations, cropping, and row or column removal. We retain only final matrices with at least two rows, two columns, six cells, and two distinct values, and require at least 75% of operations to change the matrix. String rewriting stops when no rule applies or when the next state would repeat a previous state; in the latter case, the repeated state is discarded. Candidates exceeding the transition bound are rejected. These criteria define the admitted distribution and can change the sampled operation and answer frequencies.

Solution construction. We generate worked solutions deterministically by executing the inputs and operations visible in each prompt, without teacher-model inference. Every reconstructed final answer must exactly match the generator's answer and receive full credit from its verifier; any

[1] Generators are taken from Reasoning Gym revision `49b07130b3fc`.

Table 1: Reasoning Gym generation settings and supervised solutions. Numeric ranges are inclusive. Task order defines the seed index t used below.

Task	Generation settings	Worked solution
String manipulation	Strings of 8–28 characters from $\{a, b, c\}$; 6–12 rules sampled without replacement from the generator's rule set.	Apply the first applicable rule in the displayed order; record each new string and the termination condition.
Base conversion	Integer values from 256 to 1,048,575; distinct source and target bases from 2 to 16.	Accumulate source digits into a decimal value, then show successive target-base divisions and reverse the remainders.
A::B rewriting	Programs of 18 tokens sampled from `A#`, `B#`, `#A`, and `#B`.	Rewrite the leftmost applicable pair; record each replacement and resulting program until no rule applies.
Cube rotation	Random initial face colours, 2–8 rotations, and a queried face.	Track all six face colours after every rotation and report the queried face's final colour.
Matrix manipulation	Initial dimensions of 3–6 rows and columns, entries from 0 to 9, and 3–8 operations.	Apply the listed operations in order and display the matrix after each operation.
GCD	Two integers from 10 to 100,000, sampled with a controlled common factor.	Show every Euclidean division and return the last nonzero divisor.

disagreement aborts preparation. Prompts request intermediate steps and a final result enclosed in `<answer>` and `</answer>` tags. They also specify rewrite order, rotation and reflection conventions, and the right-middle character convention for even-length strings. The supervised response contains the complete worked solution followed by the tagged final answer. Chat formatting uses thinking disabled, and next-token cross-entropy is applied to the response, including the worked solution, with prompt tokens masked.

Length control. Admission requires the complete chat-formatted prompt and response, including the end token, to fit within 4,096 tokens under the pinned Qwen3.5-2B tokenizer. Overlength candidates are rejected rather than shortened during preparation. Stored solutions therefore remain complete; shorter training contexts can truncate them when loaded. For the fixed subset described below, the 1,536-token audit finds 1,634 truncated examples (1.28%), all from string manipulation or A::B, removing 0.88% of response tokens and leaving no example with only prompt tokens.

Splits and reproducibility. We first generate 700,032 training examples and 6,000 examples each for validation and test, with exactly 116,672 training and 1,000 examples per holdout for each task. For zero-based task index t in Table 1, split index $s \in \{0, 1, 2\}$ (training, validation, test), and candidate index i, the effective random seed is $24+(3t+s)10^8+i$. Candidates are processed in order until each quota is filled, with an attempt budget of 20 times the requested count per task and split. We deduplicate both whitespace-normalised prompts and structured task inputs globally across all three splits, including cube problems with identical states and rotations but different wording. Each completed split is shuffled with seed $24+s$. Validation and test contain fresh instances from the same task distribution; they do not hold out task families, compositions, or difficulty levels. KOR-Bench is excluded from generation, filtering, mixture tuning, and checkpoint selection.

Fixed training subset. The experiments use 128,000 examples sampled uniformly without replacement within each task from the full training split, using seed 24. Proportional task quotas are rounded by largest remainder with seeded tie-breaking: cube rotation and matrix manipulation each contribute 21,334 examples, and each other task contributes 21,333. We apply no additional length or difficulty filtering and shuffle the selected examples once. All methods share this fixed membership and order, together with the original validation and test splits. Generation and subset manifests record the sampling settings, seeds, tokenizer identity, exclusion counts, and file checksums.

B Architectural Details

This appendix details GCW and the two position-based processors it is compared against: BT (Oomerjee et al., 2026), which consolidates each layer separately, and BGT, which extends BT across layers. All three attach to a frozen backbone, run on the same schedule, and write gated residual updates into the cache. They differ in what forms a workspace token, whether heads compete for access, and which entries are rewritten (Table 2). Heads $c \in \{1, \dots, N\}$ comprise N_{kv} attention (KV) heads, one per layer and KV head of the full-attention layers, and, on hybrid backbones, N_{lin} linear-attention heads, so $N = N_{\text{kv}} + N_{\text{lin}}$. Processors have width d_p; d_h is the backbone's KV head dimension.

Table 2: **Cache processors compared.** All three share the invocation schedule, recency window, and gated residual write.

	BT	BGT	GCW
Workspace token	Selected position within one layer	Selected position across all layers	One head: KV history or linear-attention state
Position selection	Shared by all layers	Shared by all layers	Per head
Time compression	None	None	Attention pooling
Competition	Among positions	Among positions	Among positions, then heads
Processor attention	$m \times m$, one processor per layer	$m \times m$, shared	$N \times b$, shared
Rotary embeddings	Kept	Kept	Removed, reapplied on write
Entries rewritten	Selected positions of its layer	Selected positions of all layers	Latest position of every head
Linear-attention states	Unchanged	Unchanged	Read and updated

B.1 Invocation schedule

The original BT consolidates the cache at newline-delimited reasoning boundaries. Here all three processors instead run between fixed causal windows of $P = 64$ tokens: the backbone processes each window as usual, and the processor rewrites the cache before the next window is read. The schedule is the same during teacher-forced training, prefill, and decoding, so a sequence of length T receives $\lceil T/P \rceil - 1$ invocations and no invocation reads a token the backbone has not yet processed. Next-token gradients pass through every rewrite into the processor.

B.2 Position selection and workspace tokens

All three processors retain the P most recent cached positions and the $k = 32$ older positions of largest salience, in temporal order, giving $m = P + k$ positions. BT and BGT select one set of positions shared by all layers and heads, ranked by a single salience score per position; for BT, this is the norm of the backbone's attention output averaged over layers. Their workspace tokens are these positions: BT forms one token per position and layer from that layer's KV heads (Equation 1 applied to each layer separately), and BGT concatenates all layers into one token per position (Equation 1). Keys keep their rotary embeddings, and the history is not compressed.

GCW instead selects positions separately for each head and compresses each head's history into one token. Let $\mathbf{x}_{c,t} = [\mathbf{k}_{c,t}; \mathbf{v}_{c,t}] \in \mathbb{R}^{2d_h}$ concatenate the key and value of KV head c at cached position t, with rotary embeddings removed from the key. Each head retains positions by salience $\|\mathbf{x}_{c,t}\|_2^2$, giving a history $\mathbf{X}_c \in \mathbb{R}^{m \times 2d_h}$ whose last row $\mathbf{x}_{c,m}$ is the latest cached position. Attention pooling with a query from that position gives one workspace token:

$$\bar{\mathbf{X}}_c = \mathrm{RMSNorm}(\mathbf{X}_c), \qquad \boldsymbol{\alpha}_c = \mathrm{softmax}\left(\frac{\bar{\mathbf{X}}_c W_q \, \bar{\mathbf{x}}_{c,m}}{\sqrt{2d_h}}\right), \qquad \mathbf{u}_c = W_{\text{in}} \, \bar{\mathbf{X}}_c^\top \boldsymbol{\alpha}_c \in \mathbb{R}^{d_p}, \quad (5)$$

with W_q and W_{in} shared across heads. Removing rotary embeddings makes the token depend on the content of the history rather than on absolute positions. Stacking the tokens of all KV heads and linear-attention heads (Appendix B.5) gives $\mathbf{U} \in \mathbb{R}^{N \times d_p}$.

B.3 HEAD SELECTION

In BT and BGT, every head at a selected position enters the same token, so competition acts only among positions. GCW adds competition among heads, selected by magnitude. A KV head is scored by the total salience of its retained history, $s_c = \sum_t \|\mathbf{x}_{c,t}\|_2^2$, and a linear-attention head by the squared norm of its state readout (Appendix B.5). The top b_{kv} KV heads and the top b_{lin} linear-attention heads are selected separately; their union $\mathcal{J}$, with $b = b_{\text{kv}} + b_{\text{lin}}$, supplies the workspace sources in Equation 2. Selection is a hard top-k: gradients reach the processor through the selected tokens, not through the choice itself.

B.4 PROCESSOR AND WRITE-BACK

All three processors stack L_p pre-norm Transformer layers of width d_p, each with attention followed by a SwiGLU feed-forward block, between input and output projections, and scale their update by a learned scalar gate $\sigma(g)$. The gate logit starts at $g_0 = -5$ ($\sigma(g_0) \approx 0.007$) on Llama 3.2 backbones and at $g_0 = -2$ ($\sigma(g_0) \approx 0.12$) on Qwen3.5 backbones, whose processors did not learn from $g_0 = -5$ and needed larger initial updates to receive a learning signal early in training. On every backbone, each processor's width is chosen so that all of its added parameters, including the compression, input, and output projections, total 1.5% of the backbone's non-embedding parameters. BT runs one such processor per layer, with its own projections and gate; BGT runs a single processor over all layers. In both, the m position tokens attend to one another with rotary embeddings at their cache positions, and the gated residual is written back to every selected position (Equation 1), so older entries can be revised.

In GCW, all N head tokens supply queries and only the b selected tokens supply keys and values. Head tokens carry no positional encoding. Each output $\mathbf{o}_c$, taken after a final RMSNorm, is decoded back to the head that produced it by the gated residual of Equation 3, with one gate and one output projection shared by all heads. Rotary embeddings are reapplied to the key half of $\mathbf{x}'_{c,m}$, which replaces the latest cache entry of every KV head, selected or not; older entries are read-only.

B.5 LINEAR-ATTENTION STATES

A Gated DeltaNet head h in layer ℓ (Yang et al., 2025) stores a state $\mathbf{S}_{\ell,h} \in \mathbb{R}^{d_k \times d_v}$ instead of per-position entries. BT and BGT index their tokens by position, so on hybrid backbones they process only the full-attention layers and leave these states unchanged. GCW reads each state with a query. The context $\mathbf{c}_\ell$ is the mean of the tokens $\mathbf{u}_c$ of the KV heads in the first full-attention layer after ℓ, and a layer-specific projection gives one unit-norm query per head:

$$\begin{aligned} \mathbf{q}_{\ell,h} &= \text{normalise}\left(\left[W_q^{(\ell)}\,\text{RMSNorm}(\mathbf{c}_\ell)\right]_h\right), \qquad \mathbf{r}_{\ell,h} = \mathbf{S}_{\ell,h}^\top \mathbf{q}_{\ell,h}, \\ \mathbf{u}_{\ell,h} &= W_{\text{in}}^{\text{lin}}\,\text{RMSNorm}\left([\mathbf{q}_{\ell,h}; \mathbf{r}_{\ell,h}]\right). \end{aligned} \tag{6}$$

The token $\mathbf{u}_{\ell,h}$ joins $\mathbf{U}$ and is scored for selection by $\|\mathbf{r}_{\ell,h}\|_2^2$. After processing, an adapter projection maps its output to a key and value direction, $[\Delta\mathbf{k}; \Delta\mathbf{v}] = W_{\text{out}}^{\text{lin}}\mathbf{o}_{\ell,h}$, written into the state as a gated rank-one update:

$$\mathbf{S}'_{\ell,h} = \mathbf{S}_{\ell,h} + \sigma(g)\,\hat{\mathbf{k}}\,\Delta\mathbf{v}^\top, \qquad \hat{\mathbf{k}} = \frac{\Delta\mathbf{k}}{\|\Delta\mathbf{k}\|_2}, \tag{7}$$

with the same gate as Equation 3. The update has the form of a single associative-memory write: $\hat{\mathbf{k}}$ chooses where in the state to write, and $\sigma(g)\Delta\mathbf{v}$ what and how much. Every linear-attention head receives an update; convolution states are unchanged.

B.6 COMPUTATIONAL COST

All three processors score every cached position, at a cost linear in the cache length T, and each invocation is amortised over P tokens. The processors then differ in what their attention scales with. BT and BGT attend among m position tokens, at $O(L_p m^2 d_p)$ per processor, and BGT's projections act on concatenations of all layers. GCW encodes heads at $O\big(N_{\text{kv}}(m\,d_h + d_h^2 + d_h d_p) + N_{\text{lin}} d_k d_v\big)$ and processes them at $O(L_p N b\,d_p + L_p N d_p d_{\text{ff}})$ for feed-forward width d_{ff}; only the attention term depends on the source budget, and it grows linearly in b.

C Extended Results

C.1 Specialisation and retention after mathematical adaptation

Figures 7 and 8 summarise accuracy and response-length changes.

The aim of this section is to assess how our models and baselines, trained on mathematical data, perform on both matched and unseen mathematical tasks, as well as their generalisation or forgetting in broader, out-of-distribution reasoning tasks. To this end, we adapt models on OpenMathInstruct-2 and compare them with backbone baselines on our benchmarks, including LogiQA and KOR-Bench for transfer beyond mathematics (see Section 3.1).

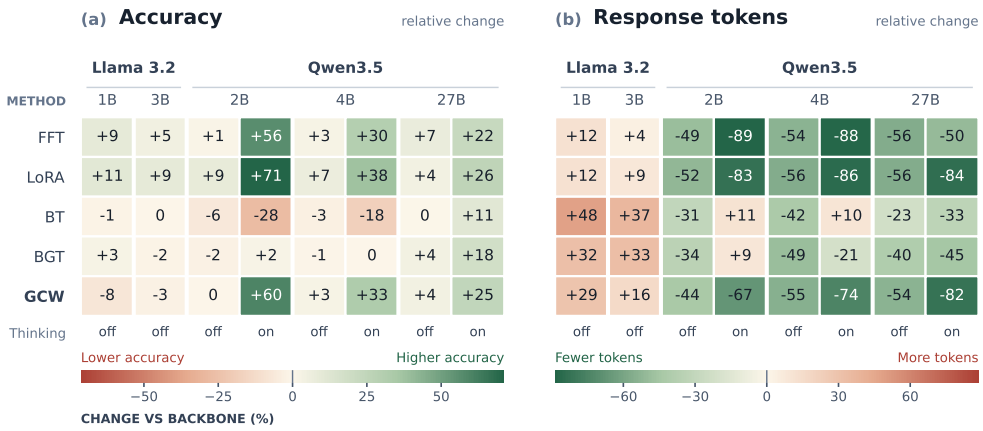


Figure 7: **Mathematical and LogiQA tasks after mathematical adaptation.** Percentage changes in accuracy and mean response length relative to the backbone, with baselines matched by backbone, task group, and thinking mode (Table 3). Accuracy averages six tasks, including LogiQA. Mean response length pools the four generation tasks. The panels use independent linear colour scales centred on zero. Forest green indicates higher accuracy or fewer tokens; terracotta indicates lower accuracy or more tokens. Off/on denote thinking disabled and enabled. Annotations are rounded to whole percentages.

Table 3: **Mathematical benchmark accuracy (%) after mathematical training.** Adapted models are trained on mathematical data; backbone rows provide baselines without this adaptation. Columns are grouped as in Section 3.1: tasks matching the training distribution, then in-domain but unseen tasks. Generated-task scores show thinking off/on where available, with 2,048 and 4,096 new-token caps, respectively. LogiQA and Gaokao use normalised option likelihood; these scores are shared across thinking modes. Math averages all six benchmarks for every model size. Mean response tokens equal total generated response tokens divided by the number of responses; totals are in millions. Token statistics cover all four generation tasks for every model size. — marks thinking-mode entries for Llama 3.2, which has no thinking mode. Trunc. reports the percentage of responses stopped by the generation cap, pooled over the same tasks as token statistics.

	Matched			Unseen				Generation		
Method	GSM8K	SVAMP	GSM8K-Hard	ThmQA	LogiQA	Gaokao	**Avg.**	Mean/resp.	Total (M)	Trunc. (%)
Llama 3.2-1B										
Backbone	43.37/—	63.33/—	12.81/—	11.25/—	30.41	27.07	**31.37/—**	260.9/—	0.975/—	3.26/—
FFT	50.64/—	69.33/—	12.89/—	14.00/—	29.95	28.21	**34.17/—**	292.0/—	1.092/—	5.03/—
LoRA	50.49/—	70.67/—	14.18/—	13.88/—	29.95	29.06	**34.70/—**	292.4/—	1.093/—	5.46/—
BT	43.67/—	63.33/—	12.43/—	10.38/—	29.80	26.50	**31.02/—**	385.8/—	1.442/—	10.11/—
BGT	45.41/—	65.33/—	12.05/—	14.12/—	30.26	26.21	**32.23/—**	345.0/—	1.290/—	7.97/—
GCW	38.82/—	60.67/—	10.61/—	8.75/—	29.19	25.93	**28.99/—**	337.8/—	1.263/—	7.06/—
Llama 3.2-3B										
Backbone	69.90/—	78.00/—	24.26/—	15.38/—	29.34	31.05	**41.32/—**	265.1/—	0.991/—	2.70/—
FFT	72.55/—	84.67/—	24.56/—	19.62/—	30.11	28.77	**43.38/—**	274.5/—	1.026/—	3.58/—
LoRA	75.36/—	84.33/—	25.93/—	22.62/—	31.64	30.77	**45.11/—**	288.2/—	1.077/—	4.57/—
BT	68.08/—	81.00/—	23.65/—	15.88/—	30.26	27.92	**41.13/—**	362.4/—	1.355/—	7.76/—
BGT	66.49/—	81.33/—	21.46/—	17.38/—	29.65	27.92	**40.70/—**	352.1/—	1.316/—	6.98/—
GCW	67.85/—	82.00/—	22.67/—	11.00/—	29.49	28.21	**40.20/—**	307.8/—	1.150/—	4.31/—
Qwen3.5-2B										
Backbone	66.94/34.50	84.33/53.67	38.21/11.14	21.38/14.75	31.03	33.62	**45.92/29.78**	623.3/3643.9	2.330/13.621	9.90/84.38
FFT	72.25/73.24	85.33/84.67	40.49/39.04	21.88/23.38	30.72	28.49	**46.53/46.59**	319.3/417.0	1.194/1.559	4.57/4.55
LoRA	76.12/76.95	85.67/88.00	44.28/46.02	27.88/29.38	32.26	33.33	**49.92/50.99**	301.3/618.9	1.126/2.313	3.88/9.42
BT	60.12/21.46	76.33/24.33	32.83/9.33	25.62/11.50	29.95	32.76	**42.94/21.56**	433.0/4055.9	1.619/15.161	8.86/94.54
BGT	64.97/39.42	77.67/47.00	36.85/20.70	25.00/10.75	30.72	33.62	**44.80/30.37**	411.0/3983.4	1.536/14.890	8.03/95.37
GCW	65.73/73.69	82.67/84.00	39.35/41.62	26.75/26.62	28.73	31.34	**45.76/47.67**	351.7/1202.9	1.315/4.496	6.10/20.89
Qwen3.5-4B										
Backbone	80.21/64.52	88.67/73.67	51.86/30.55	29.50/14.75	35.33	34.19	**53.29/42.17**	629.0/2979.2	2.351/11.136	7.09/53.72
FFT	83.85/83.93	89.67/90.00	47.16/47.23	33.38/33.88	34.72	39.60	**54.73/54.89**	291.2/370.7	1.088/1.386	2.38/2.89
LoRA	86.43/86.88	89.67/91.00	52.08/53.30	39.25/42.12	36.41	39.03	**57.14/58.12**	275.2/425.9	1.029/1.592	2.33/3.93
BT	73.84/47.84	89.00/50.00	45.34/24.26	35.88/20.12	34.10	31.62	**51.63/34.66**	364.6/3268.4	1.363/12.217	4.92/76.89
BGT	80.36/57.39	86.67/67.00	48.07/31.01	35.12/31.00	33.79	32.19	**52.70/42.06**	320.8/2359.9	1.199/8.821	4.68/52.43
GCW	83.40/86.88	86.67/89.67	51.55/51.63	35.12/36.00	34.10	37.61	**54.74/55.98**	280.9/771.5	1.050/2.884	2.57/10.65
Qwen3.5-27B										
Backbone	84.23/76.27	91.33/77.00	54.74/40.64	32.12/16.88	39.94	59.54	**60.32/51.71**	594.2/2561.9	2.221/9.576	5.51/38.60
FFT	89.99/92.87	91.00/93.00	54.28/54.36	49.38/35.50	41.78	62.39	**64.80/63.32**	259.0/1271.3	0.968/4.752	1.28/18.99
LoRA	90.07/92.12	88.33/93.33	53.07/55.72	47.62/50.88	43.47	55.27	**62.97/65.13**	259.7/415.9	0.971/1.555	1.34/2.73
BT	81.12/84.31	90.00/78.00	46.17/47.84	43.75/33.75	40.86	60.68	**60.43/57.57**	459.5/1717.4	1.718/6.420	3.64/28.71
BGT	81.96/80.97	90.33/84.00	51.48/47.23	46.38/48.38	43.93	62.11	**62.70/61.10**	359.3/1413.4	1.343/5.283	2.01/20.65
GCW	87.72/90.14	90.00/93.67	54.13/55.72	44.88/46.12	45.62	55.56	**62.98/64.47**	275.2/468.7	1.029/1.752	1.71/3.69

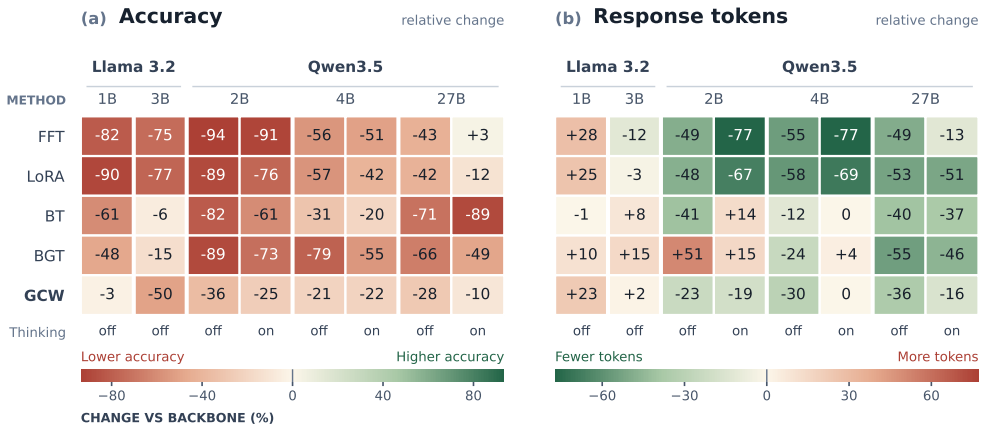


Figure 8: **KOR-Bench after mathematical adaptation.** Percentage changes in accuracy and mean response length relative to the backbone, with baselines matched by backbone, task group, and thinking mode (Table 4). Accuracy averages five tasks. Mean response length pools all 1,250 responses. The panels use independent linear colour scales centred on zero. Forest green indicates higher accuracy or fewer tokens; terracotta indicates lower accuracy or more tokens. Off/on denote thinking disabled and enabled. Annotations are rounded to whole percentages.

Table 4: **KOR-Bench zero-shot accuracy (%) after mathematical training.** Adapted models are trained on mathematical data; backbone rows provide baselines without this adaptation. All runs use chat formatting. Entries report thinking off/on, with new-token caps of 2,048/4,096; — marks thinking-mode entries for Llama 3.2, which has no thinking mode. Each category contains 250 questions, so Avg. is both the unweighted category mean and accuracy over all 1,250 questions. Mean response tokens equal total generated response tokens divided by 1,250; totals are in millions. Retained is the percentage of the matched pretrained backbone's correct answers that remain correct; Recovered is the percentage of its incorrect answers that become correct, pairing all 1,250 questions within the same thinking mode. Trunc. reports the percentage of responses stopped by the generation cap, pooled over the same tasks as token statistics.

	KOR-Bench accuracy						Backbone overlap (%)		Generation		
Method	Operation	Logic	Cipher	Puzzle	Ctr-factual	**Avg.**	Retained	Recovered	Mean/resp.	Total (M)	Trunc. (%)
Llama 3.2-1B											
Backbone	30.00/—	3.20/—	2.00/—	1.60/—	50.40/—	**17.44/—**	100.00/—	0.00/—	375.2/—	0.469/—	8.88/—
FFT	5.20/—	0.80/—	0.00/—	0.40/—	9.60/—	**3.20/—**	11.93/—	1.36/—	481.0/—	0.601/—	13.20/—
LoRA	4.40/—	0.40/—	0.00/—	0.00/—	3.60/—	**1.68/—**	6.42/—	0.68/—	470.6/—	0.588/—	14.80/—
BT	14.40/—	13.20/—	0.00/—	0.00/—	6.80/—	**6.88/—**	21.56/—	3.78/—	373.3/—	0.467/—	12.64/—
BGT	8.80/—	16.40/—	0.00/—	0.00/—	20.00/—	**9.04/—**	27.52/—	5.14/—	413.8/—	0.517/—	13.92/—
GCW	31.20/—	2.80/—	0.80/—	1.20/—	48.40/—	**16.88/—**	72.02/—	5.23/—	461.1/—	0.576/—	13.68/—
Llama 3.2-3B											
Backbone	64.40/—	12.00/—	6.40/—	2.80/—	80.80/—	**33.28/—**	100.00/—	0.00/—	407.1/—	0.509/—	9.44/—
FFT	11.60/—	1.20/—	0.00/—	0.40/—	29.20/—	**8.48/—**	22.60/—	1.44/—	359.7/—	0.450/—	7.76/—
LoRA	18.40/—	0.80/—	0.40/—	0.40/—	18.40/—	**7.68/—**	19.23/—	1.92/—	396.1/—	0.495/—	9.12/—
BT	54.80/—	10.80/—	5.20/—	3.60/—	82.00/—	**31.28/—**	81.01/—	6.47/—	440.8/—	0.551/—	11.60/—
BGT	46.00/—	6.00/—	4.40/—	3.60/—	82.00/—	**28.40/—**	74.52/—	5.40/—	466.4/—	0.583/—	12.72/—
GCW	62.40/—	4.80/—	7.20/—	2.00/—	6.40/—	**16.56/—**	37.26/—	6.24/—	414.4/—	0.518/—	9.84/—
Qwen3.5-2B											
Backbone	85.20/77.20	33.20/18.80	16.00/18.40	7.60/3.20	89.60/75.20	**46.32/38.56**	100.00/100.00	0.00/0.00	813.7/3142.1	1.017/3.928	25.76/65.92
FFT	11.60/14.80	2.00/0.00	0.80/1.20	0.00/0.40	0.40/0.40	**2.96/3.36**	5.18/6.64	1.04/1.30	417.4/716.8	0.522/0.896	9.04/11.44
LoRA	16.40/20.80	2.00/4.00	6.80/6.40	0.40/1.20	0.40/13.60	**5.20/9.20**	8.12/16.39	2.68/4.69	425.3/1050.6	0.532/1.313	10.24/19.28
BT	22.00/50.80	4.00/8.40	1.60/5.20	1.20/1.20	13.20/10.40	**8.40/15.20**	15.54/31.95	2.24/4.69	482.6/3567.3	0.603/4.459	16.96/76.00
BGT	14.00/20.00	8.40/6.40	0.40/6.80	2.00/0.80	0.40/18.00	**5.04/10.40**	8.46/21.99	2.09/3.12	1226.5/3626.6	1.533/4.533	56.24/86.24
GCW	13.20/31.60	31.60/22.00	12.00/16.00	0.80/1.60	89.60/73.20	**29.44/28.88**	56.30/60.79	6.26/8.85	626.3/2541.6	0.783/3.177	18.24/56.32
Qwen3.5-4B											
Backbone	87.20/82.00	53.60/46.80	52.00/50.80	14.80/12.00	96.40/92.80	**60.80/56.88**	100.00/100.00	0.00/0.00	840.8/2880.6	1.051/3.601	25.68/52.40
FFT	29.60/34.00	6.00/6.80	16.40/17.60	3.60/3.60	77.60/77.60	**26.64/27.92**	41.58/42.05	3.47/9.28	379.1/659.4	0.474/0.824	6.40/9.68
LoRA	34.80/40.40	1.60/6.40	22.00/26.00	5.60/5.20	67.60/88.00	**26.32/33.20**	40.39/50.77	4.49/10.02	355.6/888.2	0.445/1.110	6.32/14.96
BT	40.00/68.40	36.00/32.80	31.60/38.80	6.80/2.40	94.40/84.80	**41.76/45.44**	66.58/73.84	3.27/7.98	738.9/2872.1	0.924/3.590	26.96/64.88
BGT	19.60/48.80	8.00/9.60	6.80/20.00	2.80/3.60	27.60/46.00	**12.96/25.60**	20.53/40.65	1.22/5.75	636.3/2991.9	0.795/3.740	21.68/69.04
GCW	54.80/75.60	46.80/38.00	38.40/34.40	4.80/4.40	94.00/70.80	**47.76/44.64**	74.61/70.18	6.12/10.95	589.9/2868.6	0.737/3.586	13.44/60.80
Qwen3.5-27B											
Backbone	95.20/87.60	65.20/58.00	66.40/51.60	23.20/17.60	95.60/94.80	**69.12/61.92**	100.00/100.00	0.00/0.00	771.8/2713.1	0.965/3.391	21.20/44.16
FFT	38.00/81.20	18.00/63.20	36.40/60.00	11.20/19.60	94.00/94.80	**39.52/63.76**	54.05/89.66	6.99/21.64	393.6/2371.3	0.492/2.964	7.84/37.44
LoRA	42.80/64.80	28.80/59.60	35.60/41.60	11.60/12.00	82.40/94.40	**40.24/54.48**	55.21/78.04	6.74/16.18	363.6/1316.4	0.455/1.646	5.68/23.04
BT	50.80/22.00	18.00/2.80	3.60/1.20	10.80/6.00	16.40/3.20	**19.92/7.04**	28.12/9.82	1.55/2.52	461.3/1712.4	0.577/2.140	14.48/34.08
BGT	12.00/20.40	8.40/22.00	4.80/19.20	4.00/9.20	87.60/87.20	**23.36/31.60**	33.10/46.12	1.55/7.98	348.8/1454.4	0.436/1.818	6.08/27.76
GCW	36.00/75.60	64.00/54.80	41.60/48.00	11.60/10.80	96.00/89.20	**49.84/55.68**	69.91/80.36	4.92/15.55	492.6/2279.8	0.616/2.850	11.60/48.96

C.2 Generalisation and retention after reasoning adaptation

Figures 9 and 10 summarise accuracy and response-length changes.

The aim of this section is to assess how our models and baselines, trained on a broader, abstract reasoning dataset, perform on out-of-distribution abstract reasoning tasks, as well as the now out-of-domain mathematical tasks. To this end, we adapt models directly from the backbone on 128,000 examples from our generated Reasoning Gym dataset for 2,000 optimisation steps, without prior mathematical adaptation, and compare them with backbone baselines (see Section 3.1). Table 5 reports the available KOR-Bench evaluations, while Table 6 reports transfer to mathematical benchmarks and LogiQA.

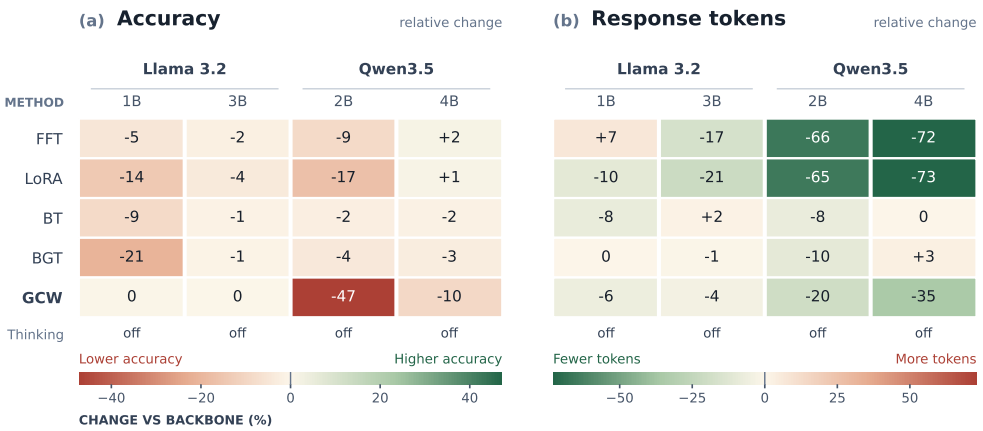


Figure 9: **Mathematical and LogiQA tasks after reasoning adaptation.** Percentage changes in accuracy and mean response length relative to the backbone, with baselines matched by backbone, task group, and thinking mode (Table 6). Accuracy averages five tasks, including LogiQA and excluding TheoremQA. Mean response length pools GSM8K, SVAMP, and GSM8K-Hard. The panels use independent linear colour scales centred on zero. Forest green indicates higher accuracy or fewer tokens; terracotta indicates lower accuracy or more tokens. Off/on denote thinking disabled and enabled. Annotations are rounded to whole percentages.

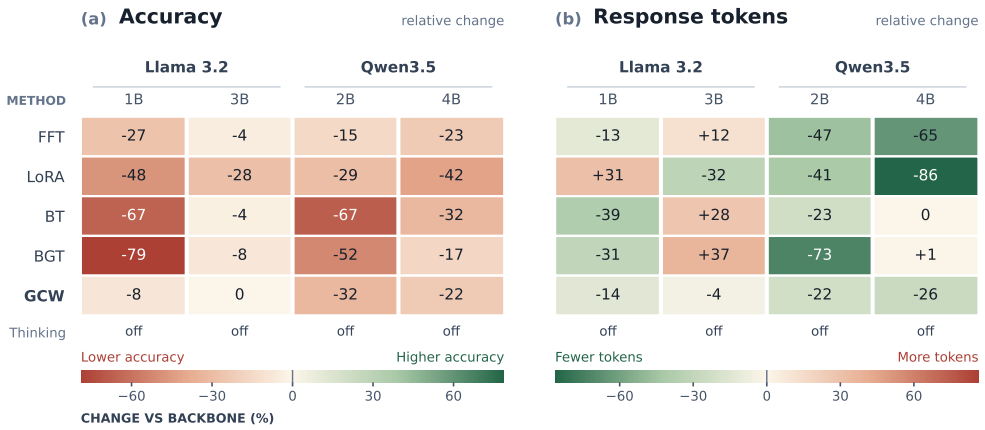


Figure 10: **KOR-Bench after reasoning adaptation.** Percentage changes in accuracy and mean response length relative to the backbone, with baselines matched by backbone, task group, and thinking mode (Table 5). Accuracy averages five tasks. Mean response length pools all 1,250 responses. The panels use independent linear colour scales centred on zero. Forest green indicates higher accuracy or fewer tokens; terracotta indicates lower accuracy or more tokens. Off/on denote thinking disabled and enabled. Annotations are rounded to whole percentages.

Table 5: **KOR-Bench zero-shot accuracy (%) after Reasoning Gym training.** Backbone baselines are reproduced from Table 4 with thinking disabled. All runs use chat formatting with thinking disabled and a 2,048-new-token cap. Each category contains 250 questions, so Avg. is both the unweighted category mean and accuracy over all 1,250 questions. Mean response tokens equal total generated response tokens divided by 1,250; totals are in millions. Retained is the percentage of the matched pretrained backbone's correct answers that remain correct; Recovered is the percentage of its incorrect answers that become correct, pairing all 1,250 questions within the same thinking mode. Trunc. reports the percentage of responses stopped by the generation cap, pooled over the same tasks as token statistics.

	KOR-Bench accuracy						Backbone overlap (%)		Generation		
Method	Operation	Logic	Cipher	Puzzle	Ctr-factual	**Avg.**	Retained	Recovered	Mean/resp.	Total (M)	Trunc. (%)
Llama 3.2-1B											
Backbone	30.00	3.20	2.00	1.60	50.40	**17.44**	100.00	0.00	375.2	0.469	8.88
FFT	13.60	2.80	2.00	1.60	44.00	**12.80**	48.62	5.23	327.2	0.409	12.24
LoRA	9.60	2.40	2.40	0.00	31.20	**9.12**	31.65	4.36	492.7	0.616	18.48
BT	1.60	4.80	1.60	0.00	21.20	**5.84**	20.18	2.81	230.5	0.288	9.04
BGT	4.00	4.00	0.80	0.00	9.60	**3.68**	11.47	2.03	258.9	0.324	9.84
GCW	24.40	3.60	0.40	2.00	49.60	**16.00**	79.36	2.62	323.5	0.404	7.52
Llama 3.2-3B											
Backbone	64.40	12.00	6.40	2.80	80.80	**33.28**	100.00	0.00	407.1	0.509	9.44
FFT	54.40	17.60	3.20	2.00	82.00	**31.84**	77.16	9.23	455.6	0.570	12.80
LoRA	18.00	15.20	3.20	1.60	81.20	**23.84**	58.41	6.59	277.8	0.347	9.28
BT	58.80	13.20	5.20	2.40	80.80	**32.08**	85.34	5.52	519.5	0.649	17.68
BGT	53.20	14.00	4.40	1.60	80.00	**30.64**	82.21	4.92	556.9	0.696	19.36
GCW	64.00	14.00	6.80	2.40	80.00	**33.44**	90.14	5.16	390.2	0.488	8.00
Qwen3.5-2B											
Backbone	85.20	33.20	16.00	7.60	89.60	**46.32**	100.00	0.00	813.7	1.017	25.76
FFT	66.80	35.20	10.80	2.80	81.20	**39.36**	72.37	10.88	435.1	0.544	12.40
LoRA	60.80	26.40	4.40	3.20	69.20	**32.80**	62.52	7.15	476.9	0.596	14.72
BT	29.60	18.80	0.80	1.20	25.20	**15.12**	29.36	2.83	629.5	0.787	28.56
BGT	10.00	13.20	2.80	1.20	83.20	**22.08**	42.83	4.17	222.8	0.279	9.04
GCW	40.00	21.60	4.40	1.20	89.20	**31.28**	62.00	4.77	633.5	0.792	22.88
Qwen3.5-4B											
Backbone	87.20	53.60	52.00	14.80	96.40	**60.80**	100.00	0.00	840.8	1.051	25.68
FFT	84.00	41.60	8.80	10.80	89.20	**46.88**	71.84	8.16	293.6	0.367	9.68
LoRA	32.40	38.40	7.60	4.80	92.40	**35.12**	55.00	4.29	114.7	0.143	2.96
BT	52.00	22.00	30.40	8.40	93.20	**41.20**	64.08	5.71	842.0	1.052	25.92
BGT	57.20	42.80	47.20	11.60	93.60	**50.48**	78.29	7.35	846.7	1.058	25.12
GCW	72.40	36.80	24.80	8.40	94.00	**47.28**	74.74	4.69	620.6	0.776	18.16

Table 6: **Mathematical and LogiQA benchmark accuracy (%) after Reasoning Gym training.** Models are adapted directly from their backbones. Backbone baselines are reproduced from Table 3 with thinking disabled; averages and token statistics exclude TheoremQA. Generation tasks use chat formatting with thinking disabled and a 2,048-new-token cap, and are scored by exact match on the extracted answer. LogiQA and Gaokao use length-normalised option likelihood. Avg. is the unweighted mean over the five displayed benchmarks. Mean response tokens equal total generated response tokens divided by the 2,938 responses across GSM8K, SVAMP, and GSM8K-Hard; totals are in millions. TheoremQA is excluded from both averages and token statistics. Trunc. reports the percentage of responses stopped by the generation cap, pooled over the same tasks as token statistics.

	Benchmark accuracy						Generation		
Method	GSM8K	SVAMP	GSM8K-Hard	LogiQA	Gaokao	**Avg.**	Mean/resp.	Total (M)	Trunc. (%)
Llama 3.2-1B									
Backbone	43.37	63.33	12.81	30.41	27.07	**35.40**	178.1	0.523	1.12
FFT	38.59	61.33	10.99	30.41	27.07	**33.68**	190.4	0.559	2.55
LoRA	31.77	57.00	8.04	29.65	25.36	**30.36**	159.6	0.469	1.97
BT	34.80	57.67	10.54	31.64	26.78	**32.29**	163.6	0.481	1.91
BGT	25.63	49.00	6.90	30.41	28.21	**28.03**	178.3	0.524	2.79
GCW	43.52	63.67	11.90	30.72	27.92	**35.55**	166.9	0.490	0.82
Llama 3.2-3B									
Backbone	69.90	78.00	24.26	29.34	31.05	**46.51**	196.6	0.578	1.33
FFT	66.49	76.67	23.81	29.80	31.34	**45.62**	163.7	0.481	1.36
LoRA	62.85	77.00	23.12	31.03	29.91	**44.78**	154.9	0.455	1.46
BT	68.23	78.33	23.50	31.34	28.21	**45.92**	200.9	0.590	2.38
BGT	67.32	79.67	23.65	28.73	31.34	**46.14**	195.3	0.574	2.11
GCW	69.22	79.00	24.26	31.18	29.34	**46.60**	189.2	0.556	1.19
Qwen3.5-2B									
Backbone	66.94	84.33	38.21	31.03	33.62	**50.83**	458.4	1.347	4.66
FFT	62.77	74.00	35.48	29.65	30.48	**46.48**	157.8	0.464	2.18
LoRA	52.69	68.33	29.26	31.49	29.63	**42.28**	159.6	0.469	3.06
BT	66.87	80.33	37.07	30.26	33.33	**49.57**	424.0	1.246	4.32
BGT	63.61	81.00	37.23	28.42	32.76	**48.60**	411.6	1.209	5.04
GCW	25.25	41.33	11.83	28.88	26.78	**26.81**	367.6	1.080	8.03
Qwen3.5-4B									
Backbone	80.21	88.67	51.86	35.33	34.19	**58.05**	489.2	1.437	3.78
FFT	83.62	89.00	54.36	35.94	32.48	**59.08**	134.7	0.396	0.37
LoRA	84.53	87.33	56.86	34.41	31.05	**58.84**	132.3	0.389	0.34
BT	81.05	86.33	50.80	34.25	32.48	**56.98**	491.1	1.443	4.08
BGT	78.77	88.67	48.98	33.33	32.19	**56.39**	505.2	1.484	4.66
GCW	69.83	84.00	43.52	32.87	32.19	**52.48**	318.5	0.936	3.30

C.3 Generalisation and retention after sequential domain adaptation

Figures 11 and 12 summarise accuracy and response-length changes.

We assess whether adaptation to abstract reasoning preserves mathematical capabilities acquired in an earlier training stage. Each method is initialised from its own checkpoint after 2,000 optimisation steps on 128,000 mathematical examples and then trained for a further 2,000 steps on the same 128,000 Reasoning Gym examples used in the preceding subsection. The second stage retains the method-specific architecture and trainable parameters and starts with a fresh optimiser and learning-rate schedule. Table 7 reports generalisation to KOR-Bench, while Table 8 reports mathematical benchmark and LogiQA performance after the second stage. The backbone baselines are repeated for reference; comparisons with the mathematics-only results assess retention, and comparisons with direct Reasoning Gym adaptation assess the effect of prior mathematical training. All reported evaluations use thinking disabled.

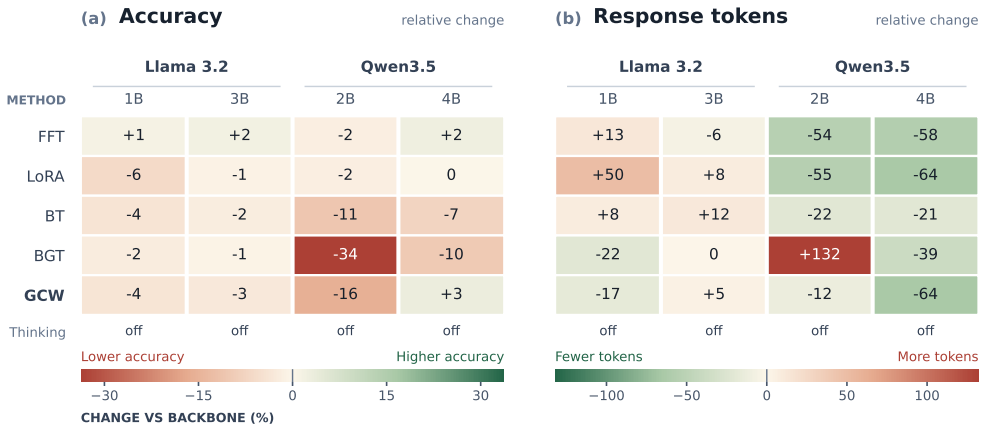


Figure 11: **Mathematical and LogiQA tasks after sequential adaptation.** Percentage changes in accuracy and mean response length relative to the backbone, with baselines matched by backbone and task group (Table 8). Accuracy averages five tasks, including LogiQA and excluding TheoremQA. Mean response length pools GSM8K, SVAMP, and GSM8K-Hard. All evaluations use thinking disabled. The panels use independent linear colour scales centred on zero. Forest green indicates higher accuracy or fewer tokens; terracotta indicates lower accuracy or more tokens. Annotations are rounded to whole percentages.

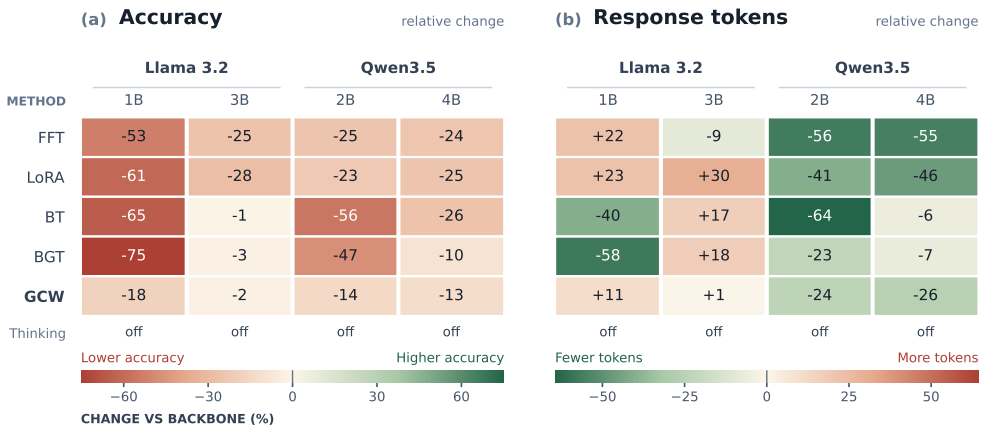


Figure 12: **KOR-Bench after sequential adaptation.** Percentage changes in accuracy and mean response length relative to the backbone, with baselines matched by backbone and task group (Table 7). Accuracy averages five tasks. Mean response length pools all 1,250 responses. All evaluations use thinking disabled. The panels use independent linear colour scales centred on zero. Forest green indicates higher accuracy or fewer tokens; terracotta indicates lower accuracy or more tokens. Annotations are rounded to whole percentages.

Table 7: **KOR-Bench zero-shot accuracy (%) after sequential mathematical and Reasoning Gym adaptation.** Backbone baselines are reproduced from Table 4 with thinking disabled. All runs use chat formatting with thinking disabled and a 2,048-new-token cap. Each category contains 250 questions, so Avg. is both the unweighted category mean and accuracy over all 1,250 questions. Mean response tokens equal total generated response tokens divided by 1,250; totals are in millions. Retained is the percentage of the matched pretrained backbone's correct answers that remain correct; Recovered is the percentage of its incorrect answers that become correct, pairing all 1,250 questions within the same thinking mode. Trunc. reports the percentage of responses stopped by the generation cap, pooled over the same tasks as token statistics.

	KOR-Bench accuracy						Backbone overlap (%)		Generation		
Method	Operation	Logic	Cipher	Puzzle	Ctr-factual	**Avg.**	Retained	Recovered	Mean/resp.	Total (M)	Trunc. (%)
Llama 3.2-1B											
Backbone	30.00	3.20	2.00	1.60	50.40	**17.44**	100.00	0.00	375.2	0.469	8.88
FFT	18.00	1.60	2.40	1.20	17.60	**8.16**	28.44	3.88	459.4	0.574	18.16
LoRA	15.20	1.20	2.00	0.40	15.20	**6.80**	25.23	2.91	460.0	0.575	18.48
BT	6.00	5.60	1.20	0.00	17.60	**6.08**	20.64	3.00	224.0	0.280	9.20
BGT	2.00	3.20	2.00	0.00	14.40	**4.32**	14.22	2.23	158.3	0.198	6.32
GCW	28.40	2.40	0.80	0.80	39.20	**14.32**	61.01	4.46	416.7	0.521	13.20
Llama 3.2-3B											
Backbone	64.40	12.00	6.40	2.80	80.80	**33.28**	100.00	0.00	407.1	0.509	9.44
FFT	43.60	3.20	3.60	0.80	72.80	**24.80**	66.11	4.20	371.4	0.464	10.00
LoRA	35.60	6.40	0.80	1.60	76.00	**24.08**	63.70	4.32	529.6	0.662	18.32
BT	63.60	14.40	4.40	0.40	81.60	**32.88**	87.26	5.76	476.6	0.596	14.32
BGT	60.40	14.00	3.60	2.80	80.80	**32.32**	84.38	6.35	478.4	0.598	14.40
GCW	62.00	12.40	4.80	2.00	81.20	**32.48**	82.93	7.31	409.6	0.512	10.16
Qwen3.5-2B											
Backbone	85.20	33.20	16.00	7.60	89.60	**46.32**	100.00	0.00	813.7	1.017	25.76
FFT	55.60	28.80	7.60	1.20	80.80	**34.80**	65.46	8.35	354.7	0.443	9.68
LoRA	58.80	28.40	5.20	2.40	84.40	**35.84**	67.70	8.35	484.1	0.605	14.08
BT	39.60	21.60	4.40	4.00	32.40	**20.40**	36.44	6.56	289.7	0.362	9.60
BGT	43.60	19.60	4.80	1.20	52.80	**24.40**	47.67	4.32	626.3	0.783	25.68
GCW	66.40	27.60	11.20	4.00	89.60	**39.76**	76.17	8.35	620.0	0.775	15.52
Qwen3.5-4B											
Backbone	87.20	53.60	52.00	14.80	96.40	**60.80**	100.00	0.00	840.8	1.051	25.68
FFT	70.80	43.20	22.00	4.80	90.80	**46.32**	71.18	7.76	374.6	0.468	7.44
LoRA	73.60	42.00	16.40	5.20	90.00	**45.44**	68.95	8.98	451.5	0.564	10.72
BT	63.20	30.40	42.80	11.60	78.40	**45.28**	70.00	6.94	788.7	0.986	22.88
BGT	82.00	46.40	42.80	10.40	92.80	**54.88**	85.66	7.14	781.1	0.976	25.28
GCW	81.20	46.00	37.60	5.20	95.60	**53.12**	82.76	7.14	623.9	0.780	14.80

Table 8: **Mathematical and LogiQA benchmark accuracy (%) after sequential mathematical and Reasoning Gym adaptation.** Each method is initialised from its corresponding mathematics-adapted checkpoint and then trained on Reasoning Gym. Backbone baselines are reproduced from Table 3 with thinking disabled; averages and token statistics exclude TheoremQA. Generation tasks use chat formatting with thinking disabled and a 2,048-new-token cap, and are scored by exact match on the extracted answer. LogiQA and Gaokao use length-normalised option likelihood. Avg. is the unweighted mean over the five displayed benchmarks. Mean response tokens equal total generated response tokens divided by the 2,938 responses across GSM8K, SVAMP, and GSM8K-Hard; totals are in millions. TheoremQA is excluded from both averages and token statistics. Trunc. reports the percentage of responses stopped by the generation cap, pooled over the same tasks as token statistics.

	Benchmark accuracy						Generation		
Method	GSM8K	SVAMP	GSM8K-Hard	LogiQA	Gaokao	**Avg.**	Mean/resp.	Total (M)	Trunc. (%)
Llama 3.2-1B									
Backbone	43.37	63.33	12.81	30.41	27.07	**35.40**	178.1	0.523	1.12
FFT	47.46	63.00	12.21	29.65	27.35	**35.93**	200.8	0.590	2.86
LoRA	37.23	58.67	10.84	29.95	29.34	**33.21**	267.9	0.787	6.84
BT	40.18	59.67	11.60	31.49	26.78	**33.94**	193.0	0.567	3.17
BGT	40.94	61.67	10.99	32.72	26.50	**34.56**	139.6	0.410	0.88
GCW	37.53	62.33	10.46	31.03	29.06	**34.08**	147.1	0.432	1.02
Llama 3.2-3B									
Backbone	69.90	78.00	24.26	29.34	31.05	**46.51**	196.6	0.578	1.33
FFT	71.49	83.00	23.81	30.11	29.06	**47.49**	184.5	0.542	1.94
LoRA	66.94	79.33	22.44	31.80	29.63	**46.03**	212.9	0.625	4.12
BT	68.16	80.00	23.35	28.26	28.49	**45.65**	220.5	0.648	2.89
BGT	68.31	80.33	24.41	28.88	29.06	**46.20**	197.4	0.580	1.80
GCW	67.70	78.33	22.37	29.03	27.92	**45.07**	206.3	0.606	1.94
Qwen3.5-2B									
Backbone	66.94	84.33	38.21	31.03	33.62	**50.83**	458.4	1.347	4.66
FFT	72.86	84.33	37.45	29.34	25.93	**49.98**	210.2	0.618	2.28
LoRA	69.14	81.33	39.35	31.34	28.77	**49.99**	205.8	0.605	2.76
BT	58.76	75.00	30.48	30.88	31.62	**45.35**	359.2	1.055	4.83
BGT	29.34	66.00	11.60	31.80	29.63	**33.67**	1063.8	3.125	40.06
GCW	54.06	72.33	27.67	30.11	30.20	**42.87**	401.2	1.179	9.87
Qwen3.5-4B									
Backbone	80.21	88.67	51.86	35.33	34.19	**58.05**	489.2	1.437	3.78
FFT	83.47	89.00	49.73	34.72	40.46	**59.48**	207.2	0.609	1.57
LoRA	82.56	88.00	48.60	33.33	38.46	**58.19**	174.8	0.514	1.06
BT	73.69	84.67	45.03	33.79	31.62	**53.76**	384.4	1.129	3.30
BGT	71.19	82.33	42.84	33.03	32.19	**52.32**	300.0	0.881	1.70
GCW	86.13	89.33	54.66	33.33	36.47	**59.98**	175.6	0.516	0.41

D Extended Analysis

Accuracy does not distinguish preserved successes from newly solved questions, while higher accuracy can increase both retention and recovery. We therefore compare the locations of successful answers on KOR-Bench, pairing each adapted model with its pretrained backbone on the same 1,250 questions and thinking setting (Tables 4, 5, and 7).

Measures. Let n_{uv} count questions with backbone correctness u and adapted-model correctness v, with $u, v \in \{0, 1\}$. The tables report retention $n_{11}/(n_{11} + n_{10})$ and recovery $n_{01}/(n_{01} + n_{00})$. To describe the composition of the adapted model's successes, we instead measure

$$s_{\text{shared}} = \frac{n_{11}}{n_{11} + n_{01}}, \qquad s_{\text{new}} = \frac{n_{01}}{n_{11} + n_{01}}. \tag{8}$$

These complementary fractions distinguish successes shared with the backbone from successes on questions it answered incorrectly (Figure 13); a larger new-success share need not imply higher accuracy.

To account also for category-level performance, we compare observed overlap with the expectation under uniform permutation of adapted correctness labels within each category, keeping the backbone outcomes fixed. For category c, let N_c, B_c, and F_c denote question, backbone-success, and adapted-success counts. Expected and maximum feasible shared successes are E and U, giving

$$E = \sum_c \frac{B_c F_c}{N_c}, \qquad U = \sum_c \min(B_c, F_c), \qquad S = \frac{n_{11} - E}{U - E}. \tag{9}$$

Thus $S = 0$ denotes the within-category random expectation and $S = 1$ the largest overlap compatible with the observed category accuracies, rather than perfect retention. All reported conditions have $U > E$.

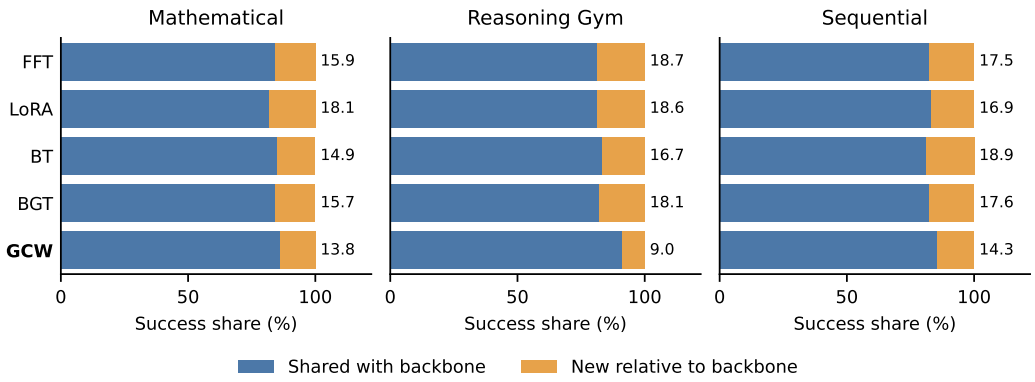


Figure 13: **Composition of adapted-model successes.** Bars show shared and new successes (Equation 8); numbers give the new-success percentage. Fractions are averaged equally across the eight reported backbone–thinking conditions after mathematical adaptation and four after each other setting.

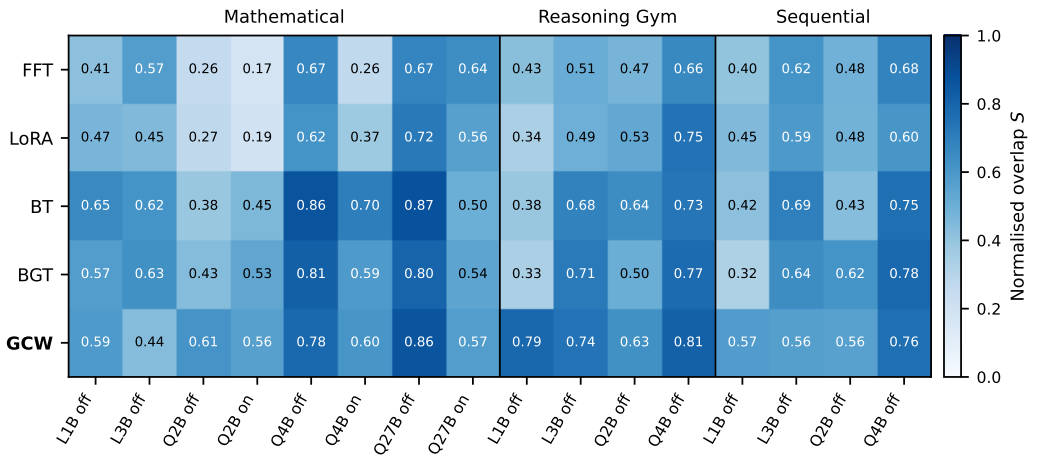


Figure 14: **Overlap conditional on category accuracies.** Cells report S (Equation 9) for each backbone and thinking setting. L/Q denote Llama 3.2/Qwen3.5; off/on denote thinking disabled/enabled. Higher values indicate greater overlap relative to the category-matched reference and feasible maximum.

Findings. GCW has the lowest mean new-success share in each setting (13.78%, 8.97%, and 14.32%), but its advantage in normalised overlap depends on the training data (Figure 14). After mathematical adaptation, mean S is similar for BT (0.630), GCW (0.626), and BGT (0.614). After Reasoning Gym, GCW leads more clearly (0.741 versus 0.611 for BT); its sequential-adaptation lead is smaller (0.614 versus 0.591 for BGT). Greater concentration on backbone successes is therefore most evident after direct reasoning adaptation. After mathematical adaptation, GCW's retention advantage over the mean of FFT and LoRA narrows with backbone size: for Qwen with thinking disabled, it falls from 49.7 to 33.6 to 15.3 percentage points at 2B, 4B, and 27B, respectively, as FFT/LoRA retention improves. This suggests that FFT and LoRA's capacity for retention scales with backbone size, while GCW maintains high retention across all backbones.