GLOBAL LATENT WORKSPACE: SHAPING THE WORKSPACE FOR MULTI-AGENT LATENT COMMUNICATION VIA REINFORCED ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent communication allows multi-agent LLMs to exchange internal representations without repeatedly decoding and re-encoding text. However, relaying intermediate states without selective filtering can introduce redundancy and tie communication cost to local reasoning length. We propose the Global Latent Workspace (GLW), a bounded communication interface that learns which information to share for downstream reasoning. Independent experts generate local latent states, and a lightweight controller selectively integrates them into a fixed-size workspace. After workspace initialization, the controller skips or blends each remaining candidate. A frozen final agent reads the resulting workspace to produce the answer. This decouples the amount of local computation from shared-context size, making communication a learnable component of multi-agent reasoning. We train only the controller using reinforcement learning with attention-derived feedback and writer-level counterfactual rewards, while keeping all agent backbones frozen. Experiments on six benchmarks with two LLM backbones show strong reasoning performance relative to three representative baselines. GLW demonstrates strong reasoning performance across all six tasks with the 8B backbone, while reducing shared-context length relative to latent relay and generated output relative to text-based collaboration. A fixed-budget state-selection study examines learned admission; reward ablations assess the contributions of frozen-judger feedback.

1 INTRODUCTION

Large language models increasingly solve complex problems through multi-step reasoning, yet a single reasoning trajectory provides only one path through the problem. Multi-agent systems extend this process by allowing multiple agents to develop, exchange, and integrate intermediate reasoning, making communication a central component of collaborative reasoning (Zou et al., 2025). The effectiveness of such systems therefore depends not only on what individual agents compute, but also on how their intermediate information is communicated. Figure 1 shows how our Global Latent Workspace combines high reasoning accuracy with efficient communication.

Text-based multi-agent systems (TextMAS) exchange intermediate reasoning as readable token sequences. Each exchange requires the sender to decode its representations into text and the receiver to encode that text, adding generation and processing costs as interactions accumulate.

Latent communication reduces repeated conversion between representations and text. Thought Communication formalizes shared and private latent thoughts (Zheng et al., 2025), while LatentMAS enables training-free reasoning and collaboration through hidden states and KV-cache transfer (Zou et al., 2025). These approaches transmit intermediate reasoning without generating a textual message at each exchange.

Removing text serialization, however, does not resolve the problem of deciding what to communicate. Relaying a sequence of hidden states ties shared context to the amount of local computation that produced it. As more experts or latent steps are added, the receiver inherits more intermediate material, even though the task may require only a few complementary pieces of evidence. A state can be useful to the process that generated it without being useful to the final reader. Even within

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

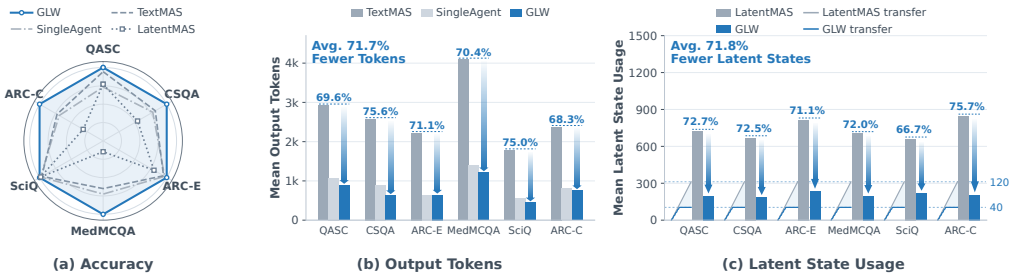


Figure 1: **Accuracy, output tokens, and latent state usage with Qwen3-8B.** (a) Accuracy relative to GLW; the radial axis is truncated. (b) Mean output tokens, including thinking, and reductions from TextMAS. (c) Mean latent state usage by the final judger and reductions from LatentMAS. This count includes the latent states transferred to the judger and the latent representations of its input, including the task question and system prompt, used to generate the final answer. The total therefore varies with the question and the judger's system prompt. Average reductions are unweighted across displayed tasks; transfer curves are schematic. CSQA, ARC-E, and ARC-C denote CommonsenseQA, ARC-Easy, and ARC-Challenge. All three panels use the same configuration for each task, as reported in Table 1.

an LLM, information can be decodable from activations yet unused for explicit factual retrieval (Bogdan & Lindsey, 2026). For latent collaboration, this motivates distinguishing *local availability* from *shared utility*: which locally computed states should become shared context? For example, a question may require complementary facts developed by different experts. Shortening local reasoning can remove useful exploration, while a fixed prefix can miss later evidence. Novelty measures difference from existing content, not usefulness to the reader. We therefore learn admission from downstream outcomes under a fixed communication budget.

Global Workspace Theory provides a conceptual basis for our GLW approach. In the theory, many specialized processes operate outside conscious access, while selected information enters a capacity-limited workspace that makes it available to other processes (Baars, 2005; Dehaene & Naccache, 2001; Mashour et al., 2020). Its architectural lesson is that extensive local processing can coexist with selective shared access. The consciousness prior formalizes a related bottleneck for selecting a few high-level concepts to condition further processing (Bengio, 2017). Later proposals develop global latent workspaces and competition among neural modules for shared capacity (VanRullen & Kanai, 2021; Goyal et al., 2022). This perspective suggests treating communication as controlled admission to a common state, rather than assuming that every intermediate representation should become shared context.

Recent evidence inside language models sharpens this motivation. Gurnee et al. (2026) identify a restricted set of verbalizable representations that supports internal reasoning, directed modulation, and flexible downstream use, alongside a much larger volume of automatic processing. Their study concerns an emergent workspace within a model. Our interest is an engineered interface between models: can a comparable separation between local processing and shared access improve collaboration? We use the workspace account as a functional design motivation, without attributing conscious experience to agents. The resulting question is: *can a controller learn what frozen agents should share under a fixed budget while retaining the benefits of local reasoning?*

Our approach. We propose the **Global Latent Workspace** (GLW), a communication interface that separates candidate generation from shared access. Frozen experts produce local latent states, and a lightweight controller selects and integrates this latent information into a workspace that is passed to a frozen final agent responsible for generating the answer. This approach lets us test whether learning what to share improves reasoning without increasing shared capacity. In the evaluated hierarchical deployment, the workspace is built after independent experts finish; a frozen final agent, termed the judger, reads it to answer the question (Figure 3). Only the communication controller is trained; all agent backbones remain frozen.

The controller learns skip/write decisions through reinforcement learning; writes apply content-addressed erase/add updates. Training combines outcome contrasts from writer-suppressed rollouts

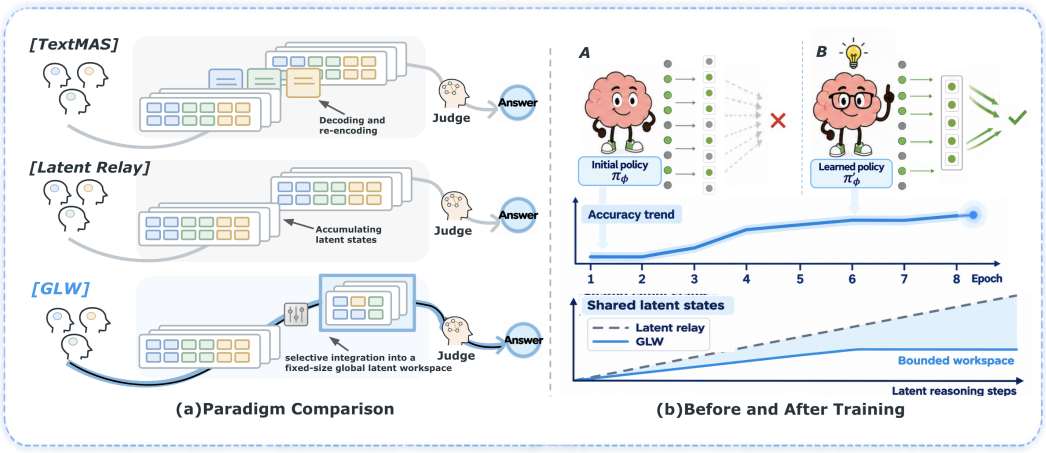


Figure 2: **Communication paradigms and performance trends before and after GLW controller training.** (a) TextMAS exchanges generated messages through decoding and re-encoding; latent relay accumulates intermediate states; GLW selectively integrates information into a capacity-limited workspace. (b) Controller training changes the sharing policy from π_ϕ to π'_ϕ while keeping agent backbones frozen. The illustration shows trends in task accuracy before and after training, while the number of latent states transferred between agents remains bounded by the fixed workspace capacity, in contrast to continued accumulation in latent relay.

with the frozen judger's attention to workspace slots. These signals supervise communication without target workspace states or gradients through the agent backbones.

We evaluate GLW on task accuracy, communication cost, and controller learning. Fixed-budget selection controls assess learned admission, while reward ablations examine frozen-judger feedback. These studies connect reasoning performance with shared-context size and generated-output length.

Our contributions are threefold: ❶ **A bounded interface for latent collaboration.** We formulate communication as learned access to shared capacity and instantiate GLW with independent candidate generation, selective workspace updates, and frozen downstream readout. The interface decouples local reasoning from shared context size. ❷ **Communication learning from frozen-agent feedback.** We develop a controller-training procedure that combines writer-level outcome contrasts with attention-derived write credit. This trains communication decisions without updating the reasoning backbones or requiring target workspace states. ❸ **Experimental evaluation.** We evaluate task accuracy, communication cost, and controller learning through task comparisons and controlled ablations. In a fixed-budget state-selection study on QASC-8B, training improves accuracy by 2.67 percentage points over the untrained controller, averaged across three runs. Complementary analyses characterize the accuracy–communication tradeoff and the contribution of frozen-judger feedback.

2 Related Work

2.1 Latent Multi-Agent Collaboration

Language-based agents communicate through messages or aggregate answers. ReAct interleaves reasoning with actions (Yao et al., 2022), and Agent Forest aggregates independently sampled answers (Li et al., 2024a). Thought Communication studies shared and private latent thoughts and their identifiability (Zheng et al., 2025). LatentMAS uses training-free autoregressive latent reasoning and layer-wise KV-cache transfer (Zou et al., 2025). RecursiveMAS learns RecursiveLink modules for recursive latent collaboration among heterogeneous agents (Zou et al., 2026). Mixture of Thoughts (MoT) learns expert routing and projection/cross-attention modules for hidden-state exchange (Fein-Ashley et al., 2025). Both RecursiveMAS and MoT keep expert backbones frozen. GLW focuses on learning which independently generated states to admit and integrate into a fixed-capacity workspace, using outcome and attention-derived feedback. The final reader re-encodes the workspace vectors (Figure 2).

2.2 Global Workspaces and Learnable Memory

Global Workspace Theory associates shared access with the integration and availability of information from specialized processes (Baars, 2005; Dehaene & Naccache, 2001; Mashour et al., 2020). VanRullen & Kanai (2021) propose a global latent workspace for translation between specialist representations, a direction also pursued by the GLoW project (European Commission, 2023). Goyal et al. (2022) train neural modules that compete for a bandwidth-limited shared workspace. GLW implements selective shared access for frozen LLM agents, integrating independently generated candidate states into a bounded workspace for downstream reasoning. Within language models, Gurnee et al. (2026) use the Jacobian lens and causal interventions to study representations with workspace-like functions. Their emergent within-model workspace motivates our distinction between local processing and shared access; GLW neither extracts their J-space nor assumes that its slots are independently verbalizable concepts. Related mechanistic evidence shows that implicit intermediate results affect compositional reasoning (Li et al., 2024b), while decodable information may remain unused in factual retrieval (Bogdan & Lindsey, 2026). These findings motivate examining downstream use, without establishing cross-agent transfer.

External memory supplies related mechanisms. Neural Turing Machines introduce differentiable addressing and erase/add updates (Graves et al., 2014), while Neural Episodic Control uses stored state–value associations (Pritzel et al., 2017). GDWM allocates consolidation updates according to contextual utility under limited computation (Mei et al., 2026). LatentMem constructs agent-specific latent memories from retrieved experience (Fu et al., 2026); MemGen learns memory invocation and the generation of latent memory tokens (Zhang et al., 2025). These approaches illustrate how learnable memory components can augment frozen core models. GLW builds a fresh workspace from current expert states for each problem, focusing on within-problem communication rather than persistent experience retrieval.

2.3 Reinforcement Learning for Memory and Communication

Memory-R1 learns textual memory operations and answer-side memory use from downstream rewards (Yan et al., 2025). Memory-R2 addresses long-horizon credit assignment using global outcomes and local rerollouts from shared intermediate memory states (Yan et al., 2026). Both motivate learning information management through its eventual usefulness. GLW instead acts on continuous states within a problem and trains an external controller while keeping every agent backbone frozen. Its writer-suppressed outcome contrast does not implement Memory-R2's long-horizon local credit scheme. Policy gradients and learned visual attention provide foundations for stochastic selection (Williams, 1992; Mnih et al., 2014); PPO and group-relative optimization offer practical sampled-outcome objectives (Schulman et al., 2017; DeepSeek-AI, 2025). GLW obtains attention from a frozen judger and uses it to shape rewards for a separate controller, supplementing writer-level outcome feedback.

3 Method

GLW combines independent candidate generation, bounded workspace updates, and controller learning from frozen-judger feedback (Figure 3).

3.1 Independent Candidates and Bounded Readout

For question x, three experts share frozen backbone weights but receive math, science, and code role prompts. At each latent step, an expert maps its last-position hidden state through a fixed alignment and norm-scaling function, then feeds the resulting vector back as the next input embedding. Using the same alignment as LatentMAS, each expert stores these input vectors as $L = 40$ candidates $H_i = (h_{i,1}, \ldots, h_{i,L})$, $h_{i,\ell} \in \mathbb{R}^d$ (Appendix F.1), where $i \in \{1, 2, 3\}$ indexes experts, ℓ indexes latent steps, and d is the backbone embedding dimension. We choose 40 states per expert based on the latent-step study in LatentMAS (Zou et al., 2025), which reports strong performance around 40–80 steps; 40 is an effective choice within this range that keeps local computation modest. The 120 independently generated candidates are interleaved by latent step and expert. The first 40 initialize $M_0 \in \mathbb{R}^{40 \times d}$ with 14/13/13 states from the three roles; the controller processes the remaining 80.

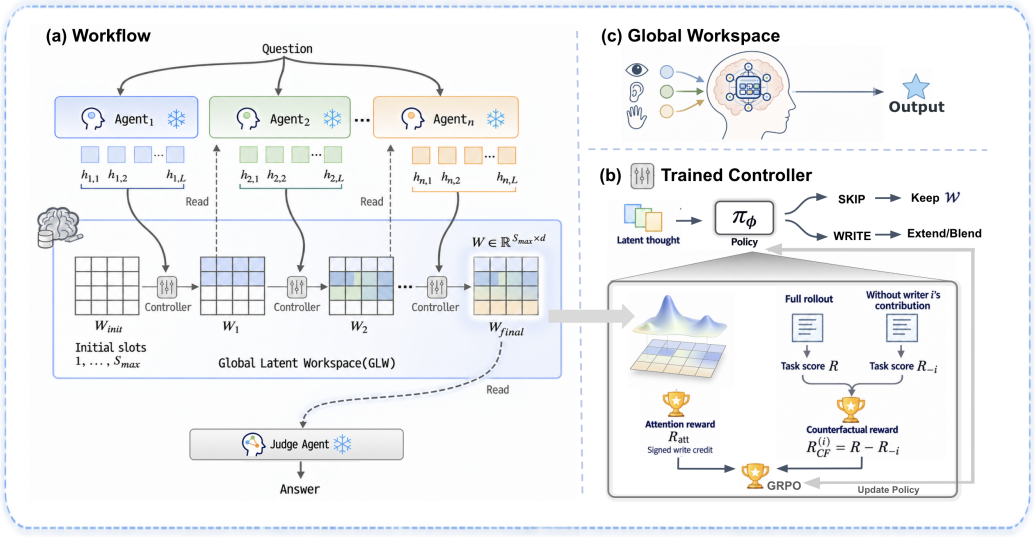


Figure 3: **Global Latent Workspace: architecture and controller training.** (a) Agents produce local latent states for selective integration into a workspace of at most $S_{\max}$ slots; a frozen judge reads the final workspace. Dashed read paths depict workspace reuse in the general framework; the hierarchical deployment uses independent experts and a single final readout. (b) The controller chooses SKIP or WRITE; in the fixed-40 deployment, each write blends into existing slots. With unused capacity, novel writes can instead extend the workspace. Attention-derived credit and writer-suppressed outcome contrasts $R_{\text{CF}}^{(i)} = R - R_{-i}$ guide GRPO training. Only the controller is trained; snowflakes denote frozen backbones. (c) Global Workspace Theory motivates selective integration and shared access.

A frozen final agent, the judger, reads the workspace as continuous embeddings inserted after the user-turn delimiter in its own prompt and generates the answer. This constructs fresh judger KV rather than transferring expert KV. Algorithm 1 specifies the rollout and readout.

3.2 Write Selection and Workspace Updates

Let $M_t \in \mathbb{R}^{S_t \times d}$ be the workspace at update t, with S_t occupied slots and maximum capacity $S_{\max}$; $M_t[j]$ denotes slot j. For candidate $h = h_{i,\ell}$, the controller computes content similarities and addressing weights:

$$c_j = \frac{(W_q h)^\top M_t[j]}{\|W_q h\|_2 \|M_t[j]\|_2 + \epsilon_{\cos}}, \quad j = 1, \ldots, S_t, \tag{1}$$

$$w_j^c = \text{softmax}_j(\beta c_j), \qquad w_j = \frac{(w_j^c)^\gamma}{\sum_k (w_k^c)^\gamma}, \quad \beta > 0,\ \gamma \geq 1. \tag{2}$$

Here $W_q \in \mathbb{R}^{d \times d}$ is a learned candidate projection, $\epsilon_{\cos} > 0$ stabilizes normalization, and β and γ are learned, candidate-dependent strength and sharpening factors. The softmax and sums run over occupied slots; w^c and w are the initial and sharpened addressing distributions. The controller summarizes the workspace as $\hat{m} = \sum_j w_j^c M_t[j]$ and defines match quality $\mu = \max_j c_j$ and novelty $\nu = 1 - \mu$. Let q be the mean-pooled embeddings of the question tokens and f_q a learned projection. The binary policy, parameterized by controller weights ϕ, is

$$a_{i,\ell} \sim \pi_\phi(a \mid h, \hat{m}, \mu, \nu, f_q(q)), \qquad a \in \{\text{SKIP}, \text{WRITE}\}. \tag{3}$$

Training samples this action; inference takes its argmax. In the hierarchical deployment, all $S_{\max} = 40$ slots are initialized before policy decisions. A skip leaves the workspace unchanged, while a write applies a content-addressed erase/add update:

$$e = \sigma(W_e h), \quad a^+ = W_a h, \qquad M_{t+1}[j] = M_t[j] \odot (1 - w_j e) + w_j a^+. \tag{4}$$

Here $W_e, W_a \in \mathbb{R}^{d \times d}$ are learned projections; σ is the elementwise sigmoid, e is the erase vector, a^+ is the add vector, and $\odot$ denotes elementwise multiplication. Thus, each of the remaining 80

candidates is skipped or blended into the same 40 positions. Local computation remains 120 latent steps, while the shared readout stays fixed. For variable-occupancy analysis, a write instead appends h when $S_t < S_{\max}$ and $\nu > \theta_\nu$, where θ_ν is the novelty threshold; otherwise, Equation 4 applies.

3.3 Learning from Frozen-Judger Feedback

Writer-level outcome credit. Let τ denote a rollout and $R = R_{\text{task}}(\tau) \in \{0, 1\}$ its final correctness. A rollout using the same candidate bank and initialization suppresses writer i's subsequent policy-controlled writes and yields score R_{-i}, giving

$$R_{\text{CF}}^{(i)} = R - R_{-i}. \tag{5}$$

The contrast supplies writer-level outcome feedback. Each suppressed score is reused across the sampled rollout group (Algorithm 3).

Attention-derived write credit. After answer generation, an additional gradient-free pass re-encodes the same latent prefix and re-tokenized answer. We aggregate final-layer attention to workspace slots (Appendix G.5) and normalize to obtain $p_j^{\text{J}} \geq 0$ with $\sum_{j=1}^{S_T} p_j^{\text{J}} = 1$, where T is the final update index and S_T is final occupancy. For a blended write, stored addressing weights are aligned to the final workspace slots, with zero weight on slots added later; for an extension, credit is the final attention on the new slot:

$$\kappa_{i,\ell} = \begin{cases} p_{j_{\text{new}}}^{\text{J}}, & \text{extension,} \\ \sum_j w_{i,\ell,j} p_j^{\text{J}}, & \text{blend.} \end{cases} \tag{6}$$

Here j_{new} indexes the appended slot and $w_{i,\ell,j}$ is the stored addressing weight for candidate (i, ℓ) on final slot j. The update-controller reward is

$$r_{i,\ell} = \mathbf{1}[a_{i,\ell} = \text{WRITE}] \left[\max(0, R_{\text{CF}}^{(i)}) + \lambda_J \operatorname{sign}(R - \tfrac{1}{2}) \kappa_{i,\ell} \right]. \tag{7}$$

Here $\mathbf{1}[\cdot]$ is an indicator and $\lambda_J \geq 0$ controls the attention reward. Since R is binary, the sign term is $+1$ for correct answers and -1 otherwise: attended writes are rewarded on correct rollouts and penalized on incorrect ones. The judger remains frozen.

Controller optimization. For G trajectories of the same question, let $\mathcal{T}$ contain the included WRITE and SKIP transitions, excluding forced initialization. We jointly standardize their rewards to obtain advantages A_k, where k indexes transitions. At each saved observation $s_k = (h_k, M_{\text{pre},k}, q)$, with candidate h_k and pre-update workspace $M_{\text{pre},k}$, each policy recomputes its addressing and gate features. With $\rho_k = \pi_\phi(a_k \mid s_k) / \pi_{\phi_{\text{old}}}(a_k \mid s_k)$, the gate objective is

$$\begin{aligned} \mathcal{L}_{\text{GRPO}} = & -\mathbb{E}_{k \in \mathcal{T}}[\min(\rho_k A_k, \text{clip}(\rho_k, 1-\epsilon, 1+\epsilon) A_k)] \\ & + \beta_{\text{KL}} D_{\text{KL}}(\pi_\phi \| \pi_{\text{ref}}) - \eta \mathcal{H}(\pi_\phi). \end{aligned} \tag{8}$$

Here $\pi_{\phi_{\text{old}}}$ is the rollout-collection policy and π_{ref} the reference policy; ϵ is the ratio-clipping margin. D_{KL} and $\mathcal{H}$ denote KL divergence and policy entropy at saved observations, weighted by β_{KL} and η. The full objective is $\mathcal{L} = \mathcal{L}_{\text{GRPO}} + \mathcal{L}_{\text{content}} + \mathcal{L}_{\text{prior}}$. The continuous write parameters receive an auxiliary signal by comparing the current erase/add output U_k with $U_{k,\text{ref}}$ from a periodically synchronized reference controller. Both use the saved pre-update workspace:

$$\mathcal{L}_{\text{content}} = \frac{\lambda_{\text{write}}}{|\mathcal{T}|} \sum_{k \in \mathcal{T}} \mathbf{1}[a_k = \text{WRITE}] \ \text{sg}(A_k) \ \frac{\|U_k - \text{sg}(U_{k,\text{ref}})\|_F^2}{S_k d}. \tag{9}$$

Here λ_{write} weights the content loss, sg stops gradients, $\|\cdot\|_F$ is the Frobenius norm, and S_k is the stored occupancy. The numerator includes only writes; the denominator counts all included transitions. Positive advantages pull toward the reference, while negative advantages push away. The write prior is $\mathcal{L}_{\text{prior}} = \lambda_{\text{ws}} \mathbb{E}_{k \in \mathcal{T}}[(p_{\text{target}} - \pi_\phi(\text{WRITE} \mid s_k))_+^2]$, where $(u)_+ = \max(0, u)$, p_{target} is the target minimum write probability, and λ_{ws} weights this prior. The gate learns from Equation 8 and the prior; the erase/add heads learn from Equation 9, which also updates addressing parameters. Candidates and saved workspaces are detached, and reference outputs and advantages are fixed during each update. Gradients update only the controller, without passing through the experts or judger.

Table 1: **Accuracy and communication on six benchmarks.** Acc.: accuracy (%); Tokens: mean generated output tokens (including thinking); States: mean latent state positions used by the final judger, including transferred states and representations of its input question and system prompt. Δ: accuracy gain (percentage points) and latent-state reduction versus LatentMAS, or output-token reduction versus TextMAS. For each task, backbone, and method, all three metrics use the same configuration, evaluation questions, and seeds; GLW uses the same validation-selected checkpoint per seed and a fixed 40-slot workspace. Entries are means over seeds 42/43/44. Bold GLW entries are best or tied per model and metric; green marks improvements; dashes are inapplicable.

Tasks	Metrics	Qwen3-4B				Δ	Qwen3-8B				Δ
		SingleAgent	TextMAS	LatentMAS	GLW		SingleAgent	TextMAS	LatentMAS	GLW	
QASC	Acc.	83.83	84.33	82.17	**84.83**	↑ **2.67**	85.17	87.00	85.50	**87.50**	↑ **2.00**
	Tokens	1014.4	2972.8	646.1	**625.5**	↓ **79.0%**	1078	2923	707	890	↓ **69.6%**
	States	–	2729.2	725.5	**197.1**	↓ **72.8%**	–	3013.2	721.5	**197.1**	↓ **72.7%**
Commonsense QA	Acc.	80.17	80.83	78.50	77.17	↓ 1.33	81.67	82.00	79.67	**83.50**	↑ **3.83**
	Tokens	941.2	2666.5	566.9	**506.7**	↓ **81.0%**	898	2576	736	**628**	↓ **75.6%**
	States	–	2421.7	673.1	**184.0**	↓ **72.7%**	–	2607.4	669.1	**184.0**	↓ **72.5%**
ARC-Easy	Acc.	99.17	99.00	97.50	**99.17**	↑ **1.67**	99.67	99.50	98.00	**100.00**	↑ **2.00**
	Tokens	702.0	2193.7	466.4	**425.5**	↓ **80.6%**	636	2222	445	642	↓ **71.1%**
	States	–	1956.8	730.9	**198.4**	↓ **72.9%**	–	2000	813.6	**235.4**	↓ **71.1%**
MedMCQA	Acc.	57.33	59.83	55.33	**60.50**	↑ **5.17**	64.67	64.17	60.83	**66.50**	↑ **5.67**
	Tokens	1539.6	4790.8	944.1	1057.9	↓ **77.9%**	1389.2	4092.0	973.8	1211.1	↓ **70.4%**
	States	–	4413.6	722.6	**196.4**	↓ **72.8%**	–	4112.5	702.6	**196.4**	↓ **72.0%**
SciQ	Acc.	95.50	96.67	95.67	96.50	↑ **0.83**	96.67	96.50	96.67	**96.83**	↑ **0.17**
	Tokens	592.0	1859.7	426.2	**382.2**	↓ **79.4%**	550	1781	530	**445**	↓ **75.0%**
	States	–	1655.8	662.8	**181.4**	↓ **72.6%**	–	1766.4	658.8	**219.4**	↓ **66.7%**
ARC-Challenge	Acc.	92.00	91.17	86.00	**92.50**	↑ **6.50**	94.17	94.50	90.50	**97.17**	↑ **6.67**
	Tokens	850.5	2014.6	503.2	573.6	↓ **71.5%**	812	2376	557	754	↓ **68.3%**
	States	–	1910	845.0	**206.3**	↓ **75.6%**	–	2150	845.0	**205.6**	↓ **75.7%**

4 EXPERIMENTS

We study three questions: **(RQ1)** Can a bounded latent workspace improve multi-agent reasoning? **(RQ2)** How much shared context and generated text does it require? **(RQ3)** What do learned selection and frozen-judger feedback contribute?

4.1 EXPERIMENTAL SETUP

Benchmarks and models. We evaluate QASC (Khot et al., 2020), CommonsenseQA (Talmor et al., 2019), ARC-Easy and ARC-Challenge (Clark et al., 2018), MedMCQA (Pal et al., 2022), and SciQ (Welbl et al., 2017), using frozen Qwen3-4B and Qwen3-8B (Yang et al., 2025). Controllers are trained on ARC-Challenge and transferred to the other benchmarks. Validation and final-evaluation slices are listed in Table A4. QASC, CommonsenseQA, and MedMCQA are evaluated on labeled subsets of their official validation splits, rather than their official hidden test splits.

Baselines. SingleAgent answers directly, TextMAS exchanges generated messages, and LatentMAS transfers expert KV. GLW and LatentMAS share expert prompts and a local budget of 40 latent steps per expert, totaling 120 states. GLW exposes 40 workspace positions to the judger. Every main-table GLW result uses the update controller in Section 3: initialize 40 slots, then skip or blend each of the remaining 80 candidates (Algorithm 1). Training uses the grouped policy objective for gating and the content auxiliary loss for erase/add and addressing, with frozen-judger outcome and attention rewards. Only controller parameters are optimized.

Protocol and reporting. Thinking-enabled evaluations use temperature 0.6, top-p 0.95, and 3072–4096 output-token caps, fixed within each configuration. Scoring uses the final boxed answer, falling back to the last explicit answer. Results average seeds 42/43/44 unless specified otherwise. For each main-table entry, accuracy and communication costs use the same questions, decoding settings, and seeds; GLW also uses the same validation-selected checkpoint per seed for all three metrics. Appendix E details the protocol.

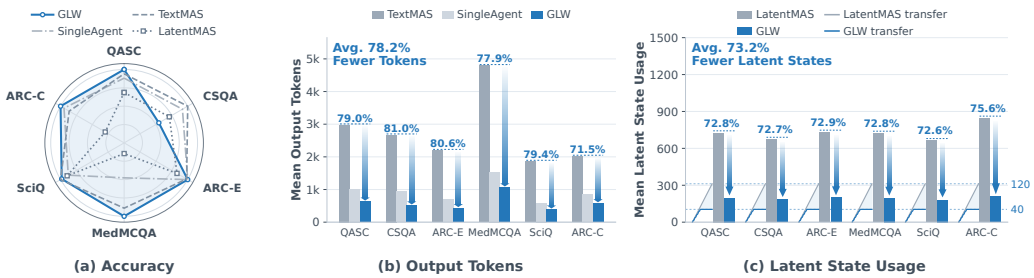


Figure 4: **Accuracy, output tokens, and judger context with Qwen3-4B.** (a) Accuracy normalized by the highest accuracy among compared methods on each task; the radial axis is truncated. (b) Mean output tokens, including thinking, and reductions from TextMAS. (c) Mean final-judger context length (input plus latent positions), labeled "latent state usage," and reductions from LatentMAS. Transfer curves are schematic. Average reductions are unweighted across the six tasks. All three panels use the same configuration for each task and method, as reported in Table 1.

4.2 Main Results (RQ1)

Table 1 and Figures 1(a) and 4(a) compare accuracy and communication costs across six tasks and both backbones. The accuracy results show two main findings: ❶ **GLW demonstrates strong reasoning performance across all six 8B tasks.** Gains over LatentMAS range from 0.17 to 6.67 percentage points. QASC and CommonsenseQA reach 87.50% and 83.50%, improving by 2.00 and 3.83 points. On MedMCQA and the official ARC-Challenge test set, GLW reaches 66.50% and 97.17%, respectively, improving over LatentMAS by 5.67 and 6.67 points. ❷ **The gains extend to the 4B backbone.** GLW exceeds LatentMAS on five of six 4B tasks and is best or tied on four. The largest gain is on ARC-Challenge (+6.50 points), followed by MedMCQA (+5.17 points). QASC and ARC-Easy also improve over LatentMAS. These are three-seed mean comparisons on fixed evaluation questions. RQ3 provides paired seed-level gains for the controlled learning and attention studies.

4.3 Communication Cost (RQ2)

We report two communication metrics: generated output tokens, including thinking and all emitting agents, and final-judger context length, including its input and shared latent positions (Table 1). Using these metrics, we summarize three main findings: ❶ **Extensive local reasoning leads to a compact shared readout.** With the same expert prompts and 120 local latent states as LatentMAS, GLW integrates the candidates into 40 workspace positions for the final judger and omits expert prompt KV. Across the six 8B tasks, reported judger contexts are 66.7–75.7% shorter than LatentMAS's; QASC decreases from 721.5 to 197.1 positions, including the judger's own input. ❷ **Latent sharing reduces generated text relative to TextMAS.** Reductions range from 68.3% to 75.6% on 8B: QASC uses 890 tokens versus TextMAS's 2923, though LatentMAS uses fewer at 707. ❸ **Communication reductions also hold for the smaller backbone.** Across the six 4B tasks, unweighted mean reductions are 78.2% in output tokens relative to TextMAS and 73.2% in judger-context length relative to LatentMAS (Figure 4(b,c)).

4.4 GLW Component Analysis (RQ3)

Effect of Learned State Selection. We use complementary controls to examine learned admission and the feedback that trains workspace updates. The QASC state-selection control isolates admission at a fixed 40-position budget; the reward ablations below test the update controller used by the main method. Selection training uses frozen-judger correct-option likelihood and correctness (Appendix B.1). We compare trained admission with its untrained initialization, fixed-first-40, random-40, and Zero-vector Workspace. The comparisons hold the frozen backbones, evaluation questions, workspace length, and decoding settings fixed, evaluating the same 200 questions across seeds 42/43/44. On QASC-8B, GLW reaches 87.50%, compared with 84.83% for the untrained selector, 84.83% for fixed-first-40, 85.00% for random-40, and 85.83% for the Zero-vector Workspace. The gain over both the untrained selector and fixed-first-40 is 2.67 percentage points. The paired

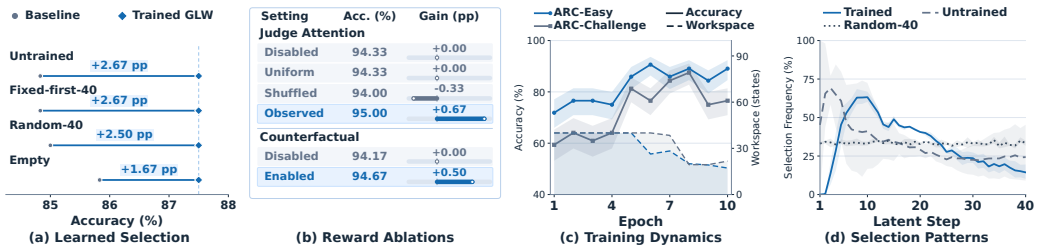


Figure 5: **GLW component analysis.** (a) QASC-8B admission controls at 40 positions; "Empty" uses zero vectors. Markers show mean accuracy (Table A1). (b) ARC-Challenge attention and counterfactual reward controls. Gains are percentage-point differences from each disabled-reward condition (Table A2). (c) Changes in validation accuracy (solid) and the mean number of occupied workspace slots (dashed) as training epochs increase, with thinking disabled and one seed. (d) Frequency with which each of an agent's 40 local latent states is selected into the global latent workspace on QASC under trained, untrained, and random selection. Training shifts the frequently selected positions relative to the untrained and random configurations. Frequencies are averaged across three experts and three seeds; shading shows the seed range.

gains over initialization are 2.00, 3.50, and 2.50 points for seeds 42/43/44, respectively, supporting learned admission across all three runs. Figure 5(a) and Table A1 report the controls.

Effect of Frozen-Judger Feedback. On ARC-Challenge-8B, mean accuracy is 95.00% with observed attention, 94.33% with uniform attention, 94.00% with shuffled attention, and 94.33% with the attention reward disabled (Table A2). Observed attention gains 0.67 percentage points over disabling the reward, averaged over seeds 42/43/44 on the same 200 questions. Retaining counterfactual reward improves accuracy from 94.17% to 94.67% (Figure 5(b)). Other settings are held fixed within each reward-on/off pair. Observed-versus-disabled gains are 0.00, 0.50, and 1.50 points across the three seeds. These small reward gains are descriptive: repeated runs use the same 200 questions, and seed variation does not quantify question-sampling uncertainty. We do not infer statistical significance from the reported means. Uniform and shuffled controls test whether slot-specific credit helps beyond the presence of an attention reward.

Workspace and Selection Behavior. During variable-occupancy training on ARC-Easy/Challenge (Qwen3-8B, thinking disabled, one seed), later epochs generally show higher validation accuracy and lower occupancy (Figure 5(c)). At fixed capacity, the QASC selector favors early-to-middle positions, whereas random selection is approximately uniform (Figure 5(d)). The comparisons test distinct parts of the proposed mechanism. Matching expert prompts and local latent steps controls the candidate-generation budget in the main comparison. Within the admission study, the untrained controller tests the effect of optimization; fixed-first-40 and random-40 test candidate choice at equal capacity; and zero vectors test whether expert content helps beyond allocating a prefix. The reward controls then intervene on the feedback used to train workspace updates. Thus, the main results evaluate the deployed system, while the controls support learned admission and reader-derived credit. They do not separately identify the contribution of erase/add blending, which would require a matched replacement-only or frozen-write-head control.

5 Conclusion

GLW learns shared access through a bounded workspace and frozen-judger feedback. Main experiments evaluate skip/blend updates, admission controls test learning at fixed capacity, and reward ablations examine the training signals. Together, they support learning what agents share while retaining independent local reasoning and a compact downstream readout. The fixed-budget controls associate improved accuracy with learned admission, while the reward comparisons measure the incremental effect of frozen-judger feedback. This links the interface, its optimization signals, and the observed behavior under matched experimental conditions. Transfer to broader settings remains a direction for future work.

AI USE STATEMENT

We used generative AI tools to assist with implementing the method, interpreting experimental results, and analyzing qualitative case studies. AI-assisted review helped check consistency between algorithm descriptions and experimental reporting. We also used these tools to draft and edit text, identify and check relevant literature, write code for figures, and format the manuscript in LaTeX. Before submission, the authors reviewed and checked all AI-assisted code and manuscript content, verified numerical reporting against experimental records, and checked citations against the original sources. We take responsibility for the final content of this work, including its text, claims, code, and figures.

ETHICS STATEMENT

This work studies communication among frozen language-model agents using existing question-answering benchmarks. Such models and benchmarks may contain biases, and improved task accuracy does not establish fairness or reliability beyond the evaluated settings. In particular, MedMCQA measures medical exam performance, not clinical safety or suitability for patient care. Latent states are less directly inspectable than text, and selective sharing does not guarantee that incorrect or harmful content is removed. The workspace analogy is an architectural motivation and does not imply that the agents have conscious experience.

REPRODUCIBILITY STATEMENT

Section 3 defines the workspace interface and training objectives, and Section 4.1 summarizes the benchmarks, baselines, decoding settings, and scoring protocol. Appendix E records study-specific configurations, seeds, checkpoint selection, and controller parameters; Algorithms 1–3 specify rollout construction, workspace updates, and optimization. Appendix H provides the expert and judger prompts, while Appendix B reports the controlled comparisons and seed-averaging protocol. Appendix D defines the scope of the communication-cost measurements. Table 1 jointly reports accuracy and communication costs under one configuration for each task–backbone–method combination, using the same evaluation questions, seeds, and, for GLW, validation-selected checkpoints.

REFERENCES

Bernard J. Baars. Global workspace theory of consciousness: Toward a cognitive neuroscience of human experience. *Progress in Brain Research*, 150:45–53, 2005. doi: 10.1016/S0079-6123(05)50004-9.

Yoshua Bengio. The consciousness prior, 2017. URL https://arxiv.org/abs/1709.08568.

Paul C. Bogdan and Jack Lindsey. Slot machines: How LLMs keep track of multiple entities, 2026. URL https://arxiv.org/abs/2604.21139.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Stanislas Dehaene and Lionel Naccache. Towards a cognitive neuroscience of consciousness: Basic evidence and a workspace framework. *Cognition*, 79(1–2):1–37, 2001. doi: 10.1016/S0010-0277(00)00123-2.

European Commission. The global latent workspace: Towards AI models of flexible cognition (GLoW). CORDIS Project Fact Sheet, Grant Agreement ID: 101096017, 2023. URL https://cordis.europa.eu/project/id/101096017. Project started on 2023-09-01; accessed 2026-09-25.

Jacob Fein-Ashley, Dhruv Parikh, Rajgopal Kannan, and Viktor Prasanna. Mixture of thoughts: Learning to aggregate what experts think, not just what they say. *arXiv preprint arXiv:2509.21164*, 2025. doi: 10.48550/arXiv.2509.21164. URL https://arxiv.org/abs/2509.21164v1.

Muxin Fu, Xiangyuan Xue, Yafu Li, Zefeng He, Siyuan Huang, Xiaoye Qu, Yu Cheng, and Yang Yang. LatentMem: Customizing latent memory for multi-agent systems, 2026. URL https://arxiv.org/abs/2602.03036.

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

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing machines, 2014. URL https://arxiv.org/abs/1410.5401.

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, 2026. URL https://transformer-circuits.pub/2026/workspace/index.html.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. QASC: A dataset for question answering via sentence composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8082–8090, 2020. doi: 10.1609/aaai.v34i05.6319. URL https://ojs.aaai.org/index.php/AAAI/article/view/6319.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need, 2024a. URL https://arxiv.org/abs/2402.05120.

Zhaoyi Li, Gangwei Jiang, Hong Xie, Linqi Song, Defu Lian, and Ying Wei. Understanding and patching compositional reasoning in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9668–9688, Bangkok, Thailand, 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.576. URL https://aclanthology.org/2024.findings-acl.576/.

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. doi: 10.1016/j.neuron.2020.01.026.

Lingrui Mei, Shenghua Liu, Yiwei Wang, Yuyao Ge, Baolong Bi, Jiayu Yao, Jun Wan, Ziling Yin, Jiafeng Guo, and Xueqi Cheng. Gated differentiable working memory for long-context language modeling, 2026. URL https://arxiv.org/abs/2601.12906.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, volume 27, 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/hash/3e456b31302cf8210edd4029292a40ad-Abstract.html.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. MedMCQA: A large-scale multi-subject multi-choice dataset for medical domain question answering, 2022. URL https://arxiv.org/abs/2203.14371.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2827–2836. PMLR, 2017. URL https://proceedings.mlr.press/v70/pritzel17a.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421/.

Rufin VanRullen and Ryota Kanai. Deep learning and the global workspace theory. *Trends in Neurosciences*, 44(9):692–704, 2021. doi: 10.1016/j.tins.2021.04.005. URL https://arxiv.org/abs/2012.10390.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions, 2017. URL https://arxiv.org/abs/1707.06209.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992. doi: 10.1007/BF00992696.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning, 2025. URL https://arxiv.org/abs/2508.19828.

Sikuan Yan, Ahmed Bahloul, Ercong Nie, Susanna Schwarzmann, Riccardo Trivisonno, Volker Tresp, and Yunpu Ma. Memory-R2: Fair credit assignment for long-horizon memory-augmented LLM agents, 2026. URL https://arxiv.org/abs/2605.21768.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models, 2022. URL https://arxiv.org/abs/2210.03629.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. MemGen: Weaving generative latent memory for self-evolving agents, 2025. URL https://arxiv.org/abs/2509.24704.

Yujia Zheng, Zhuokai Zhao, Zijian Li, Yaqi Xie, Mingze Gao, Lizhu Zhang, and Kun Zhang. Thought communication in multiagent collaboration, 2025. URL https://arxiv.org/abs/2510.20733.

Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems, 2025. URL https://arxiv.org/abs/2511.20639.

Jiaru Zou, Rui Pan, Ruizhong Qiu, Pan Lu, Shizhe Diao, Jindong Jiang, Hanghang Tong, Tong Zhang, Markus J. Buehler, Jingrui He, and James Zou. Recursive multi-agent systems. *arXiv preprint arXiv:2604.25917*, 2026. doi: 10.48550/arXiv.2604.25917. URL https://arxiv.org/abs/2604.25917v2.

A Hierarchical Collaboration

All three experts receive the question independently and produce local candidate banks. A controller builds the workspace from these banks, and one final judger reads it to produce the answer. The arrows in Figure A1 summarize this aggregation pattern; experts do not consume one another's outputs. The backbone is shared across roles and remains frozen.

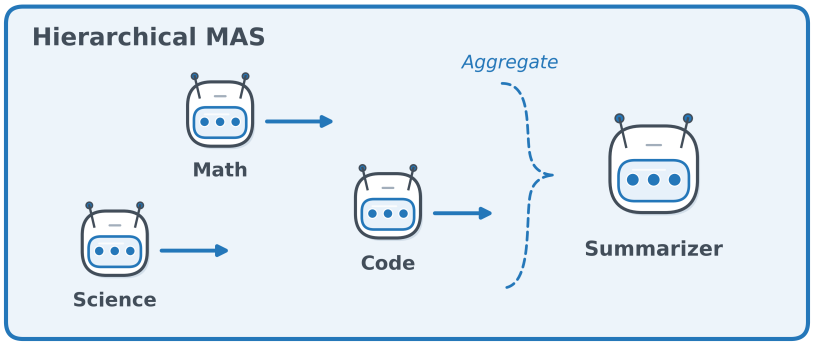


Figure A1: **Hierarchical collaboration with independent experts.** Math, science, and code experts independently process the same question. In GLW, their information is selectively integrated into a bounded workspace that a frozen final agent reads to produce the answer.

B Controlled Ablations and Reporting Protocol

The controls below answer two complementary questions: whether learning improves admission at fixed capacity, and whether frozen-judger feedback improves the update controller. Each ablation holds non-ablated factors fixed. Table A5 maps these studies to models, evaluation settings, and reporting units.

B.1 Learned Admission at Fixed Capacity

Question and controls. The QASC-8B fixed-budget state-selection study tests learned admission while holding the frozen backbones, evaluation questions, 40-position workspace, and 3072-token output cap fixed. Trained, untrained, fixed-first-40, and random-40 conditions use expert candidate states; Zero-vector Workspace fills all 40 slots with zeros.

Workspace construction and training. In this study, the workspace is formed by gathering 40 original candidate vectors from the aligned expert banks (Appendix F.1). Fixed-first-40 preserves the interleaved candidate order. Selection training compares one-for-one replacements within fixed-size candidate sets, with feedback based on answer likelihood and correctness. For each available option letter $y \in \mathcal{O}(x)$, the frozen judger computes its teacher-forcing log-likelihood $\ell_y(M)$ and normalizes over the available options:

$$p_{\text{options}}(y \mid M) = \frac{\exp(\ell_y(M))}{\sum_{u \in \mathcal{O}(x)} \exp(\ell_u(M))}. \tag{10}$$

With correct option $y^\star$ and $\hat{y}(M) = \arg\max_{y \in \mathcal{O}(x)} p_{\text{options}}(y \mid M)$, the feedback for a workspace is

$$R_{\text{sel}}(M) = \log p_{\text{options}}(y^\star \mid M) + 0.5\, \mathbf{1}[\hat{y}(M) = y^\star]. \tag{11}$$

Training feedback is computed entirely from option-letter likelihoods; reported evaluation accuracy follows the answer-generation and scoring protocol in Appendix E.

Results and interpretation. GLW reaches 87.50% accuracy, improving by 2.67 percentage points over both untrained initialization and fixed-first-40, 2.50 over random-40, and 1.67 over Zero-vector Workspace (Table A1), averaged over seeds 42/43/44 on the same evaluation questions. These results support the benefit of learned state selection at a fixed shared-context length.

Table A1: QASC-8B fixed-budget state-selection results underlying Figure 5(a). Every condition supplies a 40-position workspace to the frozen judger. The trained, untrained, fixed-first-40, and random-40 conditions use expert candidate states; Zero-vector Workspace supplies no expert content. Accuracy (%) is reported for seeds 42/43/44 and as the mean across these three seeds.

Condition	Seed 42	Seed 43	Seed 44	Mean
GLW	86.50	88.00	88.00	**87.50**
Untrained selector	84.50	84.50	85.50	84.83
Fixed-first-40 (no updates)	84.50	84.50	85.50	84.83
Random-40	84.50	85.50	85.00	85.00
Zero-vector Workspace	86.50	85.50	85.50	85.83

B.2 Frozen-Judger Attention Feedback

Question and controls. Does the spatial distribution of frozen-judger attention improve controller learning? The ARC-Challenge-8B ablation compares observed attention, uniform attention over workspace slots, shuffled attention across slots, and a disabled attention reward. The uniform and shuffled controls replace the slot weights used to compute write credit in Equation 6. The reward uses $\lambda_J = 0.02$ when enabled and 0 when disabled. Evaluation questions, decoding limits, and non-ablated training settings are held fixed.

Table A2: ARC-Challenge-8B judger-attention ablation: thinking on, held-out test evaluation, and a 4096-token cap. Observed-attention and disabled-reward results include three independent training seeds with coupled generation seeds. Uniform- and shuffled-attention results are also reported for each seed; the final column gives the mean accuracy. The reward uses $\lambda_J = 0.02$ when enabled and 0 when disabled; all non-ablated settings are fixed.

Attention feedback	Seed 42	Seed 43	Seed 44	Mean (%)
Observed	94.00	95.50	95.50	**95.00**
Uniform	94.00	95.00	94.00	94.33
Shuffled	94.00	94.00	94.00	94.00
Disabled	94.00	95.00	94.00	94.33

Results and interpretation. Observed attention achieves the highest reported accuracy, 95.00%, exceeding uniform attention (94.33%) by 0.67 percentage points and shuffled attention (94.00%) by 1.00 point. Uniform attention matches the disabled-reward accuracy of 94.33%, while shuffled attention is 0.33 points lower. These controls examine the contribution of slot-specific attention feedback beyond the presence of an attention-reward term. The observed-attention versus disabled-reward gain is 0.67 percentage points, averaged over seeds 42/43/44 on the same evaluation questions.

B.3 Writer-Level Outcome Feedback

Question and controls. Does retaining counterfactual reward improve the update controller? Figure 5(b) compares reward inclusion under settings fixed within the reported pair. The contrast $R - R_{-i}$ provides writer-level outcome feedback using the suppression procedure in Algorithm 3.

Results and interpretation. Retaining the reward raises the three-seed mean from 94.17% to 94.67%, a 0.50-point improvement. This ablation measures the contribution of writer-level outcome feedback with all other factors held fixed within the pair. Together with the judger-attention ablation, it examines the contributions of the two reward components.

B.4 Seeds and Reporting Units

The per-seed controlled comparisons use seeds 42, 43, and 44 on the same 200 questions. The trained/untrained selection comparison and the observed/disabled judger-attention comparison use three training runs with coupled generation seeds. Tables A1 and A2 report individual seed results and means for all conditions, including fixed-first-40, uniform attention, and shuffled attention. Each reported three-seed mean averages the results on the same evaluation questions across seeds

42/43/44. The main table averages accuracy and communication costs over the same seeds and evaluation questions under the common configuration protocol in Appendix E.

C Controller Behavior and Qualitative Analysis

Behavioral analyses complement the accuracy controls by showing where shared capacity is allocated and how workspace use changes during training. The analyses track GLW's fixed-budget state selection and workspace occupancy during training.

C.1 Position-Wise and Expert-Level Selection

Figure A2 resolves the aggregate selection profile in Figure 5(d) into individual expert profiles. Across the math, science, and code experts, training shifts selection from the earliest positions toward a broader early-to-middle band, with fewer admissions of later states. Random-40 remains approximately uniform. All three policies retain 40 of the 120 candidate states. The accuracy controls in Section B.1 establish the fixed-budget comparison; the common shift here describes how learned admission is distributed across experts and positions.

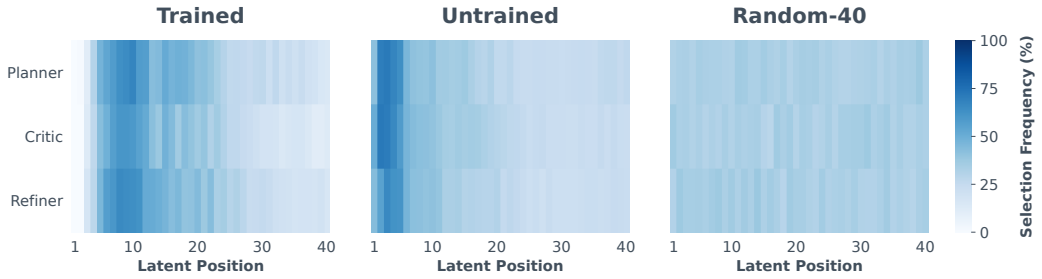


Figure A2: **Expert-level selection patterns at a fixed workspace budget.** Panels compare the trained, untrained, and random-40 policies. Rows denote experts, and horizontal axes index their 40 local latent positions. Colors show selection frequencies (%) averaged across three seeds on a shared 0–100% scale. Each policy selects 40 states from the 120 expert candidates.

C.2 Variable-Occupancy Training Dynamics

Figure 5(c) follows Qwen3-8B training on ARC-Easy and ARC-Challenge with thinking disabled and seed 42. Each curve spans ten epochs, with model fitting on the training set and monitoring on a held-out validation set. Later epochs generally show higher validation accuracy and lower mean occupancy than early epochs. These trajectories illustrate changes within the variable-occupancy executor; they are not additional test-set accuracy estimates or evidence of variability across training seeds. The main-table deployment has constant 40-slot capacity.

C.3 Qualitative Before–After Analysis

We examine two QASC examples from the GLW selection controls in Section B.1. The Qwen3-8B backbone and judger remain frozen. Within each seed and question, both conditions use the same cached pool of 120 candidates from three independent experts and retain the top 40 controller-scored states. The comparison is between the untrained controller and its validation-selected checkpoint; the only trained component is the controller. The settings follow Appendix B.1.

The figures display seed 42. Rows E1–E3 identify the math, science, and code experts; each row contains 40 candidate positions. Filled cells mark selected identities, aligned across conditions for comparison. This grid arrangement is not the workspace input order, which follows selection scores. The supporting QASC facts below are annotations from the labeled validation split used to explain the gold answer, not extra evidence text supplied during evaluation.

Aggregate context. Across the QASC final-evaluation subset and three coupled training/generation seeds, accuracy increases from 84.83% to 87.50%. Among 600 paired responses,

31 change from incorrect to correct and 15 from correct to incorrect; these are repeated evaluations of 200 unique questions. Both illustrated questions improve for seeds 42 and 43 and remain correct for seed 44. The cases illustrate changes in generated answer behavior alongside changes in the selected states.

C.4 Case 1: Respecting the Requested Answer Type

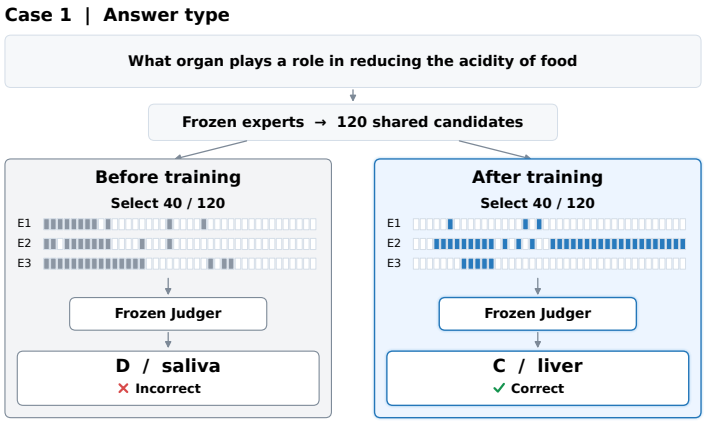


Figure A3: **Respecting the requested answer type.** A before–after GLW comparison on a QASC evaluation example from the official validation split, using Qwen3-8B and seed 42. The same 120-candidate pool and 40-position workspace budget are used in both conditions. The untrained condition answers D, saliva; the trained condition answers C, liver. Filled cells show actual selected candidate identities, with 14 shared between the two sets. The grids show selection rather than attention or decoded latent content.

The question in Figure A3 explicitly requests an *organ*. The untrained response discusses acid buffering and chooses D, saliva, despite acknowledging that it is not an organ: "even though it's not an organ itself, it is the best fit among the choices." The saved response already considers the liver and bile, so the error is a failure to enforce the answer-type constraint in the final decision, rather than simply an absence of related associations.

After training, the response identifies the liver as an organ and connects it to bile and reduced acidity, selecting C. This agrees with the QASC reference chain: bile reduces the acidity of food entering from the stomach, and bile is synthesized by the liver. For seed 42, selection counts across E1/E2/E3 change from 11/11/18 to 3/32/5. Fourteen identities are shared, so each 40-state set contains 26 identities absent from the other. These are differences between selected sets, not 26 recorded swap operations, and do not establish that E2's states specifically encode liver-related knowledge.

C.5 Case 2: Preserving Option Meaning and Linking Facts

In Figure A4, the untrained response focuses on water leaving a sponge through the osculum, but treats option H, "water vapor," as if it said "water." Its final letter H is successfully parsed, so this is an incorrect choice rather than an answer-format failure. The trained response connects sperm release through the osculum with option D, "male gametes." The QASC annotations support this relation: sperm are released into the surrounding water through the osculum, and sperm are male gametes.

Selection counts change from 33/6/1 to 7/1/32 for E1/E2/E3; the two sets share seven identities. This change accompanies a correct final choice, without locating reproductive knowledge in a particular expert. Seed 43 also improves, but its untrained answer is G, precipitation, rather than H; the repeated observation is the answer correction, not an identical textual trajectory. The analysis concerns the key relation and final choice, and does not verify every ancillary biological statement in the generated response. Complete saved responses and selection lists are retained in the case-study evidence records.

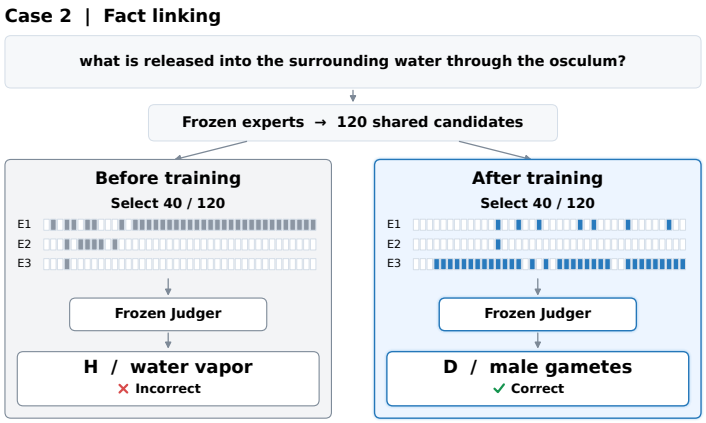


Figure A4: **Preserving option meaning and linking facts.** A GLW comparison on a QASC evaluation example from the official validation split, using Qwen3-8B and seed 42 under the same candidate-pool and capacity controls as Figure A3. The untrained condition selects H, water vapor; the trained condition selects D, male gametes. Both sets contain 40 candidates, with seven shared identities. Filled cells indicate candidate selection; their expert-aligned layout does not represent workspace order or individual contribution magnitudes.

D Communication Cost and Resource Accounting

D.1 Generated Output and Judger Context

Output tokens count generated text, including thinking. Judger-context length counts input and latent positions. In Table 1, these costs and accuracy share the same configuration for each task, backbone, and method, including the same GLW checkpoint for each seed. Table A3 reproduces the QASC-8B and ARC-Challenge-8B output-token entries from Table 1 for reference.

Table A3: Mean generated output tokens, including thinking, on QASC-8B and ARC-Challenge-8B, reproduced from Table 1. Each entry uses the same configuration and evaluation as its corresponding main-table accuracy result; GLW uses the update controller with a 40-slot workspace.

Dataset	SA	TextMAS	LatentMAS	**GLW**
QASC	1078	2923	707	890
ARC-Challenge	812	2376	557	754

D.2 Output-Length Distributions

Figure A5 complements mean token costs with the output-length distribution across individual questions. On both benchmarks, GLW lies clearly to the left of TextMAS across a broad range of quantiles, showing a distribution-wide reduction rather than a change confined to a few long outputs. On QASC-4B, GLW is generally shorter than SingleAgent and close to LatentMAS. The overlapping GLW checkpoint curves indicate similar output-length profiles among the checkpoints shown. On ARC-Challenge-8B, GLW remains substantially shorter than TextMAS and closer to SingleAgent, while LatentMAS generally generates shorter outputs.

The configured output caps are 4096 for QASC-4B and 3072 for ARC-Challenge-8B. These distributions characterize generated-output cost across both correct and incorrect responses in the displayed settings. Table 1 jointly reports accuracy and mean communication costs under the same configuration.

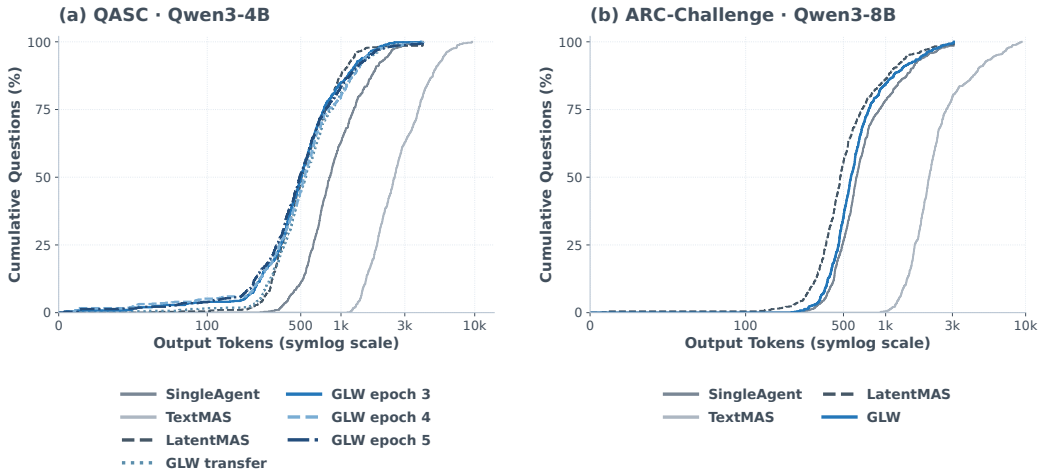


Figure A5: **Output-length distributions on QASC-4B and ARC-Challenge-8B.** Curves show empirical cumulative distributions averaged over three generation seeds on the same evaluation examples for each benchmark. Output tokens include thinking; TextMAS totals include all emitting agents. At a given cumulative percentage, a leftward curve indicates shorter outputs. The horizontal axis uses a symlog scale. QASC shows multiple GLW checkpoints; curves summarize both correct and incorrect responses.

D.3 Inference and Training Computation

GLW and LatentMAS use the same expert prompts and generate 40 local states per expert, totaling 120 states per question. The fixed-capacity GLW interface exposes 40 workspace positions to the judger, together with its own prompt. Inference includes expert computation, workspace construction, judger re-encoding, and answer generation. The reported communication metrics are generated-output length and judger-context length.

Algorithm 3 also makes the additional training work explicit. For each question, it reuses one independent candidate bank across G factual rollouts and three writer-suppressed rollouts. Each factual rollout requires an additional frozen-judger attention-extraction pass; the group then yields one controller-only optimizer step.

E Training and Evaluation Settings

Table A4 reports benchmark sizes and the training-transfer and evaluation protocol.

Table A5 summarizes the experimental protocols and reporting scope; Table A6 records shared interface dimensions and the optimization parameters for the judger-attention ablation.

Main-table configuration and metric pairing. Each task–backbone–method combination in Table 1 is evaluated with one configuration, defined by its prompts, decoding settings, output-token cap, and, for GLW, the validation-selected controller checkpoint for each seed. The same evaluation questions and seeds 42/43/44 are used for accuracy, output-token count, and judger-context length. The reported output-token cap is held fixed across all three metrics. All main-table GLW results use the update controller with blending: $N = 3$ independent experts produce $L = 40$ candidates each, the first 40 candidates initialize $S_0 = S_{\max} = 40$ workspace slots, and the remaining 80 receive SKIP/WRITE decisions. Figures 1 and 4 summarize these same main-table configurations. Table A5 also records the protocols for the selection controls, reward ablations, training curves, and output-length analyses.

Evaluation slices, labels, and scoring. Final results use the labeled evaluation slices in Table A4: QASC and CommonsenseQA use `validation[200:400]`, and MedMCQA uses `validation[0:200]`. For these three benchmarks, answers come from the public validation

Table A4: **Benchmark statistics and evaluation protocol.** All controllers are trained on ARC-Challenge and transferred to the other benchmarks. Final evaluation uses the dataset-specific labeled slices shown below. In slice notation, `val` denotes the official validation split; indices are zero-based and the upper endpoint is excluded.

Benchmark	**Official total** (train / val / test)	**Controller source**	**Validation set**	**Final evaluation**
ARC-Challenge	2,590 (1,119 / 299 / 1,172)	Trained on ARC-Challenge	200; `val[0:200]`	200; `test[0:200]`
ARC-Easy	5,197 (2,251 / 570 / 2,376)	Transferred from ARC-Challenge	200; `val[0:200]`	200; `test[0:200]`
QASC	9,980 (8,134 / 926 / 920)	Transferred from ARC-Challenge	200; `val[0:200]`	200; `val[200:400]`
CommonsenseQA	12,102 (9,741 / 1,221 / 1,140)	Transferred from ARC-Challenge	200; `val[0:200]`	200; `val[200:400]`
SciQ	13,679 (11,679 / 1,000 / 1,000)	Transferred from ARC-Challenge	200; `val[0:200]`	200; `test[0:200]`
MedMCQA	193,155 (182,822 / 4,183 / 6,150)	Transferred from ARC-Challenge	200; `val[0:200]`	200; `val[0:200]`

Table A5: **Experimental configurations and reporting scope.** Caps refer to generated output tokens. Three-seed evaluations use 42/43/44. Within each task–backbone–method combination, the main table reports all metrics under one configuration. Ablations hold non-ablated factors fixed within each comparison.

Evidence and controller	**Model, task, and evaluation**	**Replication and scope**
Main comparison; GLW update controller with blending	Qwen3-4B/8B; six tasks; thinking on; 3072–4096 caps; $N = 3$, $L = 40$, $S_0 = S_{\max} = 40$	Three-seed means; accuracy and costs share the configuration, evaluation questions, and checkpoint per seed.
Fixed-budget state-selection study	QASC-8B; held-out evaluation; thinking on; 3072-token cap; 40 selected candidate states	Trained and control selections; feedback based on answer likelihood and correctness; per-seed results and means for all conditions.
Judger-attention control; update controller	ARC-Challenge-8B; test evaluation; thinking on; 4096 cap	Observed, uniform, shuffled, and disabled attention; per-seed results and means for all four conditions.
Counterfactual pair; update controller	ARC-Challenge-8B; test evaluation; reward-on/off comparison	Three-seed means; non-ablated settings held fixed within the pair.
Training dynamics; variable-occupancy update controller	ARC-Easy/Challenge-8B; thinking off; validation monitoring; ten epochs	Seed 42; validation trajectory, not a three-seed test comparison.
Main-table output-token summary	QASC/ARC-Challenge-8B; thinking on; main-table evaluations	Reproduces the main-table token means with the corresponding accuracy configuration.
Output-length distributions	QASC-4B / ARC-Challenge-8B; final evaluation; 4096 / 3072 caps	Three-seed mean CDFs; checkpoint-specific output costs including incorrect answers.

annotations; no official hidden-test answer labels are used. These results should therefore be interpreted as validation-subset evaluations, not official test-set scores. ARC-Easy, ARC-Challenge, and SciQ use `test[0:200]`. For QASC and CommonsenseQA, the final-evaluation slice is disjoint from the `validation[0:200]` slice listed for validation. MedMCQA uses the same slice listed in the validation column; we do not characterize it as an independent held-out test set. Thinking-enabled evaluations use temperature 0.6 and top-p 0.95. Scoring prioritizes the final boxed answer

Table A6: **Interface dimensions and training parameters.** The optimization block describes the ARC-Challenge-8B judger-attention control. The old collection policy is fixed within each rollout group; the reference policy is synchronized at the interval below.

Parameter	Value	Application
Hierarchical interface and fixed-budget selection		
Experts N	3	Independent math, science, and code roles.
Local states per expert L	40	120 candidates before shared admission.
Shared-state budget	40	Main deployment and all selection controls.
Initial slots S_0	40	Fixed-capacity update executor; 80 later gate decisions.
Selection-control capacity	40	Fixed workspace length across admission controls.
Backbone updates	None	Expert and judger weights remain frozen.
Judger-attention control training		
Factual group size G	4	Sampled trajectories per question.
Optimizer / learning rate	Adam / 10^{-5}	Controller parameters only.
PPO clipping ϵ	0.2	Clipped policy-ratio objective.
KL coefficient β_{KL}	0.01	Periodically synchronized reference policy.
Entropy coefficient η	0.05	Policy entropy term.
Gradient-clipping threshold	1.0	Controller optimizer step.
Reference interval K	100 steps	Optimizer steps between reference updates.
Judger weight λ_J	0.02 / 0	Reward enabled / disabled.
Content weight λ_{write}	1	Advantage-weighted reference content term.
Prior weight λ_{ws}	1	Write-probability prior.
Target write probability	0.8	Prior target p_{target}.
Output-token cap	4096	Training and evaluation in this control.

and then the last explicit answer expression; invalid or missing answers are incorrect. Training curves report validation performance, as described in Section C.

Fixed-budget selection training. The QASC-8B selection study uses training data for optimization and non-overlapping subsets of the official labeled validation split for checkpoint selection (`validation[0:200]`) and final evaluation (`validation[200:400]`). Validation-selected checkpoints are epochs 2, 4, and 4 for seeds 42, 43, and 44, respectively. The untrained condition uses each seed's pre-training controller initialization. The qualitative examples in Section C.3 come from the final-evaluation slice `validation[200:400]` and show seed 42, with thinking enabled, a 3072-token generation cap, temperature 0.6, and top-p 0.95. The selection-control settings are specified in Appendix B.1.

E.1 Interface and Controller Parameters

Workspace-update controller architecture. The controller in Algorithm 2 uses an addressing head, erase/add heads, and a two-layer gating MLP with hidden dimension 256; the question projection has dimension 64. Linear weights use Xavier uniform initialization with gain 0.1, and biases are zero. The controller contains 21,163,268 parameters with Qwen3-4B and 52,724,996 with Qwen3-8B. These dimensions, counts, and initialization describe the SKIP/WRITE controller that performs erase/add updates. The scoring and training procedure for fixed-budget state selection is specified in Appendix B.1.

Variable-occupancy training can initialize with one candidate and expand up to its configured capacity. Fixed-capacity update inference initializes all 40 slots before the policy acts. In both update executors, deterministic initialization is excluded from policy actions and policy loss.

F Implementation and Algorithms

The architecture in Figure 3 separates independent expert computation from workspace construction. For each question, the three frozen experts generate $H_i = (h_{i,1}, \ldots, h_{i,40})$. Candidates retain their writer and step indices and are interleaved as $C = (h_{1,1}, h_{2,1}, h_{3,1}, h_{1,2}, \ldots, h_{3,40})$. Algorithm 1 constructs the workspace and performs the final readout; Algorithm 2 specifies each policy-controlled update, and Algorithm 3 defines the training loop.

F.1 Candidate Construction and Workspace Initialization

Latent generation and alignment. Each expert first encodes its role prompt and the question. At each of 40 latent steps, it extracts the last decoder layer's representation at the last input position, after the final RMSNorm and before the vocabulary output head. Let E_{in} and E_{out} be the frozen input embedding and output-head matrices, with vocabulary items as rows. For a row-vector hidden state $\tilde{h}$, the fixed alignment and scaling are

$$P = (E_{\text{out}}^{\top} E_{\text{out}} + 10^{-5} I)^{-1} E_{\text{out}}^{\top} E_{\text{in}}, \tag{12}$$

$$c(\tilde{h}) = \frac{r\,\tilde{h}P}{\max(\|\tilde{h}P\|_2, 10^{-6})}, \tag{13}$$

where r is the mean row norm of E_{in}. The aligned vector is supplied as the next step's `inputs_embeds`, reusing the expert's KV cache throughout; no intermediate text token is sampled. GLW stores the 40 aligned input vectors actually fed back to the model, denoted $h_{i,\ell} = c(\tilde{h}_{i,\ell})^{\top}$ in the column-vector notation of the algorithms. The projection P remains fixed during controller training. GLW and the local LatentMAS baseline share this generation-stage alignment function.

Initialization and capacity. Let S_0 be the number of initial slots, with $1 \le S_0 \le S_{\max}$. The fixed-capacity deployment sets $S_0 = S_{\max} = 40$: the first 40 candidates initialize the workspace and the remaining 80 receive gate decisions. Variable-occupancy training can start with the first candidate and permit expansion up to $S_{\max}$. Initialization is deterministic and contributes no policy action or policy loss. Expansion and blending are update modes chosen by capacity and novelty after the binary action WRITE.

Writer-suppressed rollouts. A suppressed rollout uses the same candidate banks and forces SKIP on policy decisions assigned to writer i. Later workspace decisions can change in response to the altered workspace, while expert candidates remain fixed. Each suppressed score is reused across the factual rollout group.

F.2 Optimization and Auxiliary Terms

Table A6 specifies the judger-attention control parameters. Group standardization is over included transition rewards, rather than final-answer scores alone. The old collection policy supplies the likelihood-ratio denominator, while the periodically synchronized reference policy supplies the KL and auxiliary reference terms.

Auxiliary controller terms. With $\mathcal{T}$ the included policy transitions and $U_k, U_{k,\text{ref}}$ the hypothetical erase/add outputs at the same detached pre-update workspace, the auxiliary terms are

$$\mathcal{L}_{\text{content}} = \frac{\lambda_{\text{write}}}{|\mathcal{T}|} \sum_{k \in \mathcal{T}} \mathbf{1}[a_k = \text{WRITE}]\, \text{sg}(A_k)\, \text{MSE}(U_k, \text{sg}(U_{k,\text{ref}})), \tag{14}$$

$$\mathcal{L}_{\text{prior}} = \lambda_{\text{ws}}\, \mathbb{E}_{k \in \mathcal{T}}[\max(0, p_{\text{target}} - \pi_\phi(\text{WRITE} \mid s_k))^2]. \tag{15}$$

The full update objective adds these terms to Equation 8. MSE averages over slot and hidden dimensions, as in Equation 9; loss averaging includes both writes and skips. The content term compares current and reference outputs at stored pre-update workspaces; it does not backpropagate task correctness through the judger. Since advantages are signed, this term is not uniformly nonnegative. The selection feedback is specified in Equation 11.

F.3 Rollout and Workspace Updates

Algorithm 1 Hierarchical GLW rollout and final readout

Require: question x, tagged independent candidates $C = (C_1, \ldots, C_{120})$, frozen judger J, controller π, initialization size S_0, capacity $S_{\max}$, novelty threshold θ_ν, judger prompt p_J, suppression index $b \in \{0, 1, 2, 3\}$, flag *training*
Ensure: answer y, policy-transition trace τ, final workspace M
$q \leftarrow \mathrm{MeanPool}(\mathrm{TokenEmbed}(x))$
$M \leftarrow [\mathrm{State}(C_1); \ldots; \mathrm{State}(C_{S_0})]$; $\tau \leftarrow \varnothing$
for $t = S_0 + 1, \ldots, 120$ **do**
  $(i, \ell, h) \leftarrow \mathrm{WriterStepState}(C_t)$
  if $i \neq b$ **then**
    $M_{\mathrm{pre}} \leftarrow \mathrm{DetachedCopy}(M)$
    $(M, a, u, \log p_a, z) \leftarrow \mathrm{ControllerStep}(h, M, q, \pi, \theta_\nu, S_{\max}, \textit{training})$
    Append $(i, \ell, h, M_{\mathrm{pre}}, a, u, \log p_a, z)$ to τ
  end if
end for
Insert M after the user-turn delimiter in the judger's embedding input
$y \leftarrow J.\mathrm{Generate}(x, p_J, M)$ with fresh judger KV
return (y, τ, M)

Here $b = 0$ denotes an ordinary rollout; $b = i$ suppresses writer i's policy-controlled writes. Training samples policy actions; final inference uses their argmax. The trace contains only controller decisions, and z records addressing weights or a new-slot index for attention credit.

The binary policy action $a \in \{\textsc{skip}, \textsc{write}\}$ is kept separate from the executed update mode $u \in \{\textsc{skip}, \textsc{expand}, \textsc{blend}\}$. Policy probabilities and likelihood ratios always refer to a. Initial slots are supplied by Algorithm 1 before the controller is called, so forced initialization is never included as a sampled action.

Algorithm 2 Binary controller step and deterministic update mode

Require: candidate h, nonempty workspace $M \in \mathbb{R}^{S \times d}$, $1 \leq S \leq S_{\max}$, question embedding q, policy π, novelty threshold θ_ν, capacity $S_{\max}$, flag *training*
Ensure: $(M', a, u, \log p_a, z)$, where z stores attention-credit metadata
Compute c, w^c, w using Equations 1–2
$\mu \leftarrow \max_j c_j$; $\nu \leftarrow 1 - \mu$; $\hat{m} \leftarrow \sum_j w_j^c M[j]$
$s \leftarrow (h, \hat{m}, \mu, \nu, f_q(q))$; $p \leftarrow \pi(\cdot \mid s)$
if *training* **then**
  $a \sim \mathrm{Categorical}(p)$
else
  $a \leftarrow \arg\max_{a' \in \{\textsc{skip}, \textsc{write}\}} p[a']$
end if
$\log p_a \leftarrow \log p[a]$
if $a = \textsc{skip}$ **then**
  return $(M, \textsc{skip}, \textsc{skip}, \log p_a, \varnothing)$
end if
if $\nu > \theta_\nu$ and $S < S_{\max}$ **then**
  $M' \leftarrow [M; h]$; $j_{\mathrm{new}} \leftarrow S + 1$
  return $(M', \textsc{write}, \textsc{expand}, \log p_a, j_{\mathrm{new}})$
end if
$e \leftarrow \sigma(W_e h)$; $a^+ \leftarrow W_a h$
for $j = 1, \ldots, S$ **do**
  $M'[j] \leftarrow M[j] \odot (1 - w_j e) + w_j a^+$
end for
return $(M', \textsc{write}, \textsc{blend}, \log p_a, \mathrm{DetachedCopy}(w))$

When $S = S_{\max}$, expansion is impossible, regardless of novelty. All blended slots are computed from the same pre-update workspace; the update does not use a slot already modified earlier in the

loop. The returned metadata z records the new index for expansion or the addressing distribution for blending. At the end of a rollout, addressing weights are padded with zeros for any slots added later before applying Equation 6.

Algorithm 3 Hierarchical controller training from frozen-judger feedback

Require: data $\mathcal{D}$, frozen experts A_1, A_2, A_3 and judger J, role prompts p_1, p_2, p_3, p_J, controller π_ϕ, group size G, initialization S_0, capacity $S_{\max}$, novelty threshold θ_ν, reward and loss coefficients, reference interval K
Ensure: trained controller parameters ϕ

$\pi_{\text{ref}} \leftarrow \text{FrozenCopy}(\pi_\phi)$
for each epoch and each $(x, y^*) \in \mathcal{D}$ **do**
  Generate H_1, H_2, H_3 independently with frozen experts; form tagged bank C
  $\pi_{\phi_{\text{old}}} \leftarrow \text{FrozenCopy}(\pi_\phi)$
  for $i = 1, 2, 3$ **do**
    $(y_{-i}, _, _) \leftarrow \text{HierarchicalRollout}(x, C, J, \pi_{\phi_{\text{old}}}, b = i, \textit{True})$
    $R_{-i} \leftarrow \mathbf{1}[\text{Parse}(y_{-i}) = y^*]$
  end for
  for $g = 1, \ldots, G$ **do**
    $(y^{(g)}, \tau^{(g)}, M^{(g)}) \leftarrow \text{HierarchicalRollout}(x, C, J, \pi_{\phi_{\text{old}}}, b = 0, \textit{True})$
    $R^{(g)} \leftarrow \mathbf{1}[\text{Parse}(y^{(g)}) = y^*]$
    $p^{\text{J,(g)}} \leftarrow \text{FrozenJudgerAttention}(x, y^{(g)}, M^{(g)})$
    for each policy transition $k = (g, i, \ell)$ in $\tau^{(g)}$ **do**
      $R_{\text{CF}}^{(g,i)} \leftarrow R^{(g)} - R_{-i}$; $r_k \leftarrow 0$
      if $a_k =$ WRITE **then**
        Compute κ_k from mode u_k, stored z_k, and $p^{\text{J,(g)}}$ using Equation 6
        $r_k \leftarrow \max(0, R_{\text{CF}}^{(g,i)}) + \lambda_J \,\text{sign}(R^{(g)} - \frac{1}{2})\kappa_k$
      end if
    end for
  end for
  $\mathcal{B} \leftarrow$ included factual policy transitions from $\tau^{(1)}, \ldots, \tau^{(G)}$
  If $\mathcal{B} = \varnothing$, skip this update
  $A_k \leftarrow \text{stopgrad}\big[(r_k - \text{mean}_{v \in \mathcal{B}}\, r_v)/(\text{std}_{v \in \mathcal{B}}\, r_v + \epsilon_A)\big]$
  Recompute current and reference distributions at the stored pre-update states
  $\rho_k \leftarrow \exp(\log \pi_\phi(a_k \mid s_k) - \log p_{a_k}^{\text{old}})$
  $\mathcal{L} \leftarrow \mathcal{L}_{\text{GRPO}} + \mathcal{L}_{\text{content}} + \mathcal{L}_{\text{prior}}$
  Take an optimizer step on ϕ only, with gradient clipping; keep experts and judger frozen
  if the optimizer-step count is a multiple of K **then**
    $\pi_{\text{ref}} \leftarrow \text{FrozenCopy}(\pi_\phi)$
  end if
end for
return ϕ

The rollout calls use Algorithm 1 with the same C, S_0, θ_ν, role prompts, and decoding settings; shared arguments are omitted from the calls for brevity. Suppression acts only after initialization. Answer parsing follows Appendix H. Rollout collection and judger-attention extraction are gradient-free. Transition traces store detached candidate states, pre-update workspaces, update metadata, and collection-time log probabilities. In the likelihood ratio, s_k denotes the raw observation $(h_k, M_{\text{pre},k}, q)$: each policy recomputes its own addressing and derived features from that observation. The constant $\epsilon_A > 0$ prevents division by zero. Any configured sample mask is applied before group standardization and loss averaging; initialization and forced suppression are excluded. The old policy is fixed during each group's optimization, whereas the reference policy is updated only at the specified interval.

G WORKED TECHNICAL EXAMPLES

These examples explain the implementation. Except for the explicitly cited selection index set, they are constructed illustrations, not observed model trajectories or additional experiments.

G.1 INDEPENDENT CANDIDATE SCHEDULING

Let $h_{i,\ell}$ denote latent step ℓ from expert i. The hierarchical update executor interleaves the banks as

$$C = (h_{1,1}, h_{2,1}, h_{3,1}, h_{1,2}, \ldots, h_{3,40}).$$

The first 39 entries contain 13 states per expert; entry 40 is $h_{1,14}$. The initial workspace therefore contains 14/13/13 states. The next controller decision sees $h_{2,14}$ and a full workspace. A skip leaves every slot unchanged; a write blends into the existing 40 slots. No expansion is possible in this executor, and initialization decisions are not trained gate actions. Every expert bank was computed before these updates, so suppressing a write at inference does not recompute the other banks.

G.2 ONE ERASE/ADD UPDATE

For illustration, take two slots and two dimensions:

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad w = (0.75, 0.25), \quad e = (0.5, 0.5), \quad a^{+} = (0.2, 0.4).$$

Equation 4 gives

$$M_1' = (1, 0) \odot (0.625, 0.625) + (0.15, 0.30) = (0.775, 0.30),$$
$$M_2' = (0, 1) \odot (0.875, 0.875) + (0.05, 0.10) = (0.05, 0.975).$$

The addressing distribution controls where an update acts; the erase and add vectors control its content. These synthetic coordinates have no assigned semantic meaning and are not decoded language-model thoughts.

When only one slot is present, softmax addressing always assigns it weight one, regardless of content similarity. Using that weight as match quality would erase the novelty signal. Raw cosine match quality preserves the distinction between a matching and an unrelated candidate at one-slot occupancy.

G.3 OUTCOME AND ATTENTION CREDIT

With final judger attention $p^{\mathrm{J}} = (0.8, 0.2)$, the preceding write receives $\kappa = 0.75(0.8) + 0.25(0.2) = 0.65$. If the full answer is correct and the writer-suppressed answer is incorrect, $R_{\mathrm{CF}}^{(i)} = 1$ and the reward at $\lambda_J = 0.02$ is $1 + 0.02(0.65) = 1.013$. If the full answer is wrong and the suppressed answer correct, the positive-part counterfactual is zero and the reward is -0.013. Thus the implemented objective discards negative counterfactual credit in this family but can still penalize an attended write on an incorrect rollout.

When p^{J} is uniform over 40 positions, any normalized addressing distribution yields $\kappa = 1/40$, so the attention contribution is only ± 0.0005. This arithmetic explains why nonuniform attention and suitable controls matter before interpreting the reward as spatial credit assignment.

G.4 A FIXED-SIZE SELECTION SWAP

For a toy capacity of three, consider a selected candidate set $\{2, 4, 3\}$. A training alternative can remove candidate 4 and add candidate 5, producing $\{2, 5, 3\}$ without changing the prefix length. The resulting workspace gathers the selected candidate vectors, and Equation 11 scores each candidate set during training.

For an actual saved QASC example (seed 42), the selected candidate indices, in the stored zero-based order, begin $56, 49, 50, 48, 51, 87, 52, 88$. The full 40-index set is included in the figure-data JSON. These indices identify the selected candidate vectors and their score-based workspace order.

G.5 Prefix Placement and Attention Extraction

Workspace vectors are inserted into the judger's embedding sequence after the user-turn delimiter, while its system instruction and question remain present. A symbolic sequence is

```
[system turn] <|im_start|>user
[workspace vector 1] ... [workspace vector 40]
[judger instruction + question] [assistant generation prefix]
```

The bracketed vectors are continuous embeddings, not literal text. The model constructs fresh attention keys and values for this sequence, unlike a transfer of previously computed expert KV. For training-time attention credit, an additional gradient-free pass appends the re-tokenized generated answer, averages final-layer attention over heads and answer-token query positions, selects workspace keys, and normalizes within that region. The query positions are those occupied by answer tokens in this pass, rather than the preceding positions whose logits predicted those tokens during generation. This gradient-free pass supplies attention credit while keeping the generated answer and backbone weights fixed.

H Prompt Templates and an Illustrative Question

The examples below are exported from the current prompt functions. Line wrapping is typographic; the wording is unchanged. The wheel question was written for this appendix and is not a benchmark example. The TextMAS context placeholder is also illustrative, not a generated response. The shared system message and hierarchical role instructions are shown below. The shaded panels preserve the prompt wording; the labels above them are presentation aids and are not sent to the model.

H.1 Role Names and Hierarchical Independence

The internal roles `planner`, `critic`, and `refiner` map to math, science, and code agent instructions in hierarchical MCQ evaluation. Each expert receives the question independently. The final judger aggregates their information through the workspace; these internal role names do not imply an ordered chain of expert interactions.

H.2 Shared System Message

SHARED SYSTEM MESSAGE

```
You are Qwen, created by Alibaba Cloud. You are a helpful assistant.
```

H.3 Hierarchical Expert: Math Role

EXPERT / MATH

```
You are a math agent. Given the input question, reason step-by-step
and put the final answer inside \boxed{YOUR_FINAL_ANSWER}.
Your final answer must be selected from the lettered options shown
in the question. For example \boxed{A}. Do not add any other
contents inside the box.

Input Question: A wheel makes 12 revolutions in 3 seconds. What is
its average rate?
A. 2 revolutions per second
B. 4 revolutions per second
C. 9 revolutions per second
D. 36 revolutions per second

Your response:
```

H.4 Science and Code Expert Instructions

The other two hierarchical experts use the same question, answer-format instruction, and response delimiter as the math expert. Their opening role sentences are given below. Each is followed by the same answer-format instruction, input question, options, and response delimiter shown above.

EXPERT / SCIENCE

```
You are a science agent. Given the input question, reason
step-by-step and put the final answer inside
\boxed{YOUR_FINAL_ANSWER}.
```

EXPERT / CODE

```
You are a code agent. Given the input question, reason step-by-step
and put the final answer inside \boxed{YOUR_FINAL_ANSWER}.
```

Each instruction belongs to a separate expert message. Role specialization changes the prompt, while all three experts use the same frozen backbone.

H.5 Hierarchical Latent Judger

FINAL READER / LATENT WORKSPACE

```
You are a task summarizer. Given the input question and responses
from previous agents as reference, reason step-by-step and put the
final answer inside \boxed{YOUR_FINAL_ANSWER}.
Your final answer must be selected from the lettered options shown
in the question. For example \boxed{A}. Do not add any other
contents inside the box.

Input Question: A wheel makes 12 revolutions in 3 seconds. What is
its average rate?
A. 2 revolutions per second
B. 4 revolutions per second
C. 9 revolutions per second
D. 36 revolutions per second

Your response:
```

H.6 TextMAS Judger with an Explicit Message Context

FINAL READER / TEXT MESSAGES

```
You are a task summarizer. Given the input question and responses
from previous agents as reference, reason step-by-step and put the
final answer inside \boxed{YOUR_FINAL_ANSWER}.
Your final answer must be selected from the lettered options shown
in the question. For example \boxed{A}. Do not add any other
contents inside the box.

Content from Previous Agents:
[ILLUSTRATIVE EXPERT TEXT: replace with actual generated messages]

Input Question: A wheel makes 12 revolutions in 3 seconds. What is
its average rate?
A. 2 revolutions per second
B. 4 revolutions per second
C. 9 revolutions per second
D. 36 revolutions per second

Your response:
```

H.7 Answer Parsing and Limits of the Example

For this constructed question the mathematically correct option is B, since $12/3 = 4$ revolutions per second. A well-formed final answer is $\texttt{\textbackslash boxed\{B\}}$. This is a format example, not an inference run. The evaluator first checks the final boxed answer and falls back to the last explicit answer expression; an option mentioned earlier in reasoning should not be counted as the prediction. Invalid or absent final answers are scored incorrect. Exact task formatting and the tokenizer's chat template must be preserved when reproducing an evaluation.

I Discussion and Reproducibility

I.1 From Local Processing to Shared Access

The workspace analogy concerns how information is made available for use. Global Workspace Theory distinguishes extensive specialized processing from the selected content available to other processes (Baars, 2005; Dehaene & Naccache, 2001). In GLW, the corresponding engineering choices are independent expert computation, controller-mediated admission, and a common reader interface. These choices do not establish conscious access in an agent. Nor does a 40-slot limit imply a 40-concept capacity: each slot is a dense vector, and its semantic content has not been decomposed.

The Jacobian-lens analysis of Gurnee et al. (2026) identifies directions within a model's residual stream, whereas our controller manages externally supplied hidden states. The two uses of "workspace" share a functional motivation but are different objects with different evaluation requirements.

I.2 Task-Dependent Admission and Downstream Reuse

One test suggested by this perspective is to hold the expert candidate bank fixed while changing the question presented to the controller and reader. For two questions that need different facts from the same bank, successful task-dependent admission should change which states are retained. A useful comparison would apply the selector conditioned on the wrong question, alongside the existing random and untrained controls, with capacity and decoding fixed. This would test whether selection responds to the intended use of information rather than only its salience in the candidate bank.

A second test would hold a selected workspace fixed and vary the downstream reasoning operation. If the same intermediate evidence supports several appropriate operations, that would provide evidence of reuse beyond one answer trajectory. Controlled state substitutions could probe which answer changes depend on the replaced content. These studies would require matched prompts, candidate banks, and stochastic decoding. These are proposed experiments, and no corresponding measurements are reported here.

I.3 Result Sources and Reproduction

The source package includes plotting code, frozen numeric JSON inputs, exported prompts, and a SHA-256 manifest of supporting evaluation artifacts. Only completed evaluations contribute benchmark scores; partial training rollouts do not. The supplementary ARC results are evaluated on held-out test data. Reproduction requires preserving the topology, output cap, checkpoint, and evaluation protocol for each configuration.