NEGOTIATED RECURSIVE REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Small recursive networks solve structured problems by repeatedly refining a latent state with shared weights. Splitting a problem into overlapping local views exposes its constraints but creates a coordination problem: in Sudoku, a row, a column, and a box can propose different values for their shared cell. We introduce Negotiated Recursive Reasoning (NRR), which coordinates local recursion while it takes place. A shared local reasoner refines a proposal and a private reasoning state for each view, while an explicit negotiation step constructs targets that reduce overlap disagreement and carries unresolved disagreement into the next round. We use a normalized soft-consensus system with a bounded condition number and characterize the gradient path through which task supervision learns the coupling. In the benchmark setting, NRR reaches 95.2% exact solve on Sudoku-Extreme and 90.4% on Maze-Hard, 5.3 points above a learned-link replacement on both tasks, with larger gains on harder Sudoku puzzles. In controlled comparisons, NRR exceeds a native TRM rerun by 6.2 and 3.6 points, and disagreement memory improves both recursive and feed-forward local solvers, with larger mean gains for recursive updates. On larger mazes, sparse coordination gives a better accuracy–latency tradeoff than global attention recurrence. These results identify the organization of interaction among partial solutions as a useful design dimension for recursive reasoning, alongside model capacity and recursion depth.

1 INTRODUCTION

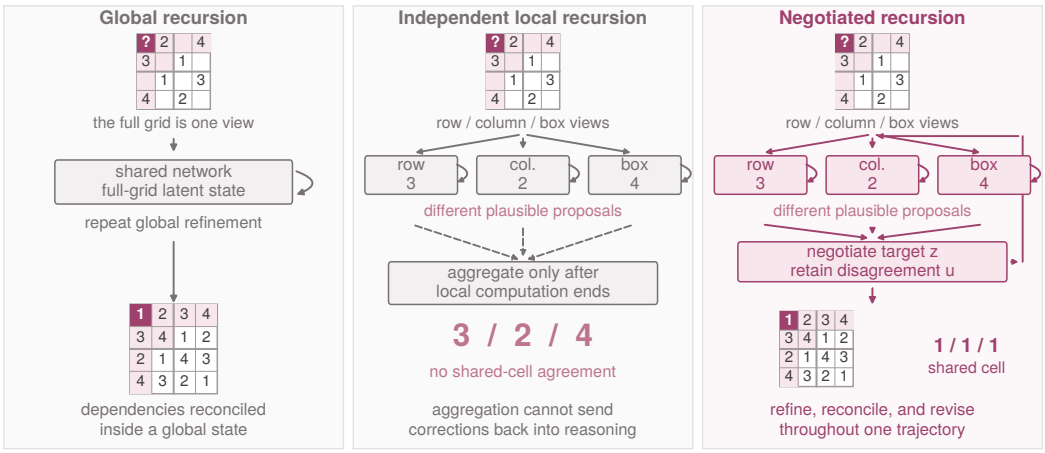
Many reasoning problems require a collection of decisions to satisfy several constraints at once. Completing a Sudoku grid requires every row, column, and box to be valid; finding a path through a maze requires local moves to form one globally connected route. A promising approach is to refine a candidate solution repeatedly with a small, shared network. Hierarchical Reasoning Models (HRM) and Tiny Recursive Models (TRM) demonstrate that this form of recursive computation can support strong structured reasoning, adding computation without adding parameters (Wang et al., 2025; Jolicoeur-Martineau, 2025). Their success raises a further question: how should repeated computation be organized when different parts of a problem must agree?

A Sudoku cell makes the coordination problem concrete. Its row, column, and box provide different subsets of the evidence needed to determine its value. A row-based proposal can be plausible within that row while conflicting with a column-based proposal for the same cell. A global recursive model must learn to represent and reconcile all such dependencies within its shared state. Alternatively, one can expose the structure by assigning a small recursive solver to each row, column, and box. Each such local view is a computational unit within one predictor, and views negotiate in latent space. This decomposition makes the relevant constraints explicit and permits parallel local computation, but it introduces multiple proposals for each shared cell. If the views recurse independently and vote only at the end, no local solver receives feedback with which to revise its reasoning in response to a conflict found by another view; in our benchmark setting, such end-of-run voting solves only 3.1% of Sudoku and 0.4% of Maze instances.

We therefore study *how to coordinate local recursive computation while it is taking place*, rather than only after it ends. Coordination during reasoning matters because additional local steps can refine a proposal without making it compatible with the rest of the problem's constraints. Effective coordination needs both a common target and a way to remember disagreements that have not yet been resolved. An instantaneous average provides a target, but can discard which local proposals

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.

(a) A shared cell must satisfy its row, column, and box



4×4 illustration; proposal changes are schematic.

(b) Global recursive baselines

	Sudoku	Maze
TRM	88.6 ± 0.9	86.5 ± 1.5
EqR, one trajectory	94.0 ± 0.5	89.8 ± 1.2
Global-RecurProx	93.6 ± 0.7	90.2 ± 1.4
NRR (ours)	**94.8 ± 0.5**	**90.1 ± 1.1**

(c) Accuracy and coordination cost

Task / width	Global	NRR
Sudoku	93.6 ± 0.7 22.0 ± 0.3 ms	**94.8 ± 0.5** 28.0 ± 0.4 ms
Maze 30	90.2 ± 1.4 52.0 ± 0.7 ms	**90.1 ± 1.1** 41.0 ± 0.7 ms
Maze 60	49.0 ± 3.1 245.0 ± 4.4 ms	**54.5 ± 3.0** 108.0 ± 2.1 ms
Maze 90	23.0 ± 3.8 750.0 ± 16.5 ms	**28.5 ± 3.6** 268.0 ± 5.9 ms

(b,c) Mean ± SD, 5 seeds. In (c), accuracy (%) above latency (ms). Panel (a) is schematic.

Figure 1: **Negotiation as an organization of recursive computation.** (a) Global recursion, independent local recursion, and repeated negotiation over a shared Sudoku cell. Proposal changes are schematic, not a recorded trajectory. (b) Five-seed mean ± SD for global recursive baselines in the controlled setting: native TRM, single-trajectory EqR, and Global-RecurProx (Section C). (c) Controlled-setting accuracy–latency comparison: five-seed mean ± SD for accuracy (above) and end-to-end batch-one A100 latency (below); "Global" denotes Global-RecurProx. NRR incurs overhead on Sudoku but improves the tradeoff on larger mazes. The controlled and benchmark settings are distinct (Section 5.1).

repeatedly depart from it. Carrying those departures forward allows subsequent local computation to respond to persistent conflicts. This motivates our hypothesis: when overlapping views expose informative constraints, recursive refinement benefits from an explicit, stateful negotiation mechanism.

Negotiated Recursive Reasoning (NRR) implements this idea by alternating private refinement and shared negotiation. Each view has a proposal and a private reasoning state, and all views share the weights of the same small recursive network. The network refines its proposal toward an anchor supplied by the preceding negotiation round. A consensus operator then compares representations of overlapping variables and constructs updated targets. Disagreement memory, represented by the dual state u, accumulates the remaining difference between each proposal and its target. The next round uses both, so local solvers preserve their own reasoning progress while adapting to evidence from other views.

Two choices make this organization practical. First, neighboring views need to agree only on what they share. A row and a box should reconcile their common cells without being forced to have identical representations of all their other cells. We express this compatibility through learned linear maps on an overlap graph, using a cellular-sheaf formulation. Second, negotiation must be both affordable and trainable within a finite recursive trajectory. We use a normalized soft-consensus system with a bounded condition number, and retain the two-round gradient path that connects one round's communication to the next round's supervised proposal. Building on Sheaf-ADMM's

coordination framework (Seely et al., 2026), NRR adds private recursion and supervision across negotiation rounds (Section F).

We evaluate NRR on Sudoku-Extreme and Maze-Hard in a benchmark setting and a controlled setting with validation-based selection (Section 5.1). In the benchmark setting, NRR achieves 95.2% and 90.4% exact solve, 5.3 percentage points above a version that replaces negotiation with a learned residual link while keeping the encoder and local reasoner. On Sudoku, the advantage grows with an external solver's backtracking count, and removing disagreement memory reduces accuracy to 89.5%. In the controlled setting, NRR exceeds a native TRM rerun by 6.2 and 3.6 points on Sudoku and Maze (Figure 1b), a persistent learned link by 1.1 and 0.5 points, and a recurrent relational network on Sudoku (94.8% versus 93.4%). Controlled experiments also show that disagreement memory improves both recursive and feed-forward local solvers, with the larger mean gain for recursion, and that degree-preserving overlap shuffling reduces retrained Sudoku accuracy from 94.8% to 86.0%.

With bounded views and overlaps, recursive work grows linearly with problem size at fixed local and communication budgets. On larger mazes, NRR gives a better accuracy–latency tradeoff than Global-RecurProx, which uses global attention recurrence (Figure 1c). Results on ARC and single-label MNIST further locate NRR's strongest use case in coordinating structured output decisions.

Our contributions are threefold:

- We formulate structured recursive reasoning as the joint problem of refining local proposals and reconciling their overlapping variables, making coordination an explicit design choice alongside recursion depth.
- We develop NRR, a shared recursive local solver coupled by consensus targets and disagreement memory, together with an analysis of finite communication accuracy, task-gradient connectivity, and computation under sparse overlap.
- We show improvements over learned-link, native TRM, and graph-based communication comparators on Sudoku-Extreme and Maze-Hard, examine the roles of recursion, disagreement memory, and overlap structure through controlled experiments, and characterize the resulting tradeoffs in problem size and latency.

2 From Structured Prediction to Local Negotiation

We define the prediction problem, compatibility between views, and the state used to coordinate them. Sudoku provides a running example; the formulation also applies to overlapping spatial patches.

2.1 One Output, Multiple Local Views

Let D denote the observed input and $Y = (Y_v)_{v \in \mathcal{V}}$ the output variables. A structured prediction problem can be written abstractly as

$$\widehat{Y}(D) \in \arg \min_{Y \in \mathcal{Y}(D)} \sum_{i=1}^{N} E_i(Y_{S_i}; D), \tag{1}$$

where $S_i \subseteq \mathcal{V}$ specifies a subset of variables, E_i describes local evidence or constraints, and $\mathcal{Y}(D)$ is the admissible output space. This formulation describes the task structure; NRR learns a predictor from (D, Y) pairs and does not require explicit numerical access to the energies E_i.

For Sudoku, $\mathcal{V}$ contains the 81 cells, $\mathcal{Y}(D)$ preserves the given clues, and the 27 sets S_i are the nine rows, columns, and boxes. The local constraints require the digits in each set to be distinct. Every cell belongs to three sets. If each local solver maintains its own copy $Y^{(i)}$ of the covered output, those copies must satisfy

$$Y_v^{(i)} = Y_v^{(j)} \quad \text{for every } v \in S_i \cap S_j. \tag{2}$$

Thus local constraint satisfaction and agreement between local copies are separate requirements. In a maze, the views are overlapping patches: a plausible path segment in one patch must connect to segments proposed by its neighbors.

A decomposition supplies the observed local inputs d_i, the sets S_i, and a graph $G = (V, E)$ connecting overlapping views. These structural views should be distinguished from the transformed evaluation inputs used for ARC. Each graph node is an *agent*: a stateful application of shared network weights, rather than a separately trained model. The decomposition is part of the model input and is constructed without the unknown target.

2.2 Compatibility Between Local Representations

Directly requiring two entire latent states to coincide would erase the distinction between their views. Instead, let $x_i \in \mathbb{R}^{d_v}$ represent a local proposal and let edge $e = (i, j)$ have a comparison space $\mathbb{R}^{d_e}$. Linear maps $F_{i\to e}$ and $F_{j\to e}$ select or transform the components to be compared. For a Sudoku row–column edge, both maps refer to the shared cell, even though it occupies different local positions in the two views. Stacking all discrepancies gives

$$(Fx)_e = F_{i\to e}x_i - F_{j\to e}x_j, \qquad \mathcal{C} = \ker F, \qquad L_{\mathcal{F}} = F^\top F. \tag{3}$$

This construction is a cellular sheaf on the graph (Hansen & Ghrist, 2019): local states can differ while their restricted representations agree. Fixed, position-aligned selectors implement literal overlap matching; learned maps allow the compared representation to adapt to the task. Agreement in this latent space is the coordination signal. Its relation to decoded answers will be examined in Section 4.

2.3 A State Structure for Repeated Coordination

To see how compatibility can guide a local update, consider the continuous template

$$\min_{x,z}\ f(x) + \chi_{\mathcal{C}}(z) \quad \text{subject to } x = z, \qquad f(x) = \sum_i f_i(x_i), \tag{4}$$

where f_i expresses a local preference and $\chi_{\mathcal{C}}$ is zero on $\mathcal{C}$ and infinite elsewhere. The separate variable z lets the alternating direction method of multipliers (ADMM) handle local preferences and compatibility in turn (Boyd et al., 2011):

$$\begin{aligned} x^{k+1} &= \mathrm{prox}_{f/\rho}(z^k - u^k), \\ z^{k+1} &= \Pi_{\mathcal{C}}(x^{k+1} + u^k), \\ u^{k+1} &= u^k + x^{k+1} - z^{k+1}. \end{aligned} \tag{5}$$

Here $\rho > 0$ and $\mathrm{prox}_{f/\rho}(a) = \arg\min_x [f(x) + \rho\|x - a\|^2/2]$. The three steps have complementary roles: propose a locally preferred state near an anchor, reconcile proposals, and retain their remaining discrepancy. In particular, u changes the next anchor instead of discarding the conflict after one averaging operation.

NRR uses this update organization with a learned local computation. A recursive solver needs a private state r_i to preserve intermediate reasoning that need not be communicated. The full state is therefore

$$h^k = (x^k, r^k, z^k, u^k), \qquad x_i, r_i, z_i, u_i \in \mathbb{R}^{d_v}.$$

Here x_i is the proposal, r_i the private reasoning state, z_i the consensus target, and u_i the accumulated proposal–target discrepancy. TRM motivates repeatedly updating private and answer states with the same small network (Jolicoeur-Martineau, 2025); NRR makes that local recursion responsive to an external negotiation anchor. The next section specifies the learned update and replaces exact projection with a finite soft-consensus solve.

3 Negotiated Recursive Reasoning

Each NRR round combines parallel local refinement, target negotiation, and discrepancy accumulation (Figure 2). Agents share the encoder, local reasoner, and decoder. For each new instance, initialize $x^0 = r^0 = z^0 = u^0 = 0$ and compute $e_i = \mathrm{Enc}_\theta(d_i) \in \mathbb{R}^{d_v}$, also conditioned on observed demonstration pairs for ARC. Numerical and gradient defaults below specify the controlled configuration; benchmark budgets appear in Table 3.

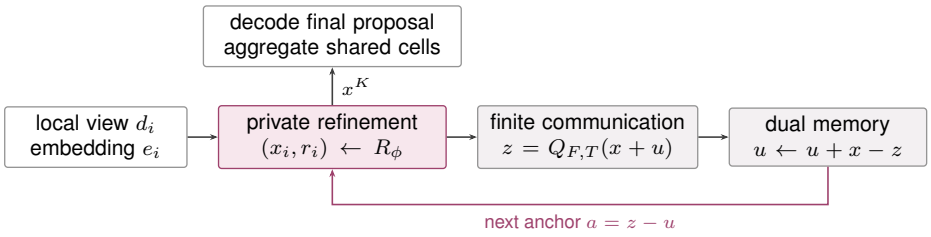


Figure 2: **One NRR round.** The local solver updates a proposal x_i and private state r_i. Communication constructs targets z_i, while the dual state u_i (disagreement memory; "dual memory" in the diagram) retains proposal–target discrepancies. Their combination $z_i - u_i$ guides the next local update. Final predictions are decoded from proposals and aggregated at shared output coordinates.

3.1 Private Recursion Guided by a Negotiated Anchor

The local solver must combine what its own view suggests with what the surrounding views currently require. We supply the latter through the anchor $a_i^k = z_i^k - u_i^k$. A shared network f_ϕ first performs n private reasoning updates and then revises the proposal:

$$r_i \leftarrow f_\phi(e_i + x_i + r_i + a_i^k) \qquad (n \text{ private updates}), \tag{6}$$

$$x_i \leftarrow f_\phi(x_i + r_i + \rho a_i^k) \qquad (\text{one proposal update}). \tag{7}$$

Starting from (x_i^k, r_i^k), this defines $(x_i^{k+1}, r_i^{k+1}) = R_\phi(a_i^k, e_i; x_i^k, r_i^k)$, which we call RecurProx. The private state persists across rounds, so the solver can build on previous reasoning while responding to a changing anchor. The network is shared across both agents and recursive applications; task-specific tokenization and fixed side inputs are detailed in Sections B.4 and D.2.1.

The anchor has the same role as the center of the proximal subproblem in (5). Unlike an exact proximal map, however, RecurProx is trained through supervised prediction and can depend on its history. This permits a richer local computation while retaining an interpretable interface between local reasoning and coordination.

3.2 Soft Consensus and Persistent Disagreement

After private refinement, negotiation balances the current proposals against overlap compatibility. For $w = x^{k+1} + u^k$, the exact soft-consensus target solves

$$z_* = \arg\min_z \frac{1}{2}\|z - w\|^2 + \frac{\beta}{2}\|Fz\|^2 = (I + \beta F^\top F)^{-1} w. \tag{8}$$

The first term preserves the information carried by local proposals; the second reduces disagreement on their overlaps. Finite-strength consensus is useful because learned compatibility maps need not define a well-conditioned, nontrivial hard-consensus space. We approximate this target using T conjugate-gradient (CG) steps, denoted $Q_{F,T}$, and complete the round with

$$\begin{aligned} w^{k+1} &= x^{k+1} + u^k, \\ z^{k+1} &= Q_{F,T}(w^{k+1}), \\ u^{k+1} &= u^k + x^{k+1} - z^{k+1}. \end{aligned} \tag{9}$$

The dual state is a running memory of the difference between a proposal and its negotiated target. Starting from zero, $u^k = \sum_{t=1}^k (x^t - z^t)$. Thus a recurring discrepancy keeps influencing the anchor, rather than being forgotten after each communication step. This gives a direct experimental comparison: retaining the same local reasoner while setting $u = 0$ tests the value of disagreement memory.

Learned compatibility with controlled scale. An edge-type base map and rank-four content modulation give raw endpoint maps $\widetilde{F}_{i\to e} = F_e^0 + U_i V_i^\top$ (dimensions in Section B.1). Normalize the stacked signed operator by

$$H_i = \sum_{e \ni i} \widetilde{F}_{i\to e}^\top \widetilde{F}_{i\to e}, \quad b = 2 \max_i \|H_i\|_\infty, \quad F = \widetilde{F}/\sqrt{b + 10^{-6}}, \tag{10}$$

where $\|H\|_\infty$ is the maximum absolute row sum. This ensures $\|F\|_2^2 \le 1$. The maps depend only on observed inputs and remain fixed within a parameter-frozen trajectory segment; derivatives are retained through their construction and normalization.

The controlled configuration uses $\beta = 3$ and unpreconditioned CG on

$$Az = w, \qquad A = I + 3F^\top F, \qquad z_0 = 0. \tag{11}$$

We use $T = 5$ for Sudoku, Maze, and ARC, and $T = 3$ for MNIST, with float32 communication and reductions. Because $I \preceq A \preceq 4I$, the system has condition number at most four. Every executed CG operation, including its coefficients, participates in backpropagation. Section B.5 specifies numerical exits and failure handling. Hard projection and finite diffusion provide alternative communication controls.

3.3 Readout and Learning Across Rounds

Training supervises decoded proposals. For output location v covered by view i, let $\ell_{iv} = \mathrm{Dec}_\psi(d_i, x_i^K)_v$ and $\bar{\ell}_v$ be the mean over covering views. Inference takes the highest-scoring class, breaking ties by class index; training differentiates the loss on $\bar{\ell}$, before argmax. Communication ablations share this reducer. ARC additionally predicts output dimensions and uses demonstration context (Section D.2.1).

Training begins with local warm-start supervision from perturbed anchors, followed by coupled training over M segments of K rounds. Forward states carry between segments, allowing later segments to refine earlier proposals. The segment losses are summarized by

$$\mathcal{L} = \frac{1}{M}\sum_{m=1}^{M}\left[\mathcal{L}_{\mathrm{task}}(\bar{\ell}^{(m)}, Y) + \lambda\frac{\|Fx^{(m),K}\|^2}{|E|d_e}\right]. \tag{12}$$

Here $\mathcal{L}_{\mathrm{task}}$ is StableMax cross-entropy; for ARC its arguments additionally include shape logits and its loss includes the shape term in Section D.2.1. The default uses $\rho = 1$ and $\lambda = 0$; the optional residual penalty is omitted for an empty graph. An optimizer update follows each segment, so the expression summarizes losses without differentiating through optimizer updates (Section B.6).

Why the retained window must span two rounds. Let B be the number of complete rounds retained for backpropagation, with earlier states detached. The final proposal x^K is computed *before* the final communication step. Consequently, with $B = 1$, parameters used only by that communication operator have no task-gradient path to the proposal readout. Retaining $B = 2$ provides

$$F \longrightarrow (z^{K-1}, u^{K-1}) \longrightarrow a^{K-1} \longrightarrow x^K \longrightarrow \mathcal{L}_{\mathrm{task}}.$$

This is the default retained window in the controlled configuration. Each retained round includes all n private updates, the proposal update, and the finite communication solve. The final communication still updates the carried state. Reading z^K instead yields a different, one-round-connected control. Section A.6 derives these paths, and Figure 7b compares the resulting training choices.

3.4 Three Computational Budgets

NRR allocates computation at three levels: n private updates per round, T numerical communication steps, and R rounds at inference. Training uses K rounds per segment and up to M segments, so a carried training state can reach KM rounds. These counts distinguish additional local refinement from additional opportunities to exchange information.

For q_i local tokens of width h, a fixed-layer attention reasoner requires $O((n+1)\sum_i(q_i^2 h + q_i h^2))$ work per round. Dense edge restrictions contribute $O(T|E|d_e d_v)$, together with CG vector operations and global reductions. With bounded patch size, graph degree, and overlap coverage, this recursive work is linear in input size at fixed widths and budgets. The proof and the vector-MLP case appear in Section A.8. End-to-end cost also includes encoding, map normalization, decoding, and any transformed evaluations. Experiments therefore compare task quality and measured latency, with local computation and communication described separately for each baseline.

4 What Finite Negotiation Guarantees

Normalized soft consensus provides a controlled numerical interface to the learned reasoner. For $A = I + 3F^\top F$, $I \preceq A \preceq 4I$; starting from zero, T exact-arithmetic CG steps give scaled solution-error and linear-residual bounds $2/3^T$ and $4/3^T$, respectively (Proposition A.2). The exact soft operator has eigenvalues in $[1/4, 1]$, preserving information even when hard projection onto a full-rank learned map's kernel would collapse to zero.

Numerical accuracy, latent compatibility, and answer correctness are distinct. The feasibility bound in Section A.1 separates proposal–target mismatch, finite-strength consensus bias, and numerical error. More solver steps reduce only the last term. The exact communication split is non-expansive in its input, but this does not establish stability of the complete learned recurrence or of the finite-CG Jacobian. Likewise, small latent residuals need not imply a correct decoded answer. Section A gives the statements, proofs, decoder conditions, gradient-window bounds, and exact-convex special case; the experiments evaluate these distinctions directly.

5 Experiments

We test whether organizing recursion around overlapping views improves whole-instance reasoning, what negotiation adds beyond learned communication, and when sparse coordination repays its overhead.

5.1 Tasks and Comparison Settings

Sudoku-Extreme has 1000 training puzzles with rule-preserving augmentation and 422786 test puzzles. Maze-Hard has 1000 training and 1000 test 30×30 mazes with shortest paths longer than 110 (Wang et al., 2025). We report exact agreement with the entire target grid or reference path mask; another valid shortest path does not count as an exact reference match. ARC-AGI-1/2 require inferring a grid transformation from demonstrations and predicting both output shape and content (Chollet, 2019; Chollet et al., 2025); public-evaluation task success allows two candidates per query.

The *benchmark* setting uses Table 3. The *controlled* setting uses grouped 900/100 training/validation splits for Sudoku/Maze, normalized consensus, and two retained rounds, with hyperparameters and checkpoints selected on validation data. NRR scores 94.8%/90.1% in this setting, versus 95.2%/90.4% in the benchmark setting. Contrasts stay within a setting. Results report mean $\pm$ sample standard deviation across five seeds (42, 123, 456, 789, 1024), with contrasts paired within seed. Sections D, D.1 and D.4 detail tasks, aggregation, and evaluation.

Baselines and interventions. Sheaf-ADMM uses local proximal solves; recurrent message-passing neural networks (MPNNs) learn graph updates. Native TRM and EqR use global recurrence, while Global-RecurProx applies our local-update family with whole-grid attention. NRR's learned-link replacement substitutes residual messages for negotiation; no-dual sets $u = 0$, removing disagreement memory, and shared maps remove content modulation. The recurrent relational network and persistent learned link provide additional graph-based comparisons, with their state and update interfaces specified in Sections C.3 and C.4. Sections C, D.2 and D.9 define the methods and ablation families; published comparisons are separate (Section F.2).

5.2 Can Existing Recursive Models Solve These Tasks Directly?

Figure 1b compares native TRM (Jolicoeur-Martineau, 2025), single-trajectory EqR (Huang et al., 2026), and Global-RecurProx. In this controlled comparison, NRR improves over TRM by 6.2/3.6 points on Sudoku/Maze and over EqR by 0.8/0.3 points. Global-RecurProx is competitive and slightly higher on Maze (90.2% versus 90.1%).

The practical distinction grows with problem size. In the controlled setting, at maze width 90, NRR achieves 28.5% accuracy at 268 ms, compared with 23.0% at 750 ms for Global-RecurProx, a $2.80\times$ latency ratio (Figure 1c). Width 30 is nearly tied in accuracy, while Sudoku incurs coordination overhead. These paired, batch-one A100 measurements include encoding and decoding. In the

separate benchmark size-transfer evaluation, NRR achieves 54.5% and 27.8% accuracy at widths 60 and 90, versus 19.7% and 3.6% for TRM; Section E.6 also gives the benchmark timing series.

5.3 What Does Explicit Negotiation Add?

Table 1: **Within-study results (%).** Sudoku and Maze use whole-instance exact solve; ARC uses two-attempt public evaluation. NRR uses 1000 transformed inputs. These are component comparisons in the benchmark configuration; the additional baseline and ablation comparisons use the grouped split in Section D.3. Entries are mean ± SD over five seeds.

Method	Sudoku	Maze	ARC-1	ARC-2
Sheaf-ADMM rerun	71.6 ± 0.9	64.8 ± 2.6	34.8 ± 1.9	5.6 ± 0.7
Recurrent MPNN rerun	42.5 ± 3.2	83.2 ± 2.3	45.2 ± 2.1	10.8 ± 0.9
NRR with learned link	89.9 ± 0.9	85.1 ± 1.6	48.2 ± 1.8	11.6 ± 0.8
NRR without dual	89.5 ± 1.1	85.8 ± 1.8	47.5 ± 2.0	11.0 ± 0.9
NRR, shared maps only	94.8 ± 0.4	87.8 ± 1.3	50.9 ± 1.6	13.6 ± 0.6
NRR	95.2 ± 0.5	90.4 ± 1.0	52.6 ± 1.6	14.4 ± 0.7

Table 1 holds the encoder and recursive local solver fixed for the learned-link replacement. NRR gains 5.3 points on both Sudoku and Maze, removing 52.5% and 35.6% of the remaining errors. ARC gains are 4.4/2.8 points at 1000 transformed inputs. Independent local recursion followed by all-view voting solves only 3.1%/0.4% of Sudoku/Maze, so decomposition alone is insufficient (Figure 5). On Sudoku, the advantage over the learned-link replacement grows from 1.2 to 13.0 points across bins of an external solver's backtracking count (Figure 6).

Comparison with additional communication baselines. The controlled setting adds stronger graph-based communication comparators (Figure 7a). A persistent learned link with five communication substeps per round reaches 93.7%/89.6% on Sudoku/Maze, versus NRR's 94.8%/90.1%, a gain of 1.1 and 0.5 points. Aligned fixed-overlap consensus reaches 93.8%/89.4%, and a recurrent relational network reaches 93.4%/89.2%. The benchmark learned-link comparison yields the larger 5.3-point gain on both tasks.

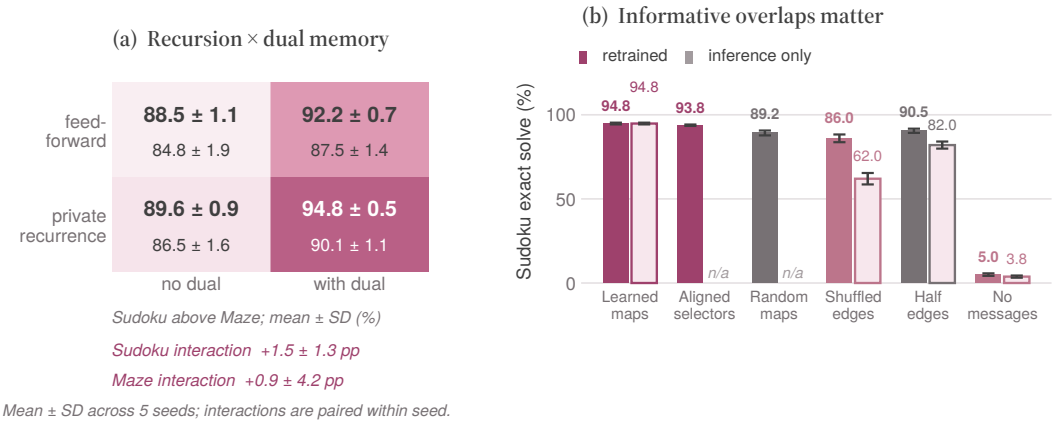


Figure 3: **Why local recursion benefits from negotiation.** (a) Factorial over local recursion and disagreement memory ("dual memory", the dual state u), with the local-solver variants defined in Section C.5. Cells give mean ± SD for Sudoku above Maze. (b) Sudoku graph controls: retraining under an altered graph versus intervening only at inference. Whiskers show ±1 SD over five seeds. Map replacements apply only to retraining. All results use the controlled setting.

Contributions of recursion and disagreement memory. Figure 3a replaces within-round recursion with a feed-forward local solver, which uses untied factorized weights, while retaining its persistent state capacity. Disagreement memory ("dual memory" in the figure) adds 3.7/2.7 points to

feed-forward solving and 5.2/3.6 to recursive solving on Sudoku/Maze. The paired difference-in-differences is $1.5 \pm 1.3/0.9 \pm 4.2$ points. Memory improves both solvers on average; the mean gain is larger with recursion. Section E.2 gives the complete contrasts and their interpretation.

The graph supplies an informative inductive bias. Degree-preserving overlap shuffling reduces retrained Sudoku accuracy from 94.8% to 86.0%; an inference-only shuffle reduces it to 62.0% (Figure 3b). Keeping half the edges gives 90.5% and 82.0%, respectively. Retraining tests adaptation to the structure; inference intervention tests reliance on the learned interface. Section E.2 details these controls.

Learning must reach the negotiation step. With proposal readout, a one-round gradient window trains the local solver but leaves coupling-only parameters without task supervision; Sudoku accuracy is 92.4%. Retaining two rounds restores this path and reaches 94.8%. The readout and longer-window controls in Figure 7b connect the analytical gradient argument to measured performance.

Scope and further evidence. Sections E.6 to E.8 expand the depth, numerical-budget, localization, and stopping analyses. In the controlled setting, residual-based stopping uses 18.5 Sudoku rounds on average instead of 24, while accuracy changes from 94.8% to 94.5%. Section E.9 examines ARC and single-label MNIST: ARC experiments show strong dependence on demonstrations and a substantial remaining content-prediction gap, and on MNIST an equally augmented CNN matches or exceeds NRR under the tested conditions. These extensions locate NRR's strongest use case in coordinating structured output decisions.

6 Discussion and Conclusion

NRR treats coordination as part of recursive reasoning itself. Its local solvers preserve private computation, negotiate targets over shared variables, and carry unresolved disagreement forward. In the benchmark setting, NRR improves whole-instance accuracy on Sudoku and Maze by 5.3 points over a learned-link replacement; in the controlled setting, it also exceeds a persistent learned link and a recurrent relational network, by smaller margins. Controlled factorial and topology experiments connect the gains to private recursion, disagreement memory, and meaningful overlaps. On larger mazes, sparse coordination also improves the accuracy–latency tradeoff.

The analysis explains how to implement the mechanism with finite computation: normalized soft consensus controls numerical conditioning, and a retained two-round path allows proposal supervision to train communication. The exposed residuals help localize errors and reduce recursive work, while exact task validity remains a distinct property. Together, these findings support a view of recursive models in which additional computation is organized around both refinement and reconciliation.

Limitations and next steps. NRR assumes an informative decomposition and overlap graph; automatically discovering useful views is an important extension. Coordination incurs overhead on small inputs, and neither increased depth nor numerical precision guarantees a better decoded answer. Learned, stateful local updates also lie outside general convex-ADMM convergence guarantees. Current experiments emphasize small structured models, and ARC results show strong dependence on demonstrations and a substantial remaining content-prediction gap. The stronger graph and global-recursion baselines are competitive, with some margins below one percentage point, so paired seed-level dispersion matters for those differences. Extending negotiation to learned decompositions, larger pretrained reasoners, and compute-adaptive interaction offers a route to testing the idea beyond the present setting.

Reproducibility Statement

Section 3 specifies the state updates, soft-consensus system, and retained gradient window; Section A provides the proofs. Sections B and C detail graph construction, training execution, and baseline interfaces; Section D specifies task conditioning, selection, and evaluation populations.

Section B.2 summarizes the benchmark and controlled configurations, and Section B.7 describes gradient and input-dependency checks.

AI USE STATEMENT

Generative AI assisted with manuscript editing, figure design, and consistency checks.

REFERENCES

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. *Advances in Neural Information Processing Systems*, 32, 2019.

Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pp. 136–145, 2017.

Artur Back de Luca and Kimon Fountoulakis. Simulation of graph algorithms with looped transformers. In *International Conference on Machine Learning*, pp. 2319–2363, 2024.

Junyeob Baek, Mingyu Jo, Minsu Kim, Mengye Ren, Yoshua Bengio, and Sungjin Ahn. Generative recursive reasoning. *arXiv preprint arXiv:2605.19376*, 2026. GRAM.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, 2019. arXiv:1909.01377.

Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end algorithm synthesis with recurrent networks: Extrapolation without overthinking. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20232–20242, 2022.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009. doi: 10.1137/080716542.

Hugh Blayney, Álvaro Arroyo, Johan Obando-Ceron, Pablo Samuel Castro, Aaron Courville, Michael M. Bronstein, and Xiaowen Dong. A mechanistic analysis of looped reasoning language models. *arXiv preprint arXiv:2604.11791*, 2026.

Cristian Bodnar, Francesco Di Giovanni, Benjamin Paul Chamberlain, Pietro Liò, and Michael M. Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in GNNs. In *Advances in Neural Information Processing Systems*, 2022.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

Gregery T. Buzzard, Stanley H. Chan, Suhas Sreehari, and Charles A. Bouman. Plug-and-play unplugged: Optimization-free reconstruction using consensus equilibrium. *SIAM Journal on Imaging Sciences*, 11(3):2001–2020, 2018. doi: 10.1137/17M1122451.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

François Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. ARC-AGI-2: A new challenge for frontier AI reasoning systems. *arXiv preprint arXiv:2505.11831*, 2025.

Justin Michael Curry. *Sheaves, Cosheaves and Applications*. PhD thesis, University of Pennsylvania, 2014.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. arXiv:1807.03819.

Mariia Drozdova, Aidan Sirbu, Pietro Miotti, Robert Obryk, Mayalen Etcheverry, Eyvind Niklasson, and Blake Richards. Diffusion as a training curriculum for timestep-free iterative reasoning, 2026. URL https://arxiv.org/abs/2609.01449.

Katie E. Everett, Lechao Xiao, Mitchell Wortsman, Alexander A. Alemi, Roman Novak, Peter J. Liu, Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, and Jeffrey Pennington. Scaling exponents across parameterizations and optimizers. In *International Conference on Machine Learning*, 2024.

Richard Freinschlag, Timo Bertram, Erich Kobler, Andreas Mayr, and Günter Klambauer. Symbol-equivariant recurrent reasoning models, 2026. URL https://arxiv.org/abs/2603.02193.

Zitian Gao, Lynx Chen, Yihao Xiao, He Xing, Ran Tao, Haoming Luo, Joey Zhou, and Bryan Dai. Universal reasoning model. *arXiv preprint arXiv:2512.14693*, 2025.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272, 2017.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *International Conference on Machine Learning*, pp. 399–406, 2010.

Tyler Hanks, Hans Riess, Samuel Cohen, Trevor Gross, Matthew Hale, and James Fairbanks. Distributed multi-agent coordination over cellular sheaves. In *IEEE 64th Conference on Decision and Control (CDC)*, pp. 3057–3064. IEEE, 2025.

Jakob Hansen and Robert Ghrist. Toward a spectral theory of cellular sheaves. *Journal of Applied and Computational Topology*, 3(4):315–358, 2019.

Jakob Hansen and Robert Ghrist. Opinion dynamics on discourse sheaves. *SIAM Journal on Applied Mathematics*, 81(5):2033–2060, 2021.

Alec Helbling, Andrey Bryutkin, Mauro Martino, Duen Horng Chau, Nima Dehmamy, and Hendrik Strobelt. Flow reasoning models: Turning flows into efficient recurrent reasoners. arXiv preprint arXiv:2606.29150v3, 2026a. URL https://arxiv.org/abs/2606.29150v3. Version 3.

Alec Helbling, Andrey Bryutkin, Mauro Martino, Nima Dehmamy, and Hendrik Strobelt. Flow reasoning models: Scaling reasoning through iterative self-refinement. arXiv preprint arXiv:2606.29150v1, 2026b. URL https://arxiv.org/abs/2606.29150v1. Version 1.

Lintai Hou. Graph machine: Exploring edge mechanisms as an inductive bias, 2026. URL https://arxiv.org/abs/2608.06834.

Benhao Huang, Zhengyang Geng, and J. Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning. *arXiv preprint arXiv:2605.21488*, 2026.

Zekai Huang, Yingyu Liang, Zhenmei Shi, Zhao Song, and Zhen Zhuang. Neural algorithmic reasoning for hypergraphs with looped transformers. *arXiv preprint arXiv:2501.10688*, 2025.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025.

Shunsuke Kamiya, Masanori Koyama, Seongcheol Jeong, Fumiya Uchiyama, Kenji Kubo, Kohei Hayashi, Masahiro Suzuki, and Yutaka Matsuo. Steering recurrent reasoners at inference time with readout feedback, 2026. URL https://arxiv.org/abs/2608.24136.

Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001. doi: 10.1109/18.910572.

Yann LeCun, Corinna Cortes, and Christopher J. C. Burges. The MNIST database of handwritten digits. Dataset website, n.d. URL https://yann.lecun.org/exdb/mnist/index.html.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *International Conference on Learning Representations*, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101. arXiv:1711.05101.

Omid Nejati Manzari, Guillaume Lajoie, and Hassan Rivaz. Recursive vision language models for general symbolic reasoning, 2026. URL https://arxiv.org/abs/2608.01534.

Sean McLeish, Ang Li, John Kirchenbauer, Dayal Singh Kalra, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Jonas Geiping, Tom Goldstein, and Micah Goldblum. Teaching pretrained language models to think deeper with retrofitted recurrence. *arXiv preprint arXiv:2511.07384*, 2025.

Vishal Monga, Yuelong Li, and Yonina C. Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2):18–44, 2021.

Arseny Moskvichev, Victor Vikram Odouard, and Melanie Mitchell. The ConceptARC benchmark: Evaluating understanding and generalization in the ARC domain. *arXiv preprint arXiv:2305.07141*, 2023.

Sajad Movahedi, Vera Milovanović, Shlomo Libo Feigin, Alexander Theus, Thomas Hofmann, Valentina Boeva, T. Konstantin Rusch, and Antonio Orvieto. Fixed-point reasoners: Stable and adaptive deep looped transformers, 2026. URL https://arxiv.org/abs/2606.18206.

Rasmus Berg Palm, Ulrich Paquet, and Ole Winther. Recurrent relational networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018. arXiv:1711.08028.

Lucas Prieto, Melih Barsbey, Pedro A. M. Mediano, and Tolga Birdal. Grokking at the edge of numerical stability. In *International Conference on Learning Representations*, 2025.

Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. Can you learn an algorithm? Generalizing from easy to hard problems with recurrent networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 6695–6706, 2021.

Jeffrey Seely, Bartłomiej Cupiał, and Llion Jones. Learning multi-agent coordination via Sheaf-ADMM. *arXiv preprint arXiv:2605.31005*, 2026. URL https://arxiv.org/abs/2605.31005. Accepted at ICML 2026.

Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. Learning a SAT solver from single-bit supervision. In *International Conference on Learning Representations*, 2019. arXiv:1802.03685.

Amin Sghaier, Ali Parviz, and Alexia Jolicoeur-Martineau. Probabilistic tiny recursive model. *arXiv preprint arXiv:2605.19943*, 2026.

David Sontag, Amir Globerson, and Tommi Jaakkola. Introduction to dual decomposition for inference. In *Optimization for Machine Learning*. MIT Press, 2011.

Singanallur V. Venkatakrishnan, Charles A. Bouman, and Brendt Wohlberg. Plug-and-play priors for model based reconstruction. In *IEEE Global Conference on Signal and Information Processing*, pp. 945–948, 2013. doi: 10.1109/GlobalSIP.2013.6737048.

Ivan Viakhirev, Kirill Borodin, Amirah Almutairi, Serguei Barannikov, Maxim Abramov, and Grach Mkrtchian. Think shallow, solve deep: Controlling recurrent dynamics for reliable test-time depth, 2026. URL https://arxiv.org/abs/2608.18222.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025.

Po-Wei Wang, Priya L. Donti, Bryan Wilder, and J. Zico Kolter. SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, pp. 6545–6554, 2019.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *International Conference on Learning Representations*, 2024. arXiv:2311.12424.

Xiyuan Yang, 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:2604.25917v1*, 2026. URL https://arxiv.org/abs/2604.25917v1. RecursiveMAS; version 1.

Yifei Zhao and Ying Tang. Energy-guided recursive model, 2026. URL https://arxiv.org/abs/2607.10128.

Yujia Zheng, Zhuokai Zhao, Zijian Li, Yaqi Xie, Mingze Gao, Lizhu Zhang, and Kun Zhang. Thought communication in multiagent collaboration. *arXiv preprint arXiv:2510.20733*, 2025.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025. LoopLM.

Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, 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. *arXiv preprint arXiv:2511.20639v1*, 2025. URL https://arxiv.org/abs/2511.20639v1. LatentMAS; version 1.

A Mathematical Analysis and Proofs

This appendix states and proves the results behind the method's design. We begin with the soft-consensus step that NRR uses. It bounds how far the proposals are from full agreement by three terms: the gap between the proposals and the consensus target, the bias from finite consensus strength, and the numerical error of the linear solve (Section A.1). The normalized system makes this solve accurate within a small CG budget (Section A.2). We then turn to control operators and special cases: finite diffusion approaches the hard projection at a geometric rate (Section A.3), exact proximal steps give the fixed points an optimization interpretation (Section A.4), and a fixed communication map splits the state stably (Section A.5). Next, we bound the gradient that reaches the parameters through recursive rounds (Section A.6). Finally, we relate latent agreement to decoded answers (Section A.7) and give the per-round cost (Section A.8). Unless subscripted otherwise, norms are Euclidean norms or their induced operator norms. The graph and F are held fixed in the operator analysis, and we write $L = F^\top F$ and $P = \Pi_{\ker F}$.

A.1 Soft consensus and numerical residuals

For $\beta > 0$, $Q_\beta = (I + \beta L)^{-1}$ is the proximal map of $\beta\|Fz\|^2/2$. It is symmetric with spectrum in $(0, 1]$. At a fixed point $x = z = Q_\beta(x + u)$, multiplication by $I + \beta L$ gives $u = \beta L x$. With an exact local proximal map this implies

$$0 \in \partial f(x) + \rho\beta L x,$$

the optimality condition of the convex penalized objective rather than of the hard-constrained program in Section A.4. The following bound separates the three sources of remaining disagreement.

Proposition A.1 (Soft-consensus feasibility). *Suppose $L \neq 0$ has smallest positive eigenvalue λ_+. Let $w = x + u$, and let z_T approximate $Q_\beta w$ with linear-system residual $\varrho_T = w - (I + \beta L)z_T$. Then*

$$\operatorname{dist}(x, \ker F) \le \underbrace{\|x - z_T\|}_{\text{distance to target}} + \underbrace{\frac{\|x + u\|}{1 + \beta\lambda_+}}_{\text{soft-consensus bias}} + \underbrace{\|\varrho_T\|}_{\text{numerical error}}. \tag{13}$$

Proof. Because Q_β and P are simultaneously diagonalizable and Q_β acts as the identity on $\ker F = \ker L$, we have $(I - P)Q_\beta = Q_\beta - P$ and $\|Q_\beta - P\|_2 = (1 + \beta\lambda_+)^{-1}$. Moreover, $z_T - Q_\beta w = -(I + \beta L)^{-1}\varrho_T$, so $\|z_T - Q_\beta w\| \le \|\varrho_T\|$. Writing $x = (x - z_T) + Q_\beta w + (z_T - Q_\beta w)$ and applying the contraction $I - P$ gives

$$\|(I - P)x\| \le \|x - z_T\| + \|(Q_\beta - P)w\| + \|z_T - Q_\beta w\|,$$

which is (13). □

Increasing T reduces the numerical term; increasing β strengthens compatibility but changes the target and worsens conditioning. These distinct interventions motivate separate solver-budget and consensus-strength experiments (Figures 8 and 10).

A.2 Conditioning and finite-CG accuracy of the default

Proposition A.2 (Finite-CG accuracy). *For the normalized F and $A = I + 3F^\top F$, $I \preceq A \preceq 4I$. Let $z_* = A^{-1}w$. Starting from zero, exact-arithmetic CG after T iterations satisfies*

$$\frac{\|z_T - z_*\|}{\max(1, \|w\|)} \le \frac{2}{3^T}, \qquad \frac{\|w - Az_T\|}{\max(1, \|w\|)} \le \frac{4}{3^T}. \tag{14}$$

If CG reaches the exact solution earlier, subsequent iterates are taken to be unchanged.

Proof of Proposition A.2. We first bound the learned operator's scale and then apply the CG polynomial error bound, which connects the map normalization directly to the communication budget. For the raw endpoint maps in (10),

$$\|\widetilde{F}x\|^2 \le 2\sum_i x_i^\top H_i x_i \le 2\max_i \lambda_{\max}(H_i)\|x\|^2 \le b\|x\|^2.$$

The last inequality follows because a symmetric matrix has spectral radius at most its maximum absolute row sum. Consequently $0 \preceq F^\top F \preceq I$, $I \preceq A \preceq 4I$ and $\kappa_2(A) \le 4$. The normalization bound holds for arbitrary raw restriction maps, including rank-deficient and zero maps. At a zero map $A = I$. Maximum and absolute-value operations are piecewise differentiable; numerical derivative checks avoid their branch boundaries. Forming H_i costs $O(|E|d_e d_v^2)$ once per input/parameter version, and must be included in encoding cost.

Let $z_* = A^{-1}w$, $z_0 = 0$ and let z_T be the exact-arithmetic CG iterate before a tolerance exit. Its energy-norm error obeys

$$\frac{\|z_T - z_*\|_A}{\|z_*\|_A} \le 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^T \le \frac{2}{3^T}, \qquad w \neq 0. \tag{15}$$

Indeed, CG minimizes the energy error over its Krylov space. For every degree-T polynomial p with $p(0) = 1$, this error ratio is at most $\max_{t\in[1,4]} |p(t)|$. The rescaled Chebyshev polynomial $p(t) = C_T((5-2t)/3)/C_T(5/3)$ gives at most $2/3^T$, where C_T is the Chebyshev polynomial and $C_T(5/3) = (3^T + 3^{-T})/2$. The zero right-hand side returns zero without a division. Furthermore, $\|z_*\|_A^2 = w^\top A^{-1} w \le \|w\|^2$, so the solution-error bound in (14) follows. Also $\|A(z_T - z_*)\| \le 2\|z_T - z_*\|_A$, giving the residual bound there. □

At five steps these bounds are approximately 0.00823 and 0.01646, respectively; at three steps the residual bound is 0.14815. Early tolerance exits replace the corresponding residual bound by 10^{-6} in exact arithmetic. The float32 accuracy check uses recomputed true residuals. These bounds concern the soft solution and do not establish hard feasibility, non-expansiveness of the finite-CG Jacobian, or a task-accuracy guarantee. The exact soft map has eigenvalues in $[1/4, 1]$, avoiding the collapse of a full-column-rank hard projector while retaining an explicit, limited strength of agreement.

A.3 Finite diffusion error and approximate feasibility

Theorem A.3 (Finite communication and feasibility)**.** *Suppose $L \neq 0$ has smallest positive eigenvalue λ_+ and largest eigenvalue $\lambda_{\max}$. For $0 < \eta \le 1/\lambda_{\max}$ and $T \ge 1$, let $S_T = (I - \eta L)^T$. Then*

$$\|S_T - P\|_2 = (1 - \eta\lambda_+)^T =: \epsilon_T. \tag{16}$$

If $z = S_T(x + u)$, then

$$\operatorname{dist}(x, \ker F) \le \|x - z\| + \epsilon_T \|x + u\|. \tag{17}$$

For $L = 0$, $S_T = P = I$ and the distance is zero.

A sufficient budget for operator error at most ε is $T \ge (\eta\lambda_+)^{-1}\log(1/\varepsilon)$. Weakly connected compatibility directions can therefore require more communication, even at equal local-network budgets.

Proof of Theorem A.3. On $\ker L$, both S_T and P act as the identity. On an eigenvector of L with eigenvalue $\lambda > 0$, P is zero and S_T has eigenvalue $(1 - \eta\lambda)^T$. Under the step-size condition these eigenvalues are nonnegative and largest at λ_+. This proves (16). Since $(I - P)S_T = S_T - P$,

$$\|(I - P)x\| \le \|(I - P)(x - z)\| + \|(S_T - P)(x + u)\| \le \|x - z\| + \epsilon_T\|x + u\|.$$

Finally, $(1 - a)^T \le e^{-aT}$ for $0 < a \le 1$ proves the sufficient iteration count. No boundedness of u follows from this proof; a large dual state can make a small operator error consequential. □

A finite-step fixed point need not be feasible. Take $F = [1, -1]$, $\eta = 1/4$, $T = 1$, $\rho = 1$, and $f(x) = \|x - b\|^2/2$ with $b = (1, -1)^\top$. Then

$$S = \begin{bmatrix} 3/4 & 1/4 \\ 1/4 & 3/4 \end{bmatrix}, \qquad x^* = z^* = u^* = b/2.$$

The local update gives $\operatorname{prox}_f(z^* - u^*) = b/2$, the communication update gives $S(x^* + u^*) = Sb = b/2$, and the dual update is unchanged. Nevertheless, $Fx^* = 1$. This example satisfies the stable-diffusion condition and uses an exact convex proximal map.

A computable step-size bound. For block restrictions $F_{i\to e}$ define

$$b_F = 2 \max_i \sum_{e \ni i} \|F_{i\to e}\|_2^2.$$

The inequality $\|a-b\|^2 \le 2\|a\|^2 + 2\|b\|^2$ gives

$$\|Fx\|^2 \le 2 \sum_i \sum_{e \ni i} \|F_{i\to e}\|_2^2 \|x_i\|^2 \le b_F \|x\|^2.$$

Thus $\eta = \mathrm{sigmoid}(\alpha)/(b_F + \delta)$, with $\delta > 0$, $\alpha \in \mathbb{R}$, and $\mathrm{sigmoid}(\alpha) = 1/(1+e^{-\alpha})$, ensures the required upper bound for every input. Frobenius norms can replace block spectral norms to obtain a looser, readily computed bound. Positivity from softplus alone is insufficient. This construction is an explicit stable diffusion option; it is not a statement about the Jacobian of truncated CG.

A.4 Exact-proximal fixed points

The ADMM template explains the role of the state variables. When the local operator is an exact proximal map, it additionally permits an optimization interpretation. For closed proper convex $f = \sum_i f_i$ and exact projection P, the fixed-point conditions are

$$x^* = z^*, \quad Fx^* = 0, \quad 0 \in \partial f(x^*) + \rho u^*, \quad u^* \in \mathrm{range}(F^\top). \tag{18}$$

These are the KKT conditions for the hard-constrained program. Standard two-block ADMM convergence further requires a primal–dual saddle point and exact subproblem solves (Boyd et al., 2011). RecurProx is a learned stateful operator, so this result identifies a special case of the update organization.

Proof of (18). At a fixed point, $u^* = u^* + x^* - z^*$ implies $x^* = z^*$. Since $z^* = P(x^* + u^*)$, $x^* \in \ker F$ and $Pu^* = 0$. Therefore $u^* \in \mathrm{range}(F^\top)$. The proximal optimality condition is

$$0 \in \partial f(x^*) + \rho\big[x^* - (z^* - u^*)\big] = \partial f(x^*) + \rho u^*.$$

Writing $\rho u^* = F^\top \nu$ gives the KKT equations. Conversely, a KKT solution and $u^* = F^\top \nu/\rho$ satisfy the fixed-point equations. This is a characterization, not an existence proof. Convergence uses the separate saddle-point assumption of standard ADMM. For example, a convex objective whose effective domain misses $\ker F$ need not admit any feasible solution. □

A.5 Communication split

Proposition A.4 (Stability of the communication split)**.** *Let S be symmetric with spectrum in $[0,1]$. The map $w \mapsto (Sw, (I-S)w)$ is non-expansive in the Euclidean product norm and is an isometry when $S = P$. The map $(x,u) \mapsto (S(x+u), (I-S)(x+u))$ has norm at most $\sqrt{2}$, with equality for $S = P$ on a nonzero state space.*

The split applies to exact soft consensus and stable diffusion. The complete round also includes the learned local update, addition of proposal and dual state, and persistent private state. Finite-CG coefficients depend on the input, so its Jacobian cannot be analyzed as a fixed matrix.

Proof of Proposition A.4. Diagonalize $S = V \,\mathrm{diag}(s_j) V^\top$ with $s_j \in [0,1]$. For any w,

$$\|Sw\|^2 + \|(I-S)w\|^2 = \sum_j \big[s_j^2 + (1-s_j)^2\big](V^\top w)_j^2 \le \|w\|^2.$$

For a projector each bracket equals one. The addition map $A(x,u) = x + u$ has norm $\sqrt{2}$, so composing it with the split gives the stated upper bound. For $S = P$, the split preserves the norm of $A(x,u)$; choosing $x = u \neq 0$ attains $\sqrt{2}$. This argument concerns derivatives with respect to w, not with respect to S. □

A.6 TRAINING DERIVATIVES AND THE MISSING PATH

For the full state update $h^{k+1} = \Phi_\vartheta(h^k)$ with trainable parameters ϑ, assume $\|\partial_h \Phi\| \le L_\Phi$, $\|\partial_\vartheta \Phi\| \le C_\vartheta$ along the retained trajectory and terminal-loss derivative bounds G, H for its state and direct-parameter arguments. Let g_B be the parameter gradient through a retained window of B rounds. Then

$$\|g_B\| \le H + GC_\vartheta \sum_{j=0}^{B-1} L_\Phi^j. \tag{19}$$

Longer windows include more interactions at increased memory cost; this bound concerns the retained computation rather than an implicit equilibrium derivative. Let h^0 be detached, and $h^{j+1} = \Phi_\vartheta(h^j)$ for $j = 0, \dots, B-1$. Set $J_j = \partial_h \Phi_\vartheta(h^j)$ and $A_j = \partial_\vartheta \Phi_\vartheta(h^j)$. Direct expansion gives

$$\frac{\partial h^B}{\partial \vartheta} = \sum\nolimits_{j=0}^{B-1} J_{B-1} \cdots J_{j+1} A_j,$$

where an empty product is the identity. Applying the chain rule to the terminal loss proves (19). A regularizer must be included in the derivative bounds if it is included in the loss. The bound can depend on T, the state magnitude, the parameterization and the input, even though it has no explicit product over the discarded $K - B$ rounds.

If a parameter ω enters only $Q_{F,T}$, and the retained final round is

$$x^K = R_\phi(\bar{z} - \bar{u}, e; \bar{x}, \bar{r}), \quad z^K = Q_{F,T}(x^K + \bar{u}), \quad \mathcal{J} = \mathcal{L}_{\text{task}}(\bar{\ell}(x^K), Y),$$

then $\partial_\omega \mathcal{J} = 0$. Here $\bar{\ell}(x^K)$ includes decoding and overlap aggregation; bars on state variables denote detachment. With two retained rounds there is a structural path

$$\omega \longrightarrow z^{K-1}, u^{K-1} \longrightarrow a^{K-1} = z^{K-1} - u^{K-1} \longrightarrow x^K \longrightarrow \mathcal{J}.$$

This permits a task gradient but does not guarantee a nonzero value on every example. An alternative is decoding z^K, which changes the readout and must be evaluated separately. Neither alternative recovers all discarded long-range derivatives.

Local Jacobian bound. For fixed ρ, suppose each application of the local block is l-Lipschitz in its arguments. Holding each trajectory's a, e, x constant during its private updates, differences between two trajectories obey

$$\|\Delta r_n\| \le l^n \|\Delta r_0\| + \Big(\sum_{j=1}^{n} l^j\Big)(\|\Delta x\| + \|\Delta a\| + \|\Delta e\|).$$

The proposal obeys $\|\Delta x^+\| \le l(\|\Delta x\| + \|\Delta r_n\| + \rho\|\Delta a\|)$. Stacking these two bounds gives a product-space bound with sums and lower-order powers of l; it is not generally proportional to l^{n+1}. Parameter derivatives require separate assumptions.

Gradient connectivity and magnitude. For logits $(\theta^2, 0)$ and a target in the second class, the cross-entropy is positive at $\theta = 0$ but its parameter gradient is zero. Adding copies of this loss cannot change that fact. Conversely, a bounded state Jacobian does not bound a parameter Jacobian on an unbounded forward trajectory. These counterexamples separate gradient connectivity from a positive lower bound on gradient magnitude.

A.7 AGREEMENT AND READOUT

Suppose both endpoints of edge e decode shared-cell logits using the same L_e-Lipschitz function g_e. Then

$$\|g_e(F_{i\to e} x_i) - g_e(F_{j\to e} x_j)\|_\infty \le L_e \|F_{i\to e} x_i - F_{j\to e} x_j\|_2.$$

If one endpoint's largest logit exceeds its second largest by more than twice this bound, both endpoints have the same argmax. The general NRR decoder also uses private coordinates and local input and does not impose this factorization. Latent residuals, decoded conflicts, and exact correctness must therefore be measured separately. Without this shared factorization the implication fails. For example, take $F_{i\to e} = F_{j\to e} = [1, 0]$, $x_i = (0, 1)$ and $x_j = (0, -1)$, and decode the sign of the second coordinate. The edge residual is zero while the labels disagree. Even a common decoder and identical labels cannot certify the ground-truth answer.

A.8 PER-ROUND COMPUTATIONAL COST

Let agent i process q_i tokens of width h in a fixed-layer attention block. One network application costs $O(q_i^2 h + q_i h^2)$. If its negotiated state has width d_v and dense edge restrictions have width d_e, one round costs

$$O\left((n+1)\sum_i (q_i^2 h + q_i h^2) + T|E|d_e d_v + TNd_v\right). \tag{20}$$

The final term accounts for vector operations and global reductions in CG. For a vector MLP, the local computation term becomes $O(N(n+1)d_v^2)$.

Suppose each nonempty local view contains a bounded number of tokens, the overlap graph has bounded degree, and each of the L_{tok} input tokens belongs to at most c views. Then $N \le \sum_i q_i \le cL_{\mathrm{tok}}$, so the per-round work is $O(L_{\mathrm{tok}})$ at fixed widths and iteration counts. Total inference cost includes the executed rounds, encoding, map construction, decoding, and any additional views or trajectories. End-to-end latency is measured separately in the experiments.

B IMPLEMENTATION DETAILS

This appendix describes how NRR is implemented. We first fix notation (Section B.1) and the benchmark and controlled settings (Sections B.2 and B.3). We then describe how each task is divided into overlapping views and how views are compared (Section B.4), how the consensus step is solved (Section B.5), and how the model is trained (Section B.6). Section B.7 lists the gradient checks. Baselines, ablation controls, and evaluation procedures are described in Sections C and D.

B.1 NOTATION AND TENSOR INTERFACES

The input is D and the supervised output is Y. We distinguish output coordinates $v \in \mathcal{V}$ from the agent nodes $i \in V = \{1, \dots, N\}$ of the graph $G = (V, E)$. The view $S_i \subseteq \mathcal{V}$ is the set of output locations covered by agent i, d_i is its observed input, and the local index map π_i sends an absolute output location to a local slot. The set $\mathcal{Y}(D)$ of valid outputs, such as the Sudoku grids that preserve the given clues, describes the task; the neural readout does not enforce every discrete validity constraint by itself.

Each of $x_i, r_i, z_i, u_i, a_i, e_i$ has width d_v, so the stacked proposal has width Nd_v. For an edge $e = (i, j)$, each endpoint restriction has shape $d_e \times d_v$, and F has shape $(|E|d_e) \times (Nd_v)$. The raw base map F_e^0 is shared by all endpoints of the same edge type; the subscript e indicates this type rather than a separate parameter tensor for every edge. The factors $U_i \in \mathbb{R}^{d_e \times 4}$ and $V_i \in \mathbb{R}^{d_v \times 4}$ add an input-dependent modulation of rank at most four. Maps are built from embeddings of the observed input, never from the current proposal or the unknown output. The orientation chosen for each edge fixes the signs in F; reversing it leaves $F^\top F$ unchanged.

The weights θ, ϕ, and ψ belong to the encoder, local reasoner, and decoder. Coupling-only parameters are those used solely to build the communication operator, so they exclude encoder weights that also shape the local embeddings. Three scalars weight different terms: ρ weights the local anchor, β sets the soft-consensus strength, and λ weights the optional agreement penalty during training. The logit vector of agent i at location v is ℓ_{iv}, its average over the covering views is $\bar{\ell}_v$, and $\mathcal{L}_{\mathrm{task}}$ is the differentiable supervised loss. The discrete prediction $\widehat{Y}$ is formed only after these logits are scored.

The budgets n, T, K, M, B, R count private updates per round, solver iterations per communication step, rounds per training segment, the maximum number of supervision segments, rounds retained for gradients, and inference rounds. We write $x^{(m),K}$ for the final proposal of segment m; states carry forward between segments, but gradients do not. Inference starts from a reset state and decodes x^R with the same decoder that training applies to x^K. For ARC, the scalar V counts transformed inputs and B_{traj} counts independent trajectories per transformed input; both are separate from the number N of views, so a run with $B_{\mathrm{traj}} = 1$ still executes the predictor V times. Finally, q_i and h are the token count and token width of local attention; a pooled proposal of width d_v need not have one slot per token.

B.2 Benchmark and Controlled Settings

Table 2 maps the two comparison settings to their configurations. Baseline interfaces appear in Section C; task, selection, and stopping procedures are collected in Section D.

Table 2: **Benchmark and controlled configurations.** The two settings use separate checkpoints, training allowances, and selection procedures.

Field	Benchmark	Controlled
Sudoku/Maze NRR score	95.2 / 90.4	94.8 / 90.1
Training split	All 1000 released training instances	Grouped 900 / 100 train / validation
Seeds	42, 123, 456, 789, 1024	42, 123, 456, 789, 1024
Communication	Normalized soft CG, $\beta = 3$; task-specific T	Normalized soft CG, $\beta = 3$; $T = 5$ (MNIST: 3)
Gradient window	$B = 2$, proposal readout	$B = 2$, proposal readout
Training schedule	60k / 100k / 20k updates for Sudoku–Maze / ARC / MNIST	Checkpoint duration selected on validation data
Local warm start	10% of updates	Procedure in Section B.6
Selection	Final EMA at fixed duration	Validation selection; six LR/WD choices (Section D.3.1)
ARC model	Compact pooled support context and shape head	Demonstration Transformer and shape head

B.3 Benchmark Configuration

Table 3: **Task-specific NRR configuration.** N is the benchmark structural agent count; d_v/d_e are proposal/comparison widths, K is rounds per segment, T is solver steps, n is private updates, and M is the maximum segment count. Parameter counts refer to the benchmark core modules. The ARC value excludes context and shape heads. The explicit controlled graphs and demonstration-conditioned ARC interface are specified separately in Sections B.4 and D.2.1.

Task	**Decomposition**	N	d_v	d_e	K	T	n	M	**Core params**
Sudoku-Extreme	constraint groups	27	288	32	24	5	6	16	5.1M
Maze-Hard	8-neighbor patches	~ 100	64	16	30	5	6	16	1.3M
ARC-AGI	8-neighbor patches	100	128	32	16	5	6	16	2.6M
MNIST	3×3 patches	~ 100	32	24	10	3	4	8	0.4M

The benchmark model uses shared encoder and decoder MLPs with one hidden layer of width 512. The local block has two layers: an MLP mixer for Sudoku, local self-attention for Maze and ARC, and a vector MLP for MNIST. Restrictions combine an edge-type base with rank-four content modulation. We train with Adam-atan2 (Everett et al., 2024), learning rate 10^{-4} with a 2000-step warm-up, weight decay 1.0, batch size 768, EMA 0.999, and StableMax cross-entropy (Prieto et al., 2025). Networks run on an A100 80GB GPU in bfloat16. Table 3 lists the task-specific state widths and iteration counts, and Section D.4 describes seed aggregation.

StableMax maps a logit t to $s(t) = t + 1$ for $t \geq 0$ and $s(t) = 1/(1 - t)$ for $t < 0$, and normalizes $p_c = s(\ell_c)/\sum_j s(\ell_j)$. We differentiate the cross-entropy through this transform, so the loss in the gradient analysis is the supervised loss actually used in training, including the normalization.

The shared local blocks use this two-layer architecture. On Sudoku, the mixer lifts nine width-32 cell slots to token width 512, uses token-mixing hidden width 64 and channel-mixing hidden width 2048 in each of two blocks, and projects each slot back to width 32. Maze local attention lifts the width-64 vector to 16 width-256 tokens, applies two layers with eight heads and feed-forward width 512, and then pools and projects back to width 64. MNIST uses a vector MLP $32 \to 512 \to 512 \to 32$ with GELU between its affine maps. The benchmark ARC block is specified in Section D.2.2. Parameter totals include all these modules (Section D.5).

B.4 Views, Overlap Graphs, and Restriction Maps

Negotiation requires knowing which part of the output each local representation describes. Each decomposition therefore specifies a local index map π_i, a mask of valid locations, and, for every edge, the shared output variable it compares. The readout averages the logits of all views that cover a location:

$$\bar{\ell}_v = \frac{1}{|I_v|} \sum_{i \in I_v} \ell_{i,\pi_i(v)}, \quad I_v = \{i : v \in S_i\}, \quad \widehat{Y}_v = \arg\max_c \bar{\ell}_{v,c}.$$

Every required output is covered by at least one view, and ties are broken by a fixed class ordering. For MNIST's single label, the output set has one element, and the reducer combines the class logits of the contributing agents instead of predicting a label at every pixel. The all-view uncoupled comparator uses the same reducer, whereas the local-only baseline keeps a single view (Section C.6).

Sudoku. The controlled graph has 27 views, one for each row, column, and box. Each view holds nine 32-dimensional cell slots, so $d_v = 288$. The three views that cover a cell are connected pairwise, giving $81 \cdot 3 = 243$ edges with $d_e = 32$; a row and a box that share three cells are therefore joined by three parallel edges. Each edge stores its two local slot indices in addition to its endpoint IDs, and every node has degree 18. Fixed semantic selectors $C_{iv} \in \mathbb{R}^{32\times 288}$ compare $C_{iv}x_i - C_{jv}x_j$ at the same absolute cell, which aligns the two coordinate systems even when the cell occupies different local positions. Grouping several shared cells into one edge would require a wider comparison space, such as width 96 for three cells, and would change the operation count, so the controlled comparisons use one edge per cell. Table 3 reports the benchmark agent count separately.

Maze and ARC. The controlled graph uses 4×4 patches with stride 3. Patch origins lie on the stride grid, and masked padding on the right and bottom completes the last patch. Widths 30, 45, 60, and 90 give 100, 225, 400, and 900 agents. Immediate eight-neighbors are connected when their valid cells overlap, and each edge stores the shared absolute coordinates. Padding creates no output variables or constraints. For an $m \times m$ patch grid, the number of undirected edges is $2m(m-1) + 2(m-1)^2$, which gives 342, 1482, and 3422 edges at widths 30, 60, and 90 before any shuffling or deletion. Patch tokens can be pooled into the negotiated state, so the token width and d_v are different quantities.

MNIST. The benchmark graph uses nonoverlapping 3×3 patches at stride 3, with origins $0, 3, \ldots, 27$ on each axis and masked bottom and right padding to width 30, giving exactly 100 agents connected by the spatial eight-neighbor graph. The controlled graph uses 3×3 patches at stride 2, with origins 0 through 26 and one masked bottom and right pixel, giving 196 agents. Because the task has a single label, edges exchange class evidence rather than values at shared pixels. Padding and corruption are applied before the graph is built.

Restriction maps. On Sudoku, we learn nine base restriction matrices, one for each local cell slot, shared by row, column, and box agents; the endpoint type is the local slot. Maze and ARC use eight endpoint types given by the signed neighbor displacement, with the reverse displacement at the other endpoint. MNIST uses one shared base map. The rank-four modulation is computed from the endpoint node's observed embedding. Replacement maps used in the ablations are defined in Section C.6.

B.4.1 Restriction Alignment and Rank

The restriction operator determines what consensus enforces: its kernel contains the states on which all views agree, and hard projection keeps only that part. Global normalization controls the scale of the maps, but it does not make them compare the right cells. Raw map norms, singular values, normalized disagreement, and decoded overlap conflict therefore describe different properties. Orthonormal initialization also does not guarantee full rank once content modulation is added. We now compute the kernel for aligned selectors and unaligned identity maps, and then explain why pooled patch states need a different treatment.

Aligned selectors. On Sudoku, rows are ordered left to right, columns top to bottom, and boxes in row-major order. The selector onto local slot p is

$$C_p = \delta_p^\top \otimes I_{32} \in \mathbb{R}^{32\times 288},$$

where $\delta_p \in \mathbb{R}^9$ is the pth standard basis vector, so that $C_{iv} = C_{\pi_i(v)}$. Any slot permutation is equivalent if the encoder, decoder, and selectors are permuted together. The three copies of each absolute cell form a triangle whose incidence matrix has rank two. The 243-edge selector operator, a square 7776×7776 matrix, therefore has

$$\operatorname{rank}(F) = 81 \cdot 2 \cdot 32 = 5184, \qquad \dim\ker F = 81 \cdot 32 = 2592.$$

Nonzero global normalization leaves these dimensions unchanged. Hard projection onto this kernel averages the three latent copies of each cell, preserving one 32-dimensional representation per absolute cell.

Unaligned identity. The unaligned control uses the rectangular identity $J = [I_{d_e}\ 0]$ at every endpoint, with $d_e \le d_v$. On a connected graph, it has rank $(N-1)d_e$ and nullity $N(d_v - d_e) + d_e$, which are 832 and 6944 for Sudoku. It compares the first d_e local coordinates regardless of which absolute cells they represent. A rectangular shape, or full row rank on each edge, therefore does not imply full column rank for the whole operator; Section C.6 constructs a control whose full rank is guaranteed.

Pooled patches. For pooled patch states, a shared absolute cell does not automatically define a linear selector on $x_i \in \mathbb{R}^{d_v}$. Suppose $x_i = \Gamma_i t_i$ pools the token states t_i, and a token-level selector C_{iv} extracts the desired cell feature $C_{iv}t_i$. A linear restriction F_{iv} with $F_{iv}\Gamma_i = C_{iv}$ exists only if $\ker\Gamma_i \subseteq \ker C_{iv}$, and mean pooling over unconstrained token states generally violates this condition. The Sudoku selector formula therefore cannot be transferred directly to pooled Maze or ARC states. An aligned control for Maze needs a compatible representation or a different comparison space, which changes the interface as well as the maps (Section C.6).

B.5 Consensus Solver

Both settings use proposal readout, $B = 2$, $\rho = 1$, $\lambda = 0$, normalized restrictions from (10), and soft-consensus CG with $\beta = 3$ (Table 2). The graph and maps depend on observed inputs only. Restriction tensors are recomputed after each parameter update, and differentiation includes the active branches of the maximum and absolute-value operations in the normalization.

Soft consensus. For $w = x + u$, we solve $Az = w$ with $A = I + 3F^\top F$, starting from zero, with no preconditioner, diagonal modification, or solver warm start. The budget is five CG steps for Sudoku, Maze, and ARC and three for MNIST. Map construction, normalization, sparse matrix–vector products, and scalar reductions run in float32 with mixed-precision autocasting disabled, and network states are cast at the solver boundary. The sparse operator and its transpose are exact algebraic adjoints. On small graphs, a float64 Cholesky solve provides the numerical reference. The strength ablation retrains separate models with $\beta \in \{1, 3, 9\}$; at $\beta = 9$, the bound on the condition number rises from four to ten.

Finite CG and backpropagation. Starting from $z_0 = 0$, we set $b = w$, $\varrho_0 = b - Az_0$, and $p_0 = \varrho_0$. Each step computes

$$\alpha_t = \frac{\varrho_t^\top \varrho_t}{p_t^\top A p_t}, \qquad z_{t+1} = z_t + \alpha_t p_t,$$

$$\varrho_{t+1} = \varrho_t - \alpha_t A p_t, \qquad p_{t+1} = \varrho_{t+1} + \frac{\varrho_{t+1}^\top \varrho_{t+1}}{\varrho_t^\top \varrho_t} p_t.$$

Each sample has its own inner products and convergence mask. Before the first division and before each direction update, we check whether $\|\varrho_t\| \le 10^{-6}\max(1, \|b\|)$. A converged sample keeps its iterate fixed, and the remaining steps act as identities; this also handles a zero right-hand side without division by zero. We differentiate through the executed iterations, their coefficients, and

both occurrences of F. Because the termination masks are discrete, numerical derivative checks avoid mask boundaries.

A nonfinite value or a nonpositive denominator before convergence counts as a solver failure. In training, the affected update is skipped; in evaluation, the instance is scored as incorrect and flagged. Numerical checks recompute the true residual $b - Az_T$, which can differ from the residual updated inside the recursion, and charge the extra matrix–vector product. CG reductions are also included in work and synchronization accounting.

Hard projection and diffusion variants. Hard projection uses $P = I - F^{\dagger}F$, computed by a float64 SVD that discards singular values below $10^{-10}\max(1, \sigma_{\max})$. Its rank and nullity determine which representations survive (Section B.4.1). A finite alternative for the hard system solves $FF^{\top}\nu = Fw$ from zero and returns $z_T = w - F^{\top}\nu_T$; its nonzero spectrum need not satisfy the soft system's condition-number bound.

The diffusion variant starts from $z_0 = w$ and repeats $z_{t+1} = z_t - \eta F^{\top}Fz_t$, using the input-dependent step size of Section A.3, which guarantees $0 < \eta \leq 1/\lambda_{\max}$. This iteration approaches hard agreement rather than the soft target; gradient descent on the soft objective would require the additional term $z_t - w$. Comparing diffusion with CG therefore changes the target as well as the numerical method. Sweeps on frozen checkpoints measure numerical sensitivity, whereas separately retrained variants measure how well the model adapts to each solver. The residual and accuracy curves in Figure 10 come from the frozen and retrained comparisons, respectively.

B.6 Coupled Training and Local Warm Start

Training carries the forward reasoning state across supervision segments but keeps only the last $B = 2$ rounds for gradients. This preserves a path from negotiation to the next supervised proposal without storing the complete trajectory.

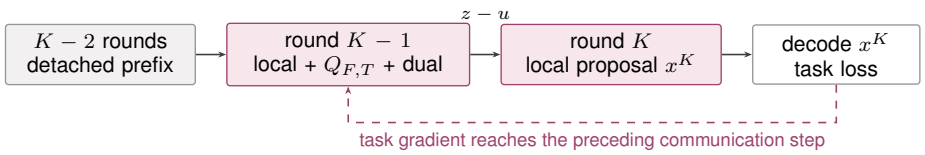


Figure 4: **Supervision reaches the preceding negotiation step.** The final proposal x^K depends on the anchor produced in round $K-1$. Retaining both rounds connects task loss to coupling parameters. The final round's communication updates carried state but does not affect its own proposal readout.

Initialize $h = (x, r, z, u) = 0$; set $B = 2$, $\rho = 1$, $\lambda = 0$.
for segment $m = 1, \ldots, M$:
  Detach h; recompute embeddings and restriction maps from the input.
  Run $K - B$ complete rounds without retaining backward graphs.
  Detach h at the retained-window boundary.
  for retained round $j = 1, \ldots, B$:
    $a = z - u$; update all (x_i, r_i) by (6)–(7).
    $w = x + u$; $z = Q_{F,T}(w)$; $u = u + x - z$.
  Decode x and aggregate all covered output coordinates.
  Backpropagate the StableMax task loss divided by M.
  Clip total gradient norm at 1; update parameters; detach h.
At inference, reset h per transformed input and execute R rounds.

Parameters stay fixed within a segment. Embeddings and restrictions are recomputed after each optimizer update, and their derivatives are retained in the final two rounds. The $1/M$ loss factor follows (12), and the stated learning rate assumes this scaling. Each retained round includes all $n + 1$ local network applications and the communication operations.

Training duration and carried state. Each executed segment performs one optimizer update, so training length is counted in segments rather than recursive rounds. If training visits batches

$b = 1, \ldots, J$, each with M_b executed segments, and warm start uses U_{warm} updates, the total number of updates is

$$U_{\text{total}} = U_{\text{warm}} + \sum_{b=1}^{J} M_b.$$

Under the fixed-M schedule of Section D.3.1, $M_b = M$ except when the update budget ends in the middle of a batch; that subsection also gives the resulting benchmark totals. The 2000-step learning-rate warm-up and the local warm start below are separate stages. With fixed M, the oldest carried state reaches KM rounds: 384, 480, 256, and 80 for Sudoku, Maze, ARC, and MNIST. Detachment stops derivatives at segment boundaries, while the numerical state carries over.

Local warm start. Before coupled training, local warm start teaches the local solver to refine an anchor that is close to the answer. Each task has a fixed, nontrainable linear target embedding E_y, generated once with seed 20260920. Sudoku maps a nine-class one-hot vector into each 32-dimensional cell slot using orthonormal columns from a Gaussian QR factorization, with a deterministic positive-diagonal convention. Maze embeds its 16 two-class cell labels into width 64 in the same way. ARC maps 160 flattened one-hot color entries into width 128 with independent Gaussian entries of variance $1/160$, and MNIST maps its ten-class label into width 32 with orthonormal columns. Each nonzero embedded target is normalized to root-mean-square one, which fixes the scale of the noise comparison below. Padding contributes zeros and is excluded from the supervised loss.

The training anchor is $a = E_y(y_i) + \sigma\xi$, with $\xi \sim \mathcal{N}(0, I)$ and default $\sigma = 0.3$. For each warm-start example, we reset (x, r) to zero, run one local round with this anchor, and train the encoder, local solver, and decoder on the same local targets and overlap reducer. Restriction-only parameters stay frozen, and this stage has no communication or dual state. All warm-start computation counts against the training budget in Section D.3.1. Validation and inference never use the target: they start from the zero state, and their anchors come from model states.

The warm-start variants in Figure 8 change only the anchor or the budget. The noise sweep uses $\sigma \in \{0.1, 0.3, 1.0\}$, and the zero-anchor variant sets $a = 0$. For detached-rollout anchors, the first 10% of the warm-start allowance trains a zero-initialized coupled pilot model. We freeze this pilot, sample a rollout length L uniformly from $1, \ldots, K$, and use its anchor $a^L = z^L - u^L$ computed from observed inputs; pilot training and anchor generation are charged to the warm-start budget. The student is then reset to the common initial weights before local pretraining, so the pilot serves only to generate anchors. The no-warm-start variant spends the entire allowance on coupled training.

B.7 Implementation and Gradient Checks

These checks test whether task gradients follow the paths analyzed in Section A.6. They treat the encoder, local reasoner, base restrictions, modulation head, and decoder separately. On small graphs, we compare directional derivatives with centered finite differences at steps 10^{-4}, 10^{-5}, and 10^{-6}. The detached prefix is held at its unperturbed value, so both calculations evaluate the same truncated derivative. Proposal readout with $B = 1$ and explicit map detachment serve as negative controls, since both remove the path from task loss to the coupling parameters, and full $K = B = 4$ unrolling serves as a reference for a short finite computation. For ARC, we also check that changing the query target leaves every inference input unchanged, so the target cannot leak into prediction.

C Baseline and Ablation Interfaces

The comparisons comprise alternative reasoning architectures, replacements of an NRR component, and interventions on a trained model or its inference budget. Source formulations and within-study implementations are described together below. Benchmark and controlled rows retain the separate configurations and reference scores in Section B.2. Resource accounting is specified in Section D.5.

C.1 Native and Global Recursive Baselines

Native TRM and EqR. TRM and EqR process the whole problem rather than assigning a recursive solver to each overlapping view. Their controlled rows retain task-appropriate native architec-

tures and use the data splits and validation-based selection in Sections D.3 and D.3.1; their state tensors and local blocks need not match NRR.

To avoid conflating source notation with NRR states, write a native TRM state as (h_y, h_r), with observed-input embedding e. One source latent cycle is

$$h_r \leftarrow f(e, h_y, h_r) \quad (n_{\text{src}} \text{ times}), \qquad h_y \leftarrow f(h_y, h_r).$$

The source uses $n_{\text{src}} = 6$, three cycles per supervision segment, a two-layer width-512 network, and at most 16 supervision segments; it retains gradients through the final cycle (Jolicoeur-Martineau, 2025). One such cycle is not one network evaluation. Thus its complete 16-segment forward budget contains $16 \cdot 3 \cdot 7 = 336$ shared-network applications, or 672 layer evaluations.

EqR modifies iterative updates through randomized initialization and damped noisy transitions. Using δ for damping and σ_{path} for path noise, its transition is

$$h^+ = h + (1 - \delta)\big(f(h, \text{cond}) - h\big) + \sigma_{\text{path}}\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I).$$

This notation separates source noise strength from NRR's consensus strength β. The source's reduced Maze architecture uses width 128 and one layer on Maze-Unique; it is not a specification of our Maze-Hard rerun (Huang et al., 2026). A single trajectory removes restart aggregation but need not remove randomized initialization or path noise. The within-study settings below specify these fields separately from the source configurations.

Native-model reruns. For TRM, we use the source two-layer width-512 model, the MLP mixer on Sudoku and eight-head attention on Maze, $n_{\text{src}} = 6$, three latent cycles per segment, and 16 inference segments, which matches the source maximum of 336 network calls (672 layer calls). For EqR, we use the same task-appropriate backbone, Gaussian initial states with standard deviation one, damping 0.05, and per-update Gaussian noise 0.01, with 64 inference segments and one trajectory. Both use full-horizon inference; the source halting head may allocate training segments but does not terminate these evaluations. Source gradient rules are retained rather than replaced by NRR's $B = 2$. Dataset splits and model selection follow Section D.3.1. Native models are not required to match NRR's parameter count or state dimension.

Global-RecurProx. Global-RecurProx is our within-study global-attention comparator. It applies the RecurProx update family to a globally coupled representation of the grid. Attention spans the whole grid rather than operating inside separate overlapping views, so dependencies are handled within the global recurrence. It tests the organization and cost of recursive computation. "Global" in the quality–latency panel and "global attention recurrence" in the efficiency discussion refer to this comparator. Benchmark timing and controlled accuracy–latency pairs are separate evaluations.

For Global-RecurProx, removing the graph does not uniquely determine the anchor. Two possible recurrences, $R_\phi(0, e; x, r)$ and $R_\phi(x, e; x, r)$, differ already on the next round. In particular, taking an edgeless NRR with exact soft communication gives

$$F = 0 \implies z^{k+1} = x^{k+1} + u^k, \quad u^{k+1} = 0,$$

and from a zero dual state gives self-feedback $a^{k+1} = x^{k+1}$, rather than a zero anchor. With the stated zero-start CG, the same result holds after one nonzero-right-hand-side iteration since $A = I$. Our global comparator uses the zero-anchor recurrence below, with size-independent positional encoding.

Global recurrence. The comparator uses one token per grid cell, two pre-norm Transformer layers, eight heads, GELU feed-forward width $2h$, zero dropout, and deterministic two-dimensional sinusoidal positions. The token width is $h = 512$ on Sudoku and 256 on Maze. Each axis uses $h/4$ sine/cosine frequency pairs, giving h channels across the two axes, with base 10000, so positions extend to larger mazes without interpolating a learned position table. We set $(x, r) = 0$ initially and use $a = 0$ in both RecurProx updates, with $n = 6$. Only (x, r) is maintained across rounds; there is no z, u state or sheaf operator. The proposal is decoded once per output cell. Training and inference use $K = R = 24$ for Sudoku and $K = R = 30$ at every maze width, with $B = 2$ and $M = 16$ in training and no inference early stopping. The encoder and readout have the same task inputs, targets, and one-hidden-layer width 512 as the local comparison, with shape adaptations for whole-grid tokens. Weights are trained separately.

The larger-maze datasets and checkpoint reuse are specified in Section D.7.

C.2 Proximal Local Solving

Sheaf-ADMM and the proximal local-solver comparator. Sheaf-ADMM (Seely et al., 2026) encodes local observations into convex local objectives and alternates a proximal proposal update, sheaf-based communication, and dual accumulation. At the local-update interface it uses

$$x_i^{k+1} = \arg\min_v \left[f_i(v; d_i) + \frac{\rho}{2}\|v - a_i^k\|^2 \right], \qquad a_i^k = z_i^k - u_i^k,$$

instead of RecurProx's repeated private-state updates. "Proximal solver" in the component figure identifies this optimization-based local-solver comparison. Its local solution is computed by the task-specific procedure below; it is separate from hard projection in communication. Its Sudoku/Maze scores reuse the Sheaf-ADMM rerun in Table 1. This comparator has a smaller parameter budget. The factorial experiment below provides a separate comparison of recursive and untied factorized local solvers.

The local objective family in Sheaf-ADMM can be written

$$f_i(v; d_i) = \tfrac{1}{2} v^\top Q_i v + q_i^\top v + \alpha_i \|v\|_1 + \chi_{\mathcal{B}_i}(v), \qquad Q_i \succeq 0, \quad \alpha_i \geq 0,$$

where the encoder produces objective coefficients and $\mathcal{B}_i$ can impose coordinate bounds. For diagonal $Q_i = \mathrm{diag}(q_{i,j}^{(2)})$ and a box $[l_i, h_i]$, define

$$d_{i,j} = q_{i,j}^{(2)} + \rho, \qquad t_{i,j} = \frac{\rho a_{i,j} - q_{i,j}}{d_{i,j}}.$$

The coordinatewise exact minimizer is

$$v_{i,j}^* = \mathrm{clip}_{[l_{i,j}, h_{i,j}]} \left[\mathrm{sign}(t_{i,j}) \max\left(|t_{i,j}| - \frac{\alpha_i}{d_{i,j}}, 0 \right) \right].$$

Setting $\alpha_i = 0$ removes thresholding; infinite bounds remove the box. For a general quadratic, a linear solve is sufficient only in the unconstrained, unregularized case. Convexity alone does not make a finite iterative solve exact.

The source hyperparameter appendix reports a closed-form local step on Maze, but 50 accelerated proximal-gradient steps on MNIST and Sudoku. It also specifies hard communication on MNIST and soft communication with $\gamma = 5$ on Maze and $\gamma = 2$ on Sudoku, all with five CG steps; its initial ρ values are $0.12/0.25/0.25$ for MNIST/Maze/Sudoku (Seely et al., 2026). These describe the source experiments, whose maze data and horizons differ from ours. The source does not evaluate ARC; our ARC objective is consequently an extension.

For source soft communication,

$$z_* = (I + (\gamma/\rho)\widetilde{F}^\top \widetilde{F})^{-1} w.$$

If the same raw maps are normalized as in (10), preserving this operator requires $\beta = (\gamma/\rho)(b + 10^{-6})$. Simply copying γ as $\beta = 3$ changes the operator. The rerun below explicitly chooses its communication operator and local-solver accuracy checks.

Proximal rerun. The rerun uses the same task views and readout as the corresponding benchmark NRR, with no private r. A one-hidden-layer width-512 encoder produces diagonal quadratic coefficients $Q_{jj} = \mathrm{softplus}(q_j^{\mathrm{raw}}) + 10^{-4}$, free linear coefficients q_j, and one positive ℓ_1 weight $\alpha_i = \mathrm{softplus}(\alpha_i^{\mathrm{raw}})$ per view, with $\rho = 1$. Proposals are nonnegative on Sudoku, lie in the box $[0, 1]^{d_v}$ on Maze, and are unconstrained on ARC and MNIST. These are latent constraints, not Sudoku validity constraints. All four tasks include the ℓ_1 term.

Following the source's per-task procedure, and extending it to ARC, Maze uses the closed-form diagonal update in this subsection, whereas Sudoku, ARC, and MNIST use 50 accelerated proximal-gradient steps from the anchor clipped to the feasible box, even though their diagonal quadratics would also admit the closed form. The smooth gradient is $Qv + q + \rho(v - a)$ with step size $1/\Lambda$, $\Lambda = \max_j(Q_{jj} + \rho)$. Each step applies soft thresholding by α_i/Λ and then box clipping, with standard FISTA momentum (Beck & Teboulle, 2009), $t_0 = 1$, $t_{s+1} = (1 + \sqrt{1 + 4t_s^2})/2$. The solve uses float32, a fixed 50-step budget, and no tolerance exit. The analytic diagonal solution provides

a reference for the relative solution error and the proximal-gradient mapping norm, so the error of the finite local solve is measured independently of task accuracy. Finite inner solves motivate the neutral "proximal solver" label. All reruns use NRR's normalized soft communication with $\beta = 3$ and the task-specific T, which isolates the local-update family more closely, although the parameter count remains smaller.

C.3 Recurrent Graph Baselines

Recurrent MPNN and recurrent relational network. A recurrent message-passing neural network (MPNN; Gilmer et al., 2017; Li et al., 2016) uses learned edge messages and a recurrent node update. At the model-family level, for node state s_i^k and neighbors $\mathcal{N}(i)$,

$$m_{ji}^k = \mathsf{Msg}_\chi(s_j^k, s_i^k), \qquad s_i^{k+1} = \mathsf{Upd}_\chi\left(s_i^k, \mathrm{Agg}_{j\in\mathcal{N}(i)}\, m_{ji}^k\right),$$

where Agg reduces neighboring messages and the learned functions share parameters across iterations. Its state evolves through learned messages without prescribed consensus targets or accumulated proposal–target discrepancies. The recurrent MPNN rerun in Table 1 is a benchmark architecture comparison. The recurrent relational network (RRN; Palm et al., 2018) in Figure 7a is a separate recurrent-graph baseline in the controlled setting. The 42.5% and 93.4% Sudoku entries belong to these distinct configurations.

Benchmark MPNN. The benchmark MPNN uses the NRR view graph and proposal width, with $s_i^0 = e_i$. Messages are $\mathsf{MLP}_{2d_v+16\to128\to d_e}[s_i, s_j, t_{ji}]$ with GELU, and incoming messages are averaged. One GRU of hidden width d_v takes $[e_i, \bar{m}_i]$ per round, and the shared NRR decoder reads s_i. It executes $K = 24/30/16/10$ steps for Sudoku/Maze/ARC/MNIST, with the same segment counts and final-two-step supervision window. This architecture comparison pairs one recurrent node update per graph round with NRR's seven local applications plus communication; its lower Sudoku score reflects the combined interface difference.

Controlled RRN. The controlled RRN uses the same 27-node Sudoku and patch Maze graphs, with hidden state $s_i \in \mathbb{R}^{4d_v}$ and a learned projection to the proposal width for readout. Its state starts at $s_i = [e_i, 0, 0, 0]$. Within each round, it performs six node-only GRU updates using $[e_i, 0]$, followed by five synchronous message/GRU updates using $[e_i, \bar{m}_i]$, then one node-only update before readout. Messages have shape $8d_v + 16 \to h_m \to d_e$, use GELU and mean aggregation, and the GRU has hidden width $4d_v$. Its updates and messages are shared across steps. The GRU matrices use low-rank factors, and h_m denotes the message hidden width. This is a stronger recurrent graph architecture with the same carried-state size; it does not contain prescribed proposal/target/dual channels. Table 4 summarizes the two graph baselines.

Table 4: **Graph-baseline interfaces.** The two baselines have different state sizes and update schedules.

Setting	State/view	Sudoku/Maze rounds	Updates per round
Benchmark MPNN	d_v	24 / 30	One message and node update
Controlled RRN	$4d_v$	24 / 30	Six node-only, five message/GRU, one node-only

C.4 Learned Communication Replacements

The benchmark learned residual link replaces NRR's consensus and dual updates with one learned communication update per round. The controlled persistent link is separately trained, carries communication memory across rounds, and uses five substeps. Both retain RecurProx and proposal readout; a residual skip connection is distinct from the linear-system residuals and disagreement measures defined in Sections B.5 and D.8.

The minimal interface distinguishing the link variants is

$$(x^{k+1}, r^{k+1}) = R_\phi(a^k, e; x^k, r^k), \qquad (a^{k+1}, c^{k+1}) = \mathcal{H}_\chi(x^{k+1}, e, c^k; G),$$

where c denotes communication memory when present. Unlike NRR, this interface does not prescribe $c = u$ or $a = z - u$. A residual skip connection alone does not specify whether it updates proposals, anchors, or c. The interfaces below make these choices explicit.

One-step benchmark link. The link starts from $a^0 = 0$, retains (x, r), and runs local refinement before communication. For each directed edge $j \to i$, it concatenates layer-normalized proposals and embeddings with a learned 16-dimensional endpoint-type embedding:

$$m_{ji} = \mathsf{MLP}_{4d_v+16\to d_v\to d_e}[\mathrm{LN}(x_i^+), \mathrm{LN}(x_j^+), \mathrm{LN}(e_i), \mathrm{LN}(e_j), t_{ji}], \qquad \bar{m}_i = \frac{1}{\deg(i)} \sum_{j\to i} m_{ji}.$$

Layer normalization has no affine parameters; MLP hidden activations are GELU, with biases and zero dropout. An isolated node uses $\bar{m}_i = 0$. Parallel edges contribute separately. A residual anchor update is

$$a_i^{k+1} = x_i^{k+1} + \mathsf{MLP}_{d_v+d_e\to d_v\to d_v}[x_i^{k+1}, \bar{m}_i].$$

It does not overwrite the supervised proposal. No independent communication memory or dual is retained. One message network is shared across all edges and rounds; type embeddings supply direction/slot information. The final anchor is carried to the next segment, with the same detachment boundary as the other states.

Persistent five-step link. Each agent carries $c_i \in \mathbb{R}^{2d_v}$, initialized to zero, giving $2d_v$ communication-state scalars per agent, the same number as the combined z, u states. Each round starts from the previous c_i and runs five synchronous substeps. At substep t,

$$m_{ji,t} = \mathsf{MLP}_{6d_v+16\to d_v\to d_e}[x_i^+, x_j^+, c_{i,t}, c_{j,t}, t_{ji}], \qquad c_{i,t+1} = \mathsf{GRU}_{2d_v}([x_i^+, e_i, \bar{m}_{i,t}], c_{i,t}).$$

Each vector input is normalized as above, incoming messages are averaged, and the message and GRU weights are shared across all five substeps and rounds. The anchor is $a_i^{k+1} = x_i^+ + W_a c_{i,5} + b_a$, and $c_{i,5}$ is carried across rounds and supervision segments, detached wherever NRR detaches z, u. The output remains the proposal, so two retained rounds also connect this learned link to task loss. These five learned updates have their own operation costs; Section D.5 distinguishes update counts from FLOPs.

C.5 Local Recursion and Disagreement-Memory Factorial

No disagreement memory. "Without dual" and "no dual" remove disagreement memory: they retain local recursion and the consensus operator but set $u^k = 0$ throughout. Hence $a^k = z^k$, $w^{k+1} = x^{k+1}$, and $z^{k+1} = Q_{F,T}(x^{k+1})$; no accumulated discrepancy enters the next anchor. This variant has its own no-dual reference, distinct from the benchmark row.

Feed-forward local solver. The feed-forward solver unrolls one round into $n+1$ distinct copies of the local block: the first n update r and the last updates x, with the same additions and side inputs as RecurProx. There is no within-round weight tying, but the same feed-forward stack is reused across rounds, and (x, r) persist. Every dense matrix in the untied copies is parameterized as a product of two low-rank factors, which is materialized before use. A factored matrix contributes its rank times the sum of its row and column dimensions to the parameter count, before biases, embeddings, and normalizations. At inference, the materialized matrices are cached; during training, materialization and its backward pass are included in the cost. This produces distinct blocks at the same active dense forward dimensions. Factorization also changes the weight class, so this control compares tied full-rank recursion with untied factorized computation; it cannot support a causal attribution to weight tying alone.

C.6 Map, Topology, and Training Interventions

Restrictions and decomposition. "Shared maps only" removes input-dependent modulation while keeping the trainable edge-type bases. The unaligned identity control uses $J = [I_{d_e}\ 0]$ at both endpoints (Section B.4.1). Fixed random maps draw independent $\mathcal{N}(0, 1/d_v)$ entries for each endpoint, seeded by the run seed, with no type sharing or content modulation; they are frozen

and receive the same global normalization. On Sudoku, the aligned maps are the semantic selectors of Section B.4.1. The Maze aligned control replaces the pooled width-64 proposal with 16 four-dimensional cell slots, and the endpoint restrictions extract the shared border cells, in absolute row-major order, into width 16, padding unused comparison coordinates with zeros. Horizontal and vertical overlaps share four cells, and diagonal overlaps share one. The reasoner lifts each cell slot to its local token and projects it back per slot, and each slot is decoded at its own cell. The total state width is unchanged, but this control changes the representation as well as the maps, so it is not a pure restriction replacement.

The generic full-rank hard control on Sudoku keeps the semantic slot support but premultiplies each endpoint selector by an independent Gaussian 32×32 matrix. For each cell triangle, the resulting 96×96 block is accepted only if $\sigma_{\min}/\sigma_{\max} > 10^{-4}$ and is otherwise resampled with the run seed. The restrictions are then frozen. Each accepted block has rank 96, so the direct sum has rank 7776 at every training step, and normalization cannot change it. Hard projection is therefore zero, whereas the aligned-selector projection preserves 2592 coordinates. Because the maps are frozen, this control tests the rank-based explanation without assuming that training preserves the rank of a random initialization.

Sudoku's constraint-group and patch decompositions change the view sets. The patch decomposition uses 3×3 patches at stride 2, with origins $0, 2, 4, 6, 8$ on each axis and masked bottom and right padding, giving 25 agents. Neighboring patches share one edge per shared valid cell, and every required cell remains covered. Maze and MNIST already use patches, so their repeated full-model entries are references rather than new variants.

Topology changes and communication removal. Shuffling applies double-edge swaps on the undirected per-cell multigraph, replacing $(a, b), (c, d)$ by $(a, d), (c, b)$ and rejecting self-loops and unchanged swaps while allowing parallel edges. Each node's endpoint slot and type move with its stub, so its degree and type counts are preserved. We accept $20|E|$ swaps using PCG64 seed $20260920 + s$, where s is the run seed, and save the resulting edge list. This holds the node-local maps fixed while destroying the original overlap pairings. The normalization is recomputed on the altered operator. The half-edge control samples 122 of 243 edge IDs without replacement with the same seed convention; removed edges do not enter the active operator or its normalization. Each run uses one graph for both retraining and its corresponding inference-only intervention.

Inference-only interventions change the graph before the first round of a reset trajectory, without transferring latent states from the original graph. The no-message control sets $a = z = u = 0$ throughout and retains local (x, r) recursion and all-view mean-logit readout; the benchmark uncoupled vote uses the same zero-anchor convention. These no-message and uncoupled-vote controls differ from the edgeless self-feedback recurrence in Section C.1 and from the no-dual model. Local-only prediction keeps only the first row-major view (the first row group on Sudoku), emits an explicit missing symbol at uncovered output locations, and scores any missing required coordinate as incorrect. MNIST has one globally shared label and uses that view's class logits.

Retraining allows adaptation to the changed graph, while an inference-only intervention uses the full checkpoint. Fixed-selector and random-map replacements appear only in the retrained comparison. Local-only zero grid scores follow from missing-coordinate scoring, whereas the all-view no-communication test covers the complete output.

Communication and training interventions. Changing β changes the soft target, whereas changing T changes its numerical approximation. Hard projection and diffusion use the operators in Section B.5; frozen-checkpoint and retrained solver-budget sweeps follow Section D.4. Warm-start variants use Section B.6. Gradient-window controls either vary $B \in \{1, 2, 4\}$ with proposal readout or decode the consensus target at $B = 1$; each is trained separately at the same 24-round Sudoku horizon. Full $K = B = 4$ unrolling and explicit map detachment are derivative checks, not additional accuracy rows.

Task-specific CNN and ARC controls are defined with their inputs in Section D.2; depth/size and stopping protocols appear in Sections D.7 and D.9.

D Evaluation and Reporting Protocol

This appendix specifies how results are produced and reported. We first describe the tasks and their interfaces (Sections D.1 and D.2), data separation and model selection (Section D.3), seed aggregation (Section D.4), and resource and timing measurement (Section D.5). We then give the protocols for individual experiments in the order in which Section E reports them: difficulty stratification, depth and size transfer, residual diagnostics, and stopping (Sections D.6 to D.9). Section D.10 lists records not included in this version. Implementation and baseline definitions are in Sections B and C; the benchmark and controlled settings are summarized in Section B.2.

D.1 Tasks and Coordination Requirements

The datasets probe different relationships between local evidence and the required answer. Section 5.1 gives split sizes and primary scoring rules; the descriptions below explain each task's role in the evaluation.

Sudoku-Extreme: simultaneous constraint satisfaction. The grid-completion task instantiates the clue-preserving row, column, and box constraints in Section 2. Its coordination challenge is to satisfy these constraints simultaneously: a locally admissible choice can still conflict with another group's completion. The benchmark tests whether reconciling such partial solutions improves the entire answer.

Maze-Hard: spatial connectivity. The model must identify a shortest route from the start to the goal. Local path segments must join into one route through the maze, so compatibility must support spatial connectivity rather than just simultaneous assignment constraints. This complements Sudoku's constraint-group structure. Section D.2 distinguishes the released reference-path task from alternative maze evaluation protocols.

ARC-AGI-1/2: inferring the task from demonstrations. The few demonstration input–output pairs specify an unknown transformation to apply to a new query. Unlike a fixed puzzle rule, the relevant relation must first be inferred from those examples; coordination then concerns consistent application of that inferred rule across the output. Section D.2.1 specifies how demonstrations condition the solver and how transformed predictions are aggregated into the allowed candidates.

MNIST: spatial evidence for one label. MNIST (LeCun et al., n.d.) classifies a handwritten digit with a single class label. Padding, pixel dropout, and additive noise test the robustness of spatially distributed evidence. It contrasts with tasks requiring a globally consistent set of output variables: combining redundant evidence can suffice even when independent local decisions would fail to solve a structured grid. The robustness experiments and stronger convolutional controls appear in Section E.9.

D.2 Task Interfaces

Sudoku and Maze use the released HRM splits. Clue preservation and constraint validity are complementary checks for Sudoku. Results on Maze-Unique are not substituted for the Maze-Hard reference-mask metric.

D.2.1 Demonstration-Conditioned ARC

ARC requires inferring the rule before coordinating its application. The controlled interface therefore receives an episode $(\mathcal{C}, g)$ with demonstration pairs $\mathcal{C} = \{(g_j^{\text{in}}, g_j^{\text{out}})\}_j$ and query input g. The query output y supplies training supervision only. There is no task-ID embedding or task-specific optimization.

Encoding demonstration context. Each observed grid occupies a 30×30 canvas with recorded height, width, origin, and a valid-input mask. PAD is distinct from the ten colors. A shared two-layer, four-head pre-norm Transformer uses width 128, feed-forward width 256, GELU, zero

dropout, and learned row/column and role embeddings. PAD keys are masked. Valid tokens are mean-pooled into 100 patches; empty patches use a learned role-specific vector.

Corresponding input/output patch summaries are combined by a shared $256 \to 128 \to 128$ MLP, with a learned $4 \to 128$ projection of $(h^{\text{in}}, w^{\text{in}}, h^{\text{out}}, w^{\text{out}})/30$. An entry is masked only if both patches are empty. A pair-summary entry, formed from the mean of valid paired entries and a summary-role vector, gives 101 context slots per demonstration. Concatenation over demonstration pairs produces the memory. Pair order has no numeric embedding, so permuting pairs permutes only the memory keys.

Query-to-output computation. The shared grid encoder also processes the query. Each of the 100 output-canvas agents combines its patch coordinates, local query summary, and four-head cross-attention to demonstration memory to form $e_i \in \mathbb{R}^{128}$. An absent-query vector represents positions beyond the observed query. All output slots exist independently of the unknown output size, allowing outputs larger than the input. The negotiation graph covers the full output canvas with $d_v = 128$ and $d_e = 32$.

For the local f_ϕ, its vector argument is broadcast to the 16 patch positions and combined with fixed positional/query embeddings. Two four-head pre-norm attention layers with feed-forward width 256, GELU, and zero dropout are applied, then valid canvas positions are mean-pooled back to width 128. These fixed side inputs are suppressed in the main update equations. Input-grid encoding uses global attention; recursive refinement uses local attention. Both contribute to the computation budget.

Predicting shape and colors. A shared $256 \to 128 \to 10$ MLP reads the proposal concatenated with each positional/query token embedding. Overlapping cell logits are averaged. Mechanical patch padding beyond coordinate 29 is masked. A $128 \to 128 \to 60$ MLP reads the mean final proposal and predicts height and width as two 30-class distributions. Their argmaxes determine the crop of the predicted canvas.

Training combines mean StableMax cell cross-entropy in the target rectangle with 0.1 times the sum of height and width cross-entropies. The target rectangle is used only in the training loss. Exact success requires both the correct dimensions and correct content. The separate shape metric requires every query in the task to have correct dimensions in its top-one candidate; hence exact top-one task success is a subset of shape success.

Training episodes and data separation. The controlled training set combines official ARC-AGI-1/2 training tasks with the 160 ConceptARC tasks (Moskvichev et al., 2023), deduplicated by canonical task content. Tasks whose sorted demonstration-pair content matches an evaluation task are excluded without accessing evaluation query outputs. Canonical hashes omit filenames and sort the demonstration pairs. Tasks with their first eight SHA256 bytes interpreted big-endian modulo ten equal to zero form validation, before augmentation.

Training episodes either hold out one demonstration pair with at least one remaining support pair, or use a released labeled training query with all demonstrations. Tasks are sampled uniformly, then eligible episodes uniformly. Uniform D4 transforms (the eight rotations/reflections of the square) and ten-color permutations are shared by the support grids, query, and supervised output. Checkpoint and aggregation choices use held-out training tasks. Public-evaluation tasks do not enter optimization.

Transformed inputs and candidate aggregation. Evaluation uses nested transform budgets $V \in \{1, 8, 64, 1000\}$. Identity comes first, followed by the other seven dihedral transformations, then seeded D4/color-permutation combinations. Each transform is shared by all grids in an episode, and D4 operations act within each grid's valid rectangle. Later inputs also use independently sampled canvas offsets that preserve every observed cell. Coordinates are relative to the recorded origin; encoded tokens are returned to those coordinates before patch pooling. Padding translation thus checks invariance rather than adding a semantic transformation.

The first eight inputs use zero offsets and identity colors. Later transforms use NumPy PCG64 with seed obtained by XOR of 20260916 and the first eight bytes of the task's observed-input SHA256 hash. The realized list is stored independently of query labels. Predicted rectangular

grids are inverse-transformed with their predicted dimensions intact. Candidates are keyed by $(h, w,$ row-major colors$)$, ranked by frequency, and tied lexicographically. The first two distinct candidates are submitted, or one if all predictions agree. A single transformed input therefore yields only one candidate.

A task is successful only if every required query is correct under the two-attempt rule. Top-one success uses the same candidate ordering. Each transformed input recomputes the predictor, including context encoding. With 100 agents, $R = 16$, $n = 6$, and $T = 5$, each query costs $11200V$ agent-local block applications and $80V$ CG iterations, plus encoding, readout, and aggregation. Parameter accounting includes the context and shape/color heads.

Context controls. The query-only control is retrained without demonstrations. The support-corruption intervention keeps the full checkpoint and input transforms fixed while replacing color c with $(c+1) \bmod 10$ in demonstration outputs only. Demonstration inputs, shapes, query inputs, and candidate ranking remain the same. This separates the effect of removing context during learning from providing misleading context at inference.

D.2.2 Benchmark ARC Interface

All six rows in Table 1 use the controlled task split, training augmentations, test transforms, and candidate ranking in Section D.2.1: $V = 1000$, one trajectory per transform, two distinct candidates, and $R = 16$. Query labels or dimensions never enter the input. The model uses the full 30×30 output canvas with 100 stride-three patch agents, including cells outside the observed query.

The compact benchmark context encoder differs from the controlled Transformer interface. It encodes every observed grid cell using ten-color/PAD, row/column, and input/output-role embeddings of width 32, pools valid tokens in each 4×4 patch, and concatenates corresponding support input/output summaries. A shared $64 \to 128 \to 128$ GELU MLP combines the pair summaries, adding a $4 \to 128$ projection of normalized grid shapes. Mean pooling over valid patches and then over support pairs gives one width-128 context vector; there are no task-ID embeddings. This vector is concatenated with each query-patch summary and position embedding, and the benchmark one-hidden-layer width-512 encoder forms e_i. The local block is width-384, six-head, two-layer local attention with feed-forward width 512, which lifts the width-128 recurrent vector and pools back after the block. Colors are decoded with the benchmark width-512 MLP conditioned on local positions, and shape with the same $128 \to 128 \to 60$ head and 0.1 shape-loss weight as the controlled interface.

The reruns and component comparisons share this context, shape policy, data, and candidate rule, but train separate weights. Their parameter counts include context, modulation, and heads, with each unique tensor counted once. The 2.6M module count in Table 3 excludes the context and shape heads and is therefore separate from this full-model accounting.

D.2.3 MNIST Convolutional Controls and Corruptions

MNIST uses the standard split of 60000 training and 10000 test images (LeCun et al., n.d.); the additional controlled-calibration holdout is specified in Section D.9. The benchmark CNN is clean-trained; the stronger CNN uses adaptive global pooling. Augmented CNN and NRR share exposure to paired corruptions. These controls separate negotiation from robustness supplied by preprocessing and training data. The following definitions fix the architectures, corruption units, and augmentation distribution.

The benchmark CNN has same-padded 3×3 convolutions $1 \to 32 \to 64$, each followed by ReLU and 2×2 max pooling. The corrupted image is resized to 28×28 by bilinear interpolation (no corner alignment) before this fixed-size CNN. The flattened $64 \times 7 \times 7$ features pass through $3136 \to 128 \to 10$ with ReLU and no dropout; including biases gives 421642 trainable parameters. The stronger CNN uses the same convolution stack on the native-size corrupted input, adaptive global average pooling, and $64 \to 128 \to 10$, giving 28426 parameters. No batch normalization is used. This is a preprocessing/architecture control; its smaller parameter count is not a parameter-matched ablation.

All models receive raw image intensities in $[0, 1]$. "Pad p" means p zero-valued pixels on every side, so $p = 4, 8, 16$ yields widths 36, 44, and 60 before model-specific resizing or graph construction.

"Drop q" independently zeros each pixel with probability q, without rescaling surviving pixels. "Noise σ" adds independent $\mathcal{N}(0, \sigma^2)$ noise in this raw scale and clips to $[0, 1]$. Each reported condition applies only one corruption. The zero border is image background rather than missing output; only mechanical patch-completion padding is masked.

Clean CNNs are trained for 20000 updates at batch 128 with AdamW (Loshchilov & Hutter, 2019), learning rate 10^{-3}, weight decay 10^{-4}, and the same five seeds. For augmented CNN/NRR, the sampled original images and corruption draws are paired, and both models see 20000 batches of 128 images. Each draw chooses uniformly among clean, padding, deletion, and noise; padding then chooses $p \in \{4, 8, 16\}$ uniformly, deletion uses $q \in \{.05, .10, .30\}$, and noise uses $\sigma \in \{.10, .20\}$. Each draw makes one training example; this matches augmented-example exposure, not optimizer FLOPs. A separate PCG64 stream is keyed by run seed, example ID, and augmentation index. Evaluation corruptions are frozen for each original test image and condition and shared across models.

D.3 Data Separation and Model Selection

All augmentations of an original puzzle or ARC task stay in the same split. Controlled Sudoku/Maze comparisons use 900 training and 100 validation instances from the released training set; benchmark models use all 1000 training instances and the released test set. Controlled validation determines solver depth, stopping thresholds, checkpoint duration, and hyperparameters before testing. ARC grouping and selection use whole tasks as specified in Section D.2.1.

The controlled search has six learning-rate/weight-decay choices from $\{3 \cdot 10^{-5}, 10^{-4}, 3 \cdot 10^{-4}\} \times \{0.1, 1.0\}$ per model. Component variants retain the encoder, decoder, graph, and state capacity except for the explicitly named intervention. Native TRM and EqR retain their appropriate architectures. Local computation and edge operations are accounted for separately in Section D.5.

D.3.1 Training and Selection Schedule

For controlled Sudoku/Maze, we serialize each original training input as its row-major discrete grid in UTF-8 JSON with compact separators and no filename, sort the SHA256 hashes, and reserve the first 100 of 1000 for validation; runs use the seed set in Section D.4. The grouping and benchmark split follow Section D.3. Models use the graph and state interfaces of the controlled specification unless the benchmark configuration table or Section D.2.2 overrides them.

Benchmark NRR and its component replacements use the settings in Table 2, with $\rho = 1$ and $\lambda = 0$, and the optimizer settings in Section B.3. The fixed optimizer-update totals are 60000 on Sudoku/Maze, 100000 on ARC, and 20000 on MNIST. The first 10% go to local warm start and the remainder to coupled segments, giving 54000, 90000, and 18000 coupled updates. These are update counts, not source-paper epochs. All $M = 16$ segments per batch are executed on Sudoku/Maze/ARC and $M = 8$ on MNIST, with no training halt; if the update allowance ends mid-batch, only its remaining allowed segments are run and the carry is discarded. We use the final EMA checkpoint. The benchmark proximal and MPNN models, which have no RecurProx warm start, use the full allowance for their own supervised updates; this is a duration-matched architecture comparison, not an assertion of equal training FLOPs.

For the controlled experiments, hyperparameters and checkpoint duration are selected on held-out validation data using the six learning-rate/weight-decay choices above. The search uses seed 42; the selected configuration and training duration are then rerun with seeds 123, 456, 789, and 1024, and we report the mean and standard deviation of all five held-out-test scores. Native baselines retain their model-specific update schedules. Auxiliary stopping-head fitting is described separately in Section D.9.1.

D.4 Five-Seed Aggregation and Paired Evaluation

Within-study comparisons use five training seeds: 42, 123, 456, 789, and 1024. Denote this set by $\mathcal{S}$. Each run varies parameter initialization, training-example order, and training augmentation. Dataset partitions, test instances, difficulty bins, test corruptions, and ARC transform lists are fixed across seeds and methods. Benchmark and controlled configurations retain separate checkpoints and aggregation groups. Selection and compute duration follow Section D.3.1. The replicate is the

training seed; original puzzles and whole ARC tasks are the evaluation units. Transformed inputs, correlated queries, rounds, and timing repetitions do not increase the seed count.

Means, dispersion, and comparisons. For a run-level quantity q_s, report $\bar{q} = \frac{1}{5}\sum_{s\in\mathcal{S}} q_s$ and

$$s_q = \sqrt{\frac{1}{4}\sum_{s\in\mathcal{S}}(q_s - \bar{q})^2}.$$

Tables use $\bar{q} \pm s_q$. Curves show means with one-SD ribbons; whiskers show the same interval, and faint points or traces expose individual seeds where space permits. This SD describes between-run variation, not a standard error or a confidence interval. For a paired effect, compute $d_s = q_{A,s} - q_{B,s}$ first, then report $\bar{d} \pm s_d$. The factorial interaction is likewise a four-cell within-seed contrast. Ratios are computed within seed before averaging. Shared reference checkpoints reuse identical run-level scores wherever they appear.

Retraining versus frozen-checkpoint evaluation. Benchmark and controlled baselines, component replacements, recursion/dual factorial cells, gradient windows, solver training budgets, strength and warm-start sweeps, and retrained graph controls each have five independent training runs. Depth sweeps, maze-size transfer, inference-only topology changes, corrupt-support interventions, and MNIST corruption conditions evaluate each corresponding checkpoint across all conditions. They do not treat conditions as independent seeds. For solver residuals, one $T = 5$ checkpoint per seed is frozen and only the numerical iteration budget varies; the accuracy sweep instead retrains for each budget. Each residual p95 and each population-specific correlation is computed within seed, and those five values are aggregated. Difficulty accuracies use identical original-puzzle bins across seeds.

Stopping, ARC, and resource accounting. Stopping comparisons reuse each checkpoint's complete trajectories and preserve the paired identity in (22) before seed aggregation. Fixed horizons have no sampling SD. ARC top-one, two-attempt, and shape metrics share task denominators, checkpoint, and transform ordering within each seed; the outcome partition is $(\mathrm{exact}, \mathrm{shape}{-}\mathrm{exact}, 100{-}\mathrm{shape})$. At $V = 1$, top-one and two-attempt results are identical.

The timing procedure in Section D.5 is applied separately to each seed, and we report the mean and SD of the five latency medians. Peak device allocation is summarized once per seed under the same execution conditions; fixed tensor shapes and execution budgets yield identical allocations across seeds. Trial counts do not serve as independent training replicates. Exact operation counts, model dimensions, parameter budgets, theoretical bounds, and implementation acceptance thresholds are deterministic. Published comparison scores retain their source statistics and are excluded from within-study seed aggregation.

D.5 Resource Counts and Measurement

NRR's interface determines several resource counts. Excluding fixed embeddings, maps, and solver temporaries, it carries $4Nd_v$ state scalars in (x, r, z, u), of which $2Nd_v$ are communication state. Its local forward work is $R(n+1)N$ block applications; retained local work per training segment is $B(n+1)N$. With dense endpoint maps, one application of $F^\top F$ entails four matrix–vector products per undirected edge: two restrictions and two transpose restrictions. Counting a multiply and an add as two FLOPs gives the leading cost $8|E|d_e d_v$, plus edge differences and scatter additions. CG also performs node-vector operations, inner-product reductions, and convergence masking. Recomputing the true residual adds another operator application.

For the stated Sudoku graph, these counts give 31104 carried state scalars, 4536 local block applications in a 24-round trajectory, and approximately 17.92 million leading dense-map FLOPs per application of $F^\top F$. Five CG steps over 24 rounds give approximately 2.15 billion such FLOPs before the remaining costs. These are analytical dense-operation counts, not a measured hardware profile; selector or fused implementations can cost less. Five learned edge-MLP updates and five CG iterations therefore do not establish equal computation. Section C.5 specifies the untied replacement's local computation.

Parameters, state, and executed work. Parameter totals count each shared trainable tensor once, including the encoder, local solver, communication modules, and decoder. Persistent-state capacity is separate from temporary solver buffers. Local network calls, learned message updates, and CG iterations describe different operations and are reported separately. The latency comparisons below measure the complete predictor under the stated hardware and timing conditions.

Benchmark and controlled timing. Benchmark timing uses a median over 1000 synchronized trials after 200 warm-up executions, on one A100 80GB at batch size one. It includes encoding, the full horizon, communication, decoding, and aggregation. Global-RecurProx uses two layers of whole-grid self-attention and uses the same encoder/decoder interface, private-update count, round budget, data, and precision. The 20/26 ms Sudoku and 50/38 ms width-30 Maze pairs belong to the benchmark timing series. The controlled series in Figure 1c, including the 22/28 ms and 52/41 ms pairs and the larger-maze measurements, applies the same procedure to the controlled checkpoints as a separate measurement series. Per-seed aggregation is defined in Section D.4.

D.6 Difficulty Stratification

Each original Sudoku puzzle is assigned one external backtracking-count bin, fixed across seeds. The solver convention is specified below; the solver version and bin manifest are not included in this version (Section D.10). The comparison is interpreted descriptively as performance versus search difficulty.

Deterministic solver. The solver uses row/column/box bitmasks, repeatedly fills naked singles in row-major order, and then branches on minimum remaining values with row-major ties and ascending digits. It counts one backtrack whenever a tried digit branch is exhausted without a solution and does not count successful descents or forced assignments. The plotted boundaries define the bins.

D.7 Depth and Size-Transfer Protocols

The depth experiment varies inference rounds R for checkpoints trained with $K = 16$ or 24; numerical budget T and trajectory count stay fixed. Size transfer evaluates width-30 checkpoints on larger mazes without finetuning. These are frozen-checkpoint condition sweeps, with results in Section E.6.

Larger-maze datasets and checkpoint reuse. Paired size transfer evaluates the same width-30 checkpoint at widths 30, 60, and 90, with the same horizon and no finetuning. Width 30 uses the released test set. For widths 60 and 90, we generate 1000 independent mazes with the released Maze-Hard generator at the new width, minimum path length $\lceil 110W/30 \rceil$, fixed generator seeds $20260920 + W$, and row-major deterministic shortest-path tie breaking. The benchmark TRM transfer and controlled Global comparison share these test instances but retain separate training configurations.

D.8 Residuals and Error Localization

Negotiation explicitly maintains proposal–target mismatch, target change, and overlap discrepancy; an unconstrained learned link does not prescribe these quantities. Their interpretation depends on the readout, as shown in Section A.7. The following populations and denominators distinguish error severity and localization from stopping decisions, whose protocol appears in Section D.9.

Fixed-horizon population. The evaluation uses all original test instances at the fixed final horizon, separately by seed: 422786 Sudoku puzzles at $R = 24$ and 1000 Maze grids at $R = 30$. Each untransformed instance is evaluated once without early stopping. Augmentations and intermediate rounds do not increase this sample size. Failed numerical solves remain accuracy failures and are flagged in residual analyses.

Residual and error variables. For $D_v = Nd_v$ and $D_e = |E|d_e$, define

$$\varrho_p^k = \frac{\|x^k - z^k\|}{\sqrt{D_v}}, \quad \varrho_d^k = \frac{\rho\|z^k - z^{k-1}\|}{\sqrt{D_v}}, \quad \varrho_F^k = \frac{\|Fx^k\|}{\sqrt{D_e}}, \quad s_v^k = \frac{\max(\|x^k\|, \|z^k\|)}{\sqrt{D_v}}.$$

The normalized variables are $\widehat{\varrho}_p = \varrho_p/(s_v + 10^{-8})$, $\widehat{\varrho}_d = \varrho_d/(\rho s_v + 10^{-8})$, and

$$\widehat{\varrho}_F = \frac{\|Fx\|}{\|F\|_{\mathrm{F}}\|x\|/\sqrt{D_v} + 10^{-8}}.$$

The sheaf normalization removes global map scale up to its numerical floor. Empty or zero maps have sheaf residual zero and a degeneracy flag. Hamming error err counts mismatched labels on all 81 Sudoku cells, including givens, or all 900 Maze cells against the reference path mask. Exact correctness is $\mathrm{err} = 0$.

Correlations and ties. Spearman correlation is Pearson correlation of average ranks, with exact equality defining ties. Zero-variance ranks give an undefined correlation. All-instance correlations and failed-only correlations ($\mathrm{err} > 0$) use different populations: the latter measures severity conditional on failure. Any trajectory analysis is separate and clusters samples by original instance. Bootstrap intervals use 2000 instance resamples with seed 20260916; failed-only resampling uses the fixed failure subset, and intervals with more than 5% undefined draws are omitted.

If residual ranks are untied and n_0 of the n Hamming errors are zero, the correlation cannot exceed

$$\sqrt{1 - \frac{n_0^3 - n_0}{n^3 - n}}. \tag{21}$$

Untied ranks have variance $(n^2 - 1)/12$; averaging the block of n_0 zero-error ranks removes $(n_0^3 - n_0)/(12n)$. Sorting both rankings together maximizes covariance, giving the bound, while further error ties can only lower it. The controlled exact accuracies 94.8/90.1% give approximate ceilings 0.385/0.518. For tied residual ranks, the attainable maximum is instead computed from the two observed rank multisets, $\mathrm{corr}(\mathrm{sort}(R_\varrho), \mathrm{sort}(R_{\mathrm{err}}))$.

Localization and low-residual mistakes. A failed Sudoku cell receives the mean squared discrepancy of its three semantic overlap edges. For Maze, each edge's squared discrepancy is distributed uniformly over its shared cells and then averaged across incident contributions. This is an attribution rule for pooled patch states. The top $\lceil 0.2|\mathcal{V}| \rceil$ cells per failed instance are selected, with row-major tie breaking. Micro recall divides selected erroneous cells by all erroneous cells in failed instances. The low-residual error rate instead selects the $\lfloor n/2 \rfloor$ instances with smallest $\widehat{\varrho}_p$, breaking ties by instance ID, and divides wrong answers by this subset size.

D.9 Stopping Rules and Paired Evaluation

All stopping rules return the current proposal on the same frozen checkpoint trajectories. The fixed-horizon rule runs to R; adaptive rules start no earlier than round four, with maxima 24/30 for controlled Sudoku/Maze. The residual rule requires all three normalized thresholds from Section D.8 to pass at each of $k-2, k-1, k$. If no predicate passes, evaluation returns the full-horizon answer. Stopped residuals are not reused as the fixed-horizon correlation population.

For any stopping time $\tau \le R$, define added and recovered error on all original test instances. Their paired identity is

$$A_R - A_\tau = \Pr(\widehat{Y}_R = Y, \widehat{Y}_\tau \neq Y) - \Pr(\widehat{Y}_R \neq Y, \widehat{Y}_\tau = Y). \tag{22}$$

We report both terms, net accuracy change, and mean rounds together, with within-seed aggregation as in Section D.4. The identity checks arithmetic but does not determine the stopping thresholds.

D.9.1 Comparison Rules and Calibration

Every round's states and logits from the selected checkpoint's full trajectory are cached. Confidence, entropy, and the auxiliary head use the following reducers and interfaces. With StableMax probabilities $p^k_{v,c}$ of the mean logits, we use all 81 Sudoku cells to compute

$$C_k = \min_{v \in \mathcal{V}} \max_c p^k_{v,c}, \qquad H_k = -\frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \sum_c p^k_{v,c} \log p^k_{v,c}.$$

Confidence requires $C_k \geq \theta_C$ for three consecutive rounds; entropy requires $H_k \leq \theta_H$ for three rounds. Answer stability requires $\widehat{Y}_k = \widehat{Y}_{k-1} = \widehat{Y}_{k-2}$. Complete validity checks the current grid's 81 cells, all 27 digit sets, and every given clue immediately, without patience. On unique-solution Sudoku this certifies a solution; confidence, stability, and latent agreement alone do not.

For the auxiliary correctness head, the backbone is frozen and a $5d_v + 2 \to 128 \to 1$ GELU MLP with sigmoid output is trained. Its features are mean (e, x, r, z, u) over agents, k/R, and mean cell entropy. All features are detached, and the head is trained with binary cross-entropy against current whole-grid correctness on the 900 training originals only, with rounds sampled uniformly from $4, \ldots, R$. Training uses AdamW, learning rate 10^{-3}, weight decay 10^{-4}, batch 256, at most 2000 head-only updates, and the run seed. Its loss does not modify the backbone, so all stopping rules evaluate the same trajectories. The rule stops after the head score exceeds θ_{corr} in three consecutive rounds, and its training and inference cost is counted separately.

Residual calibration searches the Cartesian product of the three 10th–90th validation-percentile grids in ten-point increments, and each scalar comparison rule uses the same percentile grid. For each candidate, we evaluate the actual first-hitting stopping time on the 100 held-out originals, with no stopping included as a candidate. A candidate is admissible only if it adds at most 0.5 percentage points of error on this validation set; with 100 originals, this means zero observed added errors. Among admissible candidates, calibration selects the fewest mean rounds and resolves ties by smaller added error and then lexicographic thresholds. Answer stability and validity have no fitted thresholds; their fixed rule is admitted only if it satisfies the same constraint, and otherwise the no-stopping fallback is used. The selected thresholds are frozen per checkpoint and seed before test evaluation. Thus 0.6 points of test added error for stability can coexist with zero observed validation added error.

Benchmark stopping. Benchmark stopping transfers the normalized residual thresholds selected on the corresponding controlled validation split, without recalibrating on test labels or selecting a new benchmark checkpoint. Benchmark models train on all released training instances, including these validation originals, so the transferred thresholds are not a held-out calibration for the benchmark checkpoints, although test labels are never used. Evaluation uses benchmark maxima 24/30/10 for Sudoku/Maze/MNIST. The MNIST controlled calibration reserves the first 1000 SHA256-sorted original training images and uses the same added-error criterion; this is distinct from the 100-instance Sudoku/Maze validation.

D.10 Records Not Included in This Version

Some records needed for exact reproduction are not included in this version: the sorted split-ID/hash manifest for the controlled Sudoku/Maze split (Section D.3.1); the generator version and generated grid/reference-path manifests for the width-60/90 mazes (Section D.7); and the per-puzzle difficulty-bin manifest, per-bin sample sizes, solver version, and count-weighted recovery of the aggregate accuracy (Section D.6). The width-45 results in Figure 9 additionally lack a reported sample count, generator version and seed, path-length and tie-breaking specification, and checkpoint and horizon record; the width-60/90 protocol does not establish these details for width 45. Acceptance thresholds for the implementation checks in Section B.7 are kept in the authors' verification notes rather than reported as experimental outcomes.

E Additional Experiments and Ablations

These analyses extend the main comparisons with coupling, difficulty, mechanism, numerical, and task-specific results. Benchmark results use the setting of Table 1; controlled results share the 94.8%/90.1% NRR reference of Figure 1b. Task and baseline definitions precede the results in Sections C and D.1, and evaluation and seed aggregation are specified in Sections D and D.4.

E.1 Interaction and Search Difficulty

Figure 5 distinguishes incomplete local prediction from independent recursion with all-view voting; both differ from communication during inference. The all-view vote covers every output location with the same reducer as NRR and still has low grid accuracy. The local-only baseline uses incomplete output coverage. On single-label MNIST, the end-of-run vote already reaches 78.6%, whereas

incomplete local prediction solves neither Sudoku nor Maze. This contrast helps explain why final aggregation can integrate classification evidence yet leave mutually dependent output decisions unresolved.

Figure 6 gives the complete binwise accuracies behind the difficulty trend in the main text. Backtracking revisits an earlier tentative assignment after search encounters a contradiction. These externally defined bins characterize search effort, without identifying a specific internal NRR operation. Section D.6 specifies the solver convention; the bin manifest is not included in this version (Section D.10).

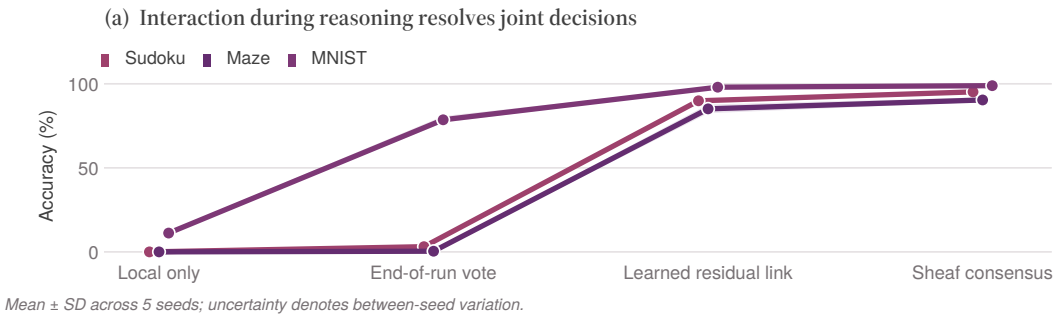


Figure 5: **Interaction during reasoning versus final aggregation.** The same local views are combined through incomplete prediction, an end-of-run vote, learned residual communication, or sheaf consensus. MNIST aggregates evidence for one label; Sudoku and Maze require joint structured outputs. Marks and annotations report five-seed means; whiskers and shaded bands denote ± 1 SD. Differences and ratios are formed within seed.

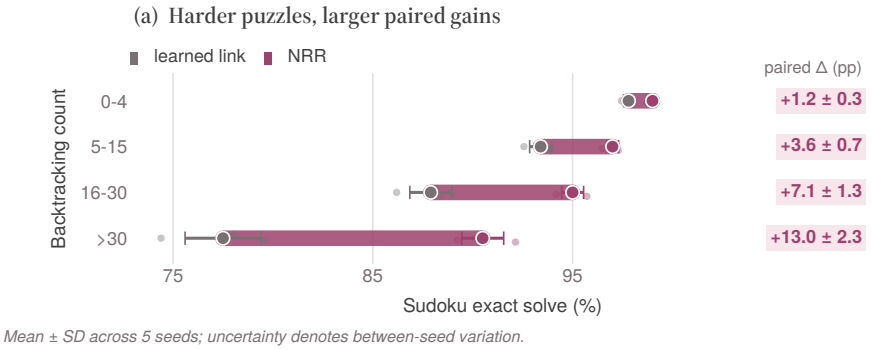


Figure 6: **Difficulty-stratified Sudoku results.** Bins use an external solver's backtracking count. Accuracy gains grow with search effort; this is a descriptive comparison, whose reproducibility requirements are given in Section D.6. Thick connectors join mean accuracies; whiskers show ± 1 SD. Gain labels report paired mean $\pm$ SD over five seeds.

E.2 Factorial Effects and Graph Interventions

The factorial varies the local solver and disagreement memory while retaining the data, decoder, and restrictions. The local-solver variants are defined in Section C.5. On Sudoku/Maze, adding disagreement memory improves feed-forward solving by 3.7/2.7 points and private recursion by 5.2/3.6 points. The within-seed interaction is $(A_{\text{rec,dual}} - A_{\text{rec,no dual}}) - (A_{\text{ff,dual}} - A_{\text{ff,no dual}})$; its mean and SD are $1.5 \pm 1.3/0.9 \pm 4.2$ points. Memory improves both solvers on average, with a larger mean gain when local computation repeatedly refines its state. Private recursion improves a proposal, while accumulated discrepancy makes subsequent refinement respond to unresolved conflicts.

Degree-preserving overlap shuffling reduces retrained Sudoku accuracy from 94.8% to 86.0%; applying the shuffle only at inference reduces it to 62.0% (Figure 3b). Retraining tests the value of the

structure after the model adapts, whereas the inference intervention tests reliance on a learned interface. Keeping half the edges gives 90.5% and 82.0%, respectively. Meaningful overlap relations therefore matter beyond allocating additional recurrent states. Together with the global-recursion comparison, these controls are consistent with a simple account of why decomposition can help: it exposes which proposals must agree, and negotiation feeds their conflicts back into reasoning.

E.3 Additional Communication and Gradient Controls

The controlled comparisons below complement the factorial and topology experiments in Figure 3. They compare explicit negotiation to stronger graph-based communication and vary the path from supervision to the coupling parameters.

The component definitions in Section C.4 distinguish the benchmark learned-link replacement from the stronger persistent-link control. The latter maintains communication memory and executes five substeps per round.

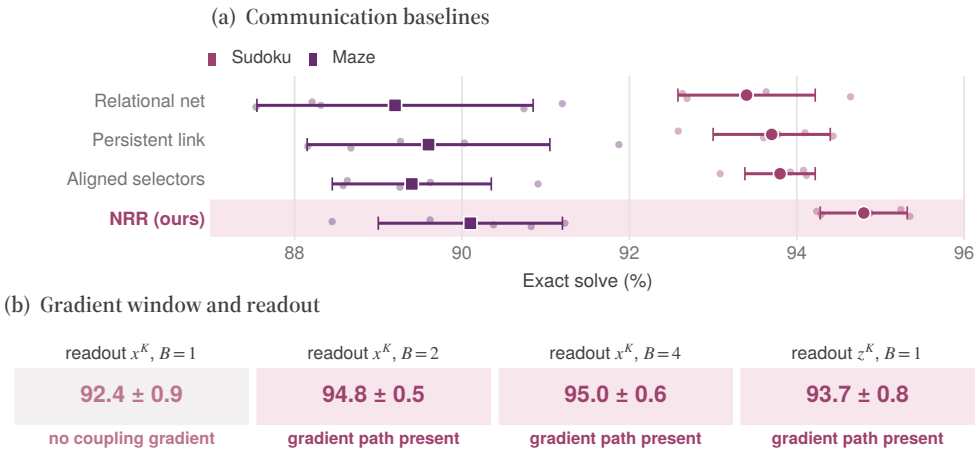


Figure 7: **Communication baselines and gradient connectivity.** (a) Graph-based communication baselines; native global models appear in Figure 1b. (b) Retained gradient window and readout: task-gradient connectivity is analytical, while accuracy is measured after separate training. The controlled NRR reference is shared with Figure 3. In (a), points and whiskers show mean ± 1 SD over five seeds. Cards in (b) report mean $\pm$ SD accuracy.

Beyond the one- versus two-round comparison in the main text, retaining four rounds yields 95.0% Sudoku accuracy. Alternatively, reading the consensus target after one retained round restores task-gradient connectivity and reaches 93.7%. Longer retention and changing the readout are distinct interventions: the former includes more inter-round interactions, while the latter supervises a different state. These controls support retaining the preceding negotiation step when the output is the proposal.

E.4 Local Computation, Communication, and Decomposition

The component ablations in Figure 8a–c change one part of the benchmark model at a time. They ask three questions: does the result depend on the recursive local solver, on how proposals communicate, and on how the problem is divided into views?

Replacing the recursive local solver causes a large loss: with the proximal-solver comparator, Sudoku accuracy falls from 95.2% to 71.6%. This comparator reuses the smaller-budget Sheaf-ADMM rerun in Table 1, so the drop reflects both the form of the solver and its smaller budget. Figure 3a gives a cleaner comparison between recursive and untied factorized local solvers, with and without disagreement memory. Removing local warm start costs less, lowering Sudoku accuracy to 92.7%.

How proposals communicate also matters. Replacing conjugate-gradient consensus with gradient-descent diffusion lowers accuracy on all three tasks, to 90.3% on Sudoku. Removing disagreement

memory gives 89.5%, 85.8%, and 98.1% on Sudoku, Maze, and MNIST, compared with 95.2%, 90.4%, and 98.9% for the full model. Input-dependent low-rank modulation of the restrictions has a smaller effect: without it, accuracy is 94.8%, 87.8%, and 97.9%.

The largest loss comes from the restrictions. With unaligned identity maps, Sudoku accuracy drops to 17.6%. The aligned-selector control separates alignment from learning: fixed selectors that compare the correct shared cells reach 93.8%, close to the 94.8% of learned restrictions. Two views order their local cells differently, so an identity map generally compares different cells across an edge and enforces the wrong equalities. Most of the benefit therefore comes from comparing the right quantities, with learning adding about one point.

Finally, the choice of views matters. On Sudoku, views defined by constraint groups (rows, columns, and boxes) reach 95.2%, compared with 91.4% for overlapping patches. Views that expose the task's actual constraints thus make negotiation more effective. Maze and MNIST already use spatial patches, so the constraint-group variant does not apply to them.

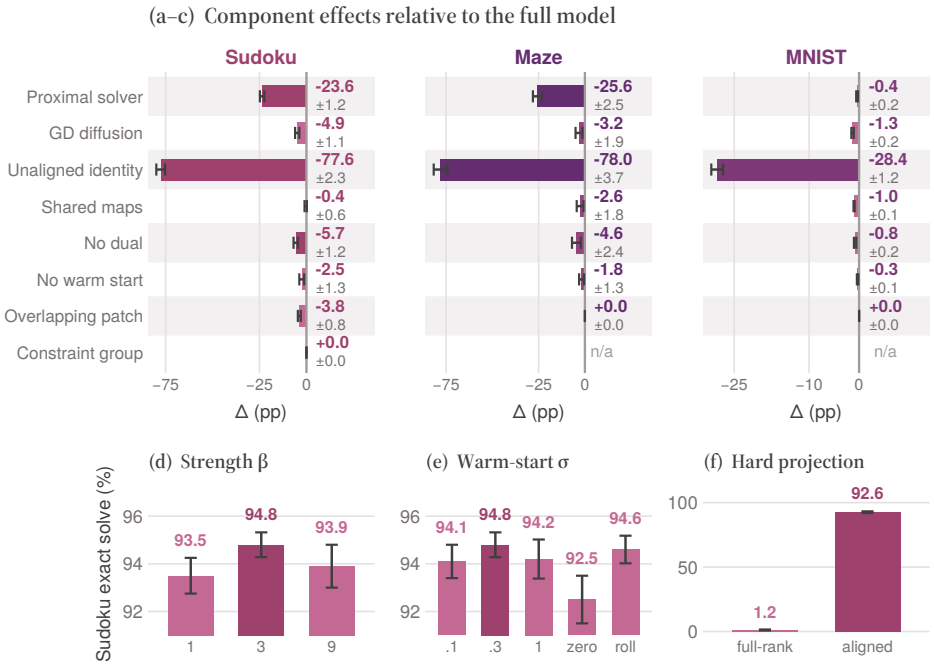


Figure 8: **Component, strength, and initialization ablations.** (a–c) Benchmark changes from the full model on each task. The proximal solver has a smaller parameter budget; unaligned identity maps and correctly aligned selectors are distinct controls. Sudoku constraint groups are not applicable to the patch tasks. (d–f) Controlled Sudoku experiments with separately trained variants and validation-based selection. Soft strength changes the balance between preserving proposals and enforcing compatibility; hard projection depends on the restriction operator's nullspace; warm-start variants change initialization. Bars show five-seed means; whiskers denote ± 1 SD. Component labels give paired mean changes and SDs.

E.5 Consensus Strength and Initialization

The controlled Sudoku experiments in Figure 8d–f vary two settings that the component ablations hold fixed: how strongly consensus is enforced and how the local solver is initialized.

The consensus strength β controls a tradeoff in (13). A larger β pulls the target closer to exact agreement, but it also moves the target further from the proposals and makes the linear system harder to solve. Accuracy is highest at an intermediate value: 93.5%, 94.8%, and 93.9% for $\beta = 1, 3, 9$. Enforcing agreement more strongly is therefore not enough on its own; the target must also keep useful information from the local proposals.

Hard projection is the limit of infinite strength, where the target is forced exactly into the consensus space. Whether this works depends on what that space retains. With generic full-rank restrictions, the consensus space contains only zero, so the projection removes all information from the target and accuracy falls to 1.2%. It is not exactly zero because local inputs and private states still reach the decoder. With aligned selectors, the consensus space keeps one shared representation for each cell, and hard projection reaches 92.6%.

Warm start pretrains the local solver to refine anchors, which are noisy embeddings of the local targets (Section B.6). The most important property is that these anchors carry information about the answer. Zero anchors give the lowest accuracy, 92.5%. Noisy targets give 94.1%, 94.8%, and 94.2% for noise levels $\sigma = 0.1, 0.3, 1.0$, and anchors taken from detached rollouts of a pilot model give 94.6%. Once the anchors are informative, the noise level has a smaller effect, with the moderate setting $\sigma = 0.3$ performing best. This matches the role of warm start: during negotiation, the local solver receives anchors that are informative but imperfect, so it benefits from being trained on anchors of the same kind.

E.6 Depth and Size Transfer

We examine how NRR responds to longer inference and larger inputs. We first vary the number of inference rounds for fixed Sudoku checkpoints, then evaluate models trained on 30×30 mazes at larger sizes and compare their inference costs.

Accuracy depends on training and inference depth. We evaluate benchmark Sudoku checkpoints at different numbers of inference rounds, starting each run from zero-initialized reasoning states. For the checkpoint trained with $K = 24$ rounds per segment, accuracy rises from 74.5% at eight test rounds to 95.2% at 24 and 95.7% at 32, then changes little at 48 (95.5%). A model trained with $K = 16$ falls from 92.1% at 24 test rounds to 90.9% at 48 (Figure 9a). Repeated negotiation can improve a solution, but its useful horizon depends on training.

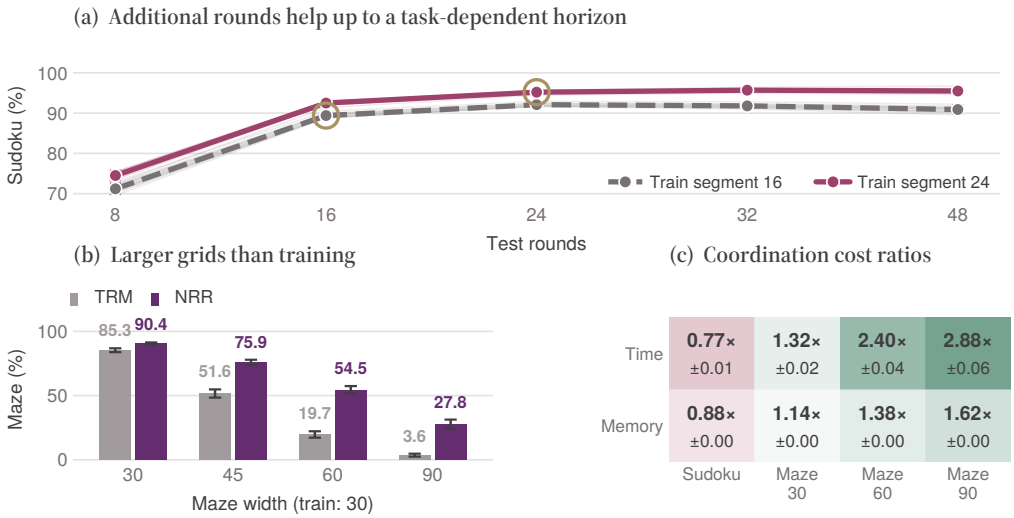
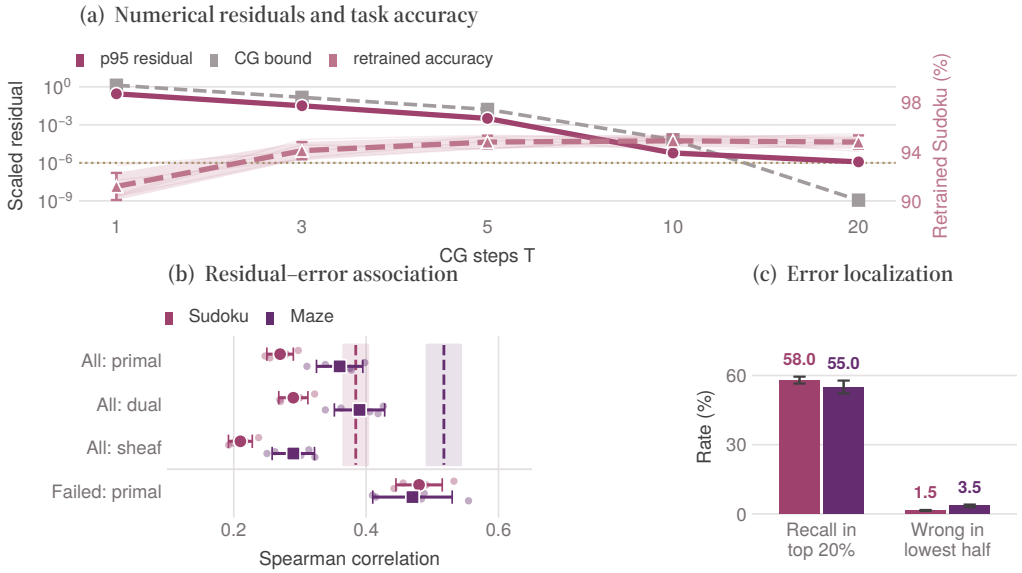


Figure 9: **Depth and problem size expose different computational tradeoffs.** (a) Sudoku accuracy versus test rounds. Gold rings mark the horizon of one training segment; states can carry across 16 segments. Both one-round reset evaluations score zero across seeds and are omitted from the plot. (b) Benchmark maze size generalization after training at width 30. (c) Benchmark timing series: within-seed Global-RecurProx cost divided by NRR cost; ratios above one favor NRR. The accuracy baseline in (b) is TRM, whereas the cost baseline in (c) is Global-RecurProx, which uses global attention recurrence. Curves and bars show means with ± 1 SD over five seeds. Heatmap cells in (c) give the mean ratio above its SD.

Sparse coordination becomes more attractive as the grid grows. After training on 30×30 mazes, NRR achieves 75.9%, 54.5%, and 27.8% exact solve at widths 45, 60, and 90; TRM achieves 51.6%, 19.7%, and 3.6% (Figure 9b). The widening gap shows better size transfer for NRR. In the benchmark timing series (Figure 9c), Global-RecurProx, which uses global attention recurrence, takes $2.40\times$ and $2.88\times$ as long as NRR at widths 60 and 90. This series is separate from the controlled accuracy–latency comparison in Section 5, which reports a $2.80\times$ ratio at width 90. On Sudoku the ratio is $0.77\times$, reflecting coordination overhead on a small input.

E.7 Numerical Accuracy and Prediction Errors

Numerical precision and task quality saturate at different rates. Here the scaled linear-system residual is $s_T = \|w - Az_T\| / \max(1, \|w\|)$; p95 denotes its 95th percentile, pooled over instances and rounds within each seed. Increasing T from three to five reduces the frozen-checkpoint p95 linear-system residual from 0.0320 to 0.0032, while independently retrained Sudoku accuracy improves from 94.1% to 94.8% (Figure 10a). Across the complete one-to-twenty-step sweep, residuals fall by more than five orders of magnitude and accuracy ranges from 91.2% to 94.9%, a 3.7-point range. Beyond five steps, accuracy stays within 0.1 point despite further residual reduction. This is the distinction predicted by (13): reducing numerical error eventually leaves the learned proposal dynamics and finite consensus strength as the relevant factors.



Mean ± SD, 5 seeds. Residual p95 is computed within seed; CG bound assumes exact arithmetic.

Figure 10: **Numerical accuracy and prediction errors.** (a) Residuals from one frozen checkpoint per seed and independently retrained task accuracy across solver budgets; the exact-arithmetic bound is shown separately. (b) Residual–error correlations on all instances and on failed instances. Dashed lines are approximate all-instance Spearman ceilings for untied residual ranks; pale strips show their between-seed variation. (c) Error localization and wrong answers among the lowest-residual half of instances. The controlled NRR reference is 94.8% Sudoku and 90.1% Maze. Marks and annotations report five-seed means; whiskers and shaded bands denote ± 1 SD. Differences and ratios are formed within seed.

Residuals localize some errors without certifying correctness. The top 20% of locations ranked by discrepancy contain 58.0% of Sudoku errors and 55.0% of Maze errors. Among the half of test instances with the lowest normalized primal residual, 1.5% of Sudoku predictions and 3.5% of Maze predictions are incorrect (Figure 10c). All-instance correlations are evaluated at the fixed final round and separately from failed-only correlations. High exact accuracy produces many tied zero errors; Section D.8 derives the resulting Spearman ceilings and defines the evaluation populations. These observations support disagreement as a useful allocation and localization signal, with task validity remaining a separate check.

E.8 Can Disagreement Guide Adaptive Stopping?

The rules and calibration are specified in Section D.9. A fixed horizon spends the same number of rounds on easy and hard instances. NRR's explicit residuals offer an alternative: stop when proposals, targets, and overlap relations have stabilized. The analysis also shows why this decision needs a risk evaluation, since a stable latent configuration can decode to an incorrect answer.

In the benchmark setting, residual-based stopping reduces mean Sudoku rounds from 24 to 17.9 while accuracy changes from 95.2% to 95.0%. Maze changes from 30 to 22 rounds and 90.4% to 90.1%; MNIST changes from 10 to 6.4 rounds and 98.9% to 98.8% (Figure 11c). These correspond to 25.4%, 26.7%, and 36.0% fewer rounds. The controlled Sudoku evaluation compares multiple stopping rules on the same trajectories. The three-residual rule uses 18.5 rounds on average, saving 22.9%, and reaches 94.5% against the 94.8% fixed-horizon reference.

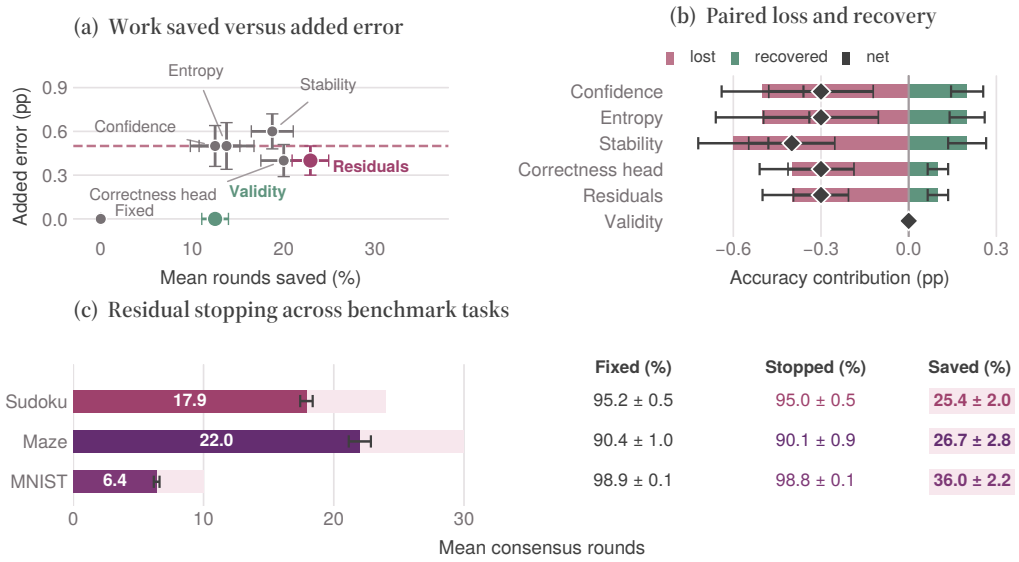


Figure 11: **Adaptive stopping trades computation against changed answers.** (a,b) Controlled Sudoku results. Added error counts full-horizon successes lost by stopping; recovered error counts the reverse, and their difference gives the net accuracy loss. (c) Benchmark mean round counts and accuracies. Round savings measure recursive work; wall-clock savings additionally depend on batching. The certification distinction follows Section D.9. Marks show five-seed means; whiskers denote ±1 SD. Loss, recovery, net change, and savings are formed within seed.

The controlled rule's 0.3-point net loss consists of 0.4 points of added error offset by 0.1 recovered points. Other learned or confidence-based rules also lose some full-horizon successes; the complete Sudoku validity check preserves the reference accuracy. Separating added from recovered errors matters because a small net change can conceal a larger set of altered decisions.

E.9 Demonstration-Conditioned and Single-Label Tasks

The constraint-solving experiments establish the main use case. We now examine tasks where the relation between local evidence and the answer is less directly specified, to understand how far the same organization transfers.

ARC requires demonstrations as well as coordination. The benchmark ARC scores are 52.6% and 14.4% at 1000 transformed inputs. Their relative error reductions over the learned-link replacement are 8.5% on ARC-1 and 3.2% on ARC-2, smaller than on the explicit constraint tasks. In the separately evaluated demonstration-conditioned interface, task success increases from 31.0%/5.2% with one input to 50.0%/12.5% with 1000 transforms and two attempts (Figure 12a). All transformed evaluations contribute to the inference budget. At 1000 transforms, corrupting demonstra-

tion outputs reduces success by 28.8/8.0 points; a query-only model retrained without demonstrations reaches 8.0%/1.8%. Thus the solver uses the context that defines the task, not simply the query's spatial structure.

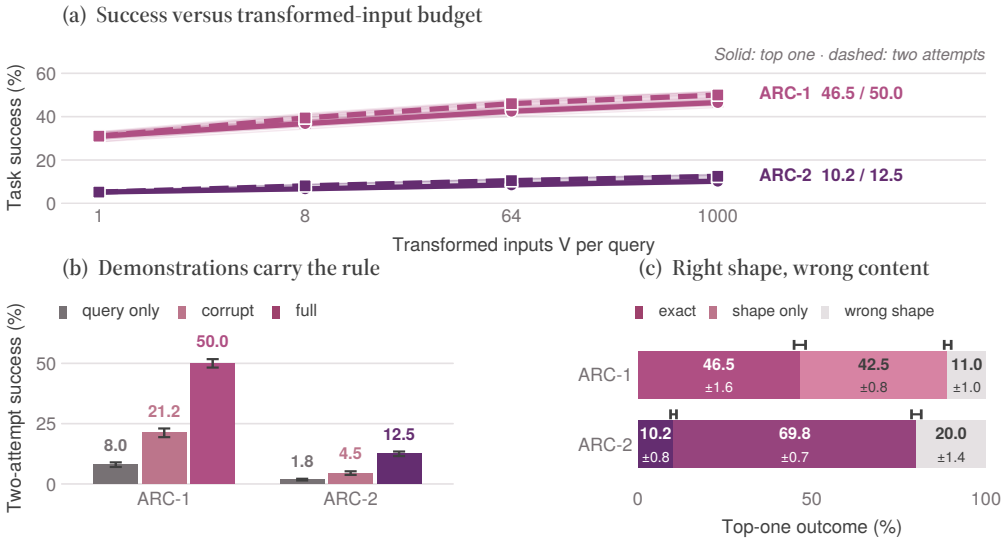


Figure 12: **ARC conditioning and transformed-input budgets.** (a) Solid curves give top-one success and dashed curves give two-attempt success. Shaded bands show between-seed SD. Each query costs $11200V$ agent-local block applications and $80V$ CG iterations, plus encoding and decoding, for V transformed inputs. (b) Query-only retraining and corruption of demonstration outputs on the full checkpoint. (c) Top-one outcomes partitioned by output shape and content correctness. Bars and curves show five-seed means with ± 1 SD. Stacked cells give each outcome fraction and its paired SD; whiskers above stacks give cumulative-boundary SDs.

Correct output dimensions are much easier than exact content: at 1000 transforms, 42.5 and 69.8 percentage points of ARC-1/2 outcomes have the correct shape but incorrect cell values. Negotiation can coordinate a proposed output across patches, but correct content remains much harder to obtain than correct shape.

MNIST separates robustness from constraint reasoning. On the single-label task introduced in Section D.1, NRR reaches 98.9% clean accuracy, compared with 99.3% for the benchmark CNN. Under 16-pixel padding and noise with $\sigma = 0.20$, it retains 89.5% and 38.7%, compared with 11.4% and 9.9% for that CNN. Its gain over Sheaf-ADMM also grows from 0.4 points on clean data to 6.8 under the strongest noise (Figure 13a,b).

The stronger controls clarify the interpretation. An adaptive-pooling CNN substantially closes the padding gap, and under equal augmentation budgets the augmented CNN matches or exceeds NRR in every tested condition (Figure 13c). The strongest case for NRR is therefore coordination of structured output decisions; robustness on a single-label task also depends on the encoder, preprocessing, and training distribution.

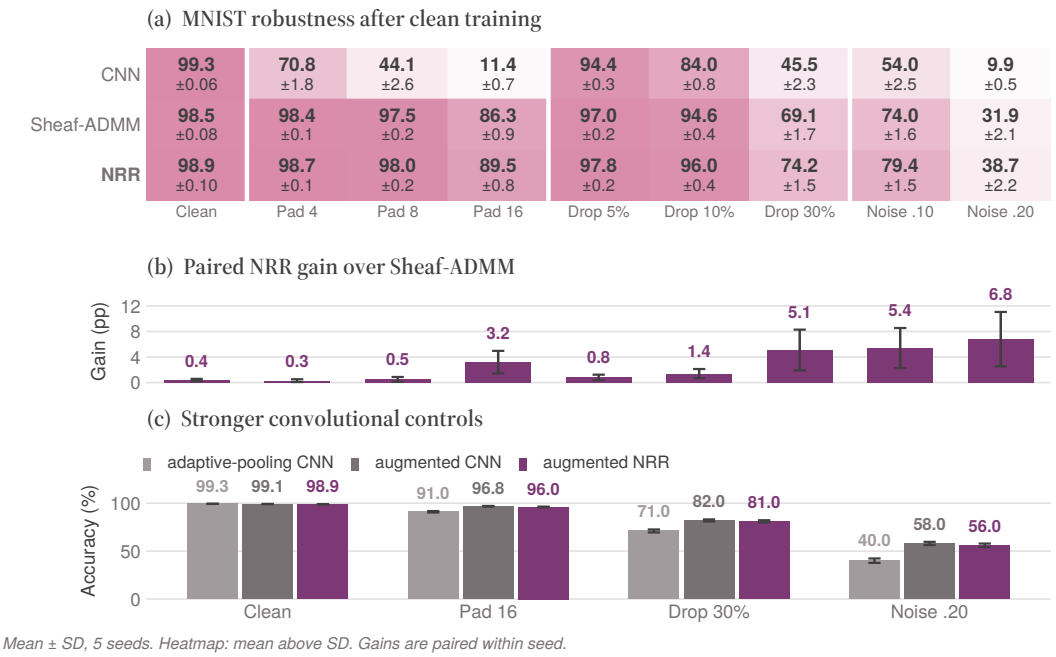


Figure 13: **Robustness depends on the comparison setting.** (a) Clean-trained models across padding, dropout, and noise conditions on one absolute accuracy scale. (b) NRR minus Sheaf-ADMM in percentage points. (c) Stronger convolutional controls: adaptive pooling with clean training, and CNN/NRR under the same augmentation budget. Heatmap cells give mean above SD; bars show means with ± 1 SD over five seeds. Gains in (b) are paired within seed.

F Related Work and Published Comparisons

NRR brings together three lines of work: parameter sharing across reasoning steps, relational computation over structured inputs, and optimization-inspired coordination of local states. Their intersection clarifies the central design choice in this paper: how a recursive local solver should interact with other partial views of the same answer.

F.1 Connections to Prior Work

Recursive computation for reasoning. Universal Transformers, looped learning algorithms, and deep equilibrium models develop weight sharing across depth (Dehghani et al., 2019; Yang et al., 2024; Bai et al., 2019). Task-level recursive architectures and their reported results are discussed together in Section F.2. Energy guidance, diffusion curricula, and dynamical depth control offer further ways to improve iterative solving (Zhao & Tang, 2026; Drozdova et al., 2026; Viakhirev et al., 2026). NRR studies a complementary axis: distributing recursive state across overlapping views and explicitly coordinating their proposals. Our private-state update is motivated by tiny recursion, while the supervised path extends across negotiation rounds.

Relational and algorithmic reasoning. Recurrent Relational Networks solve Sudoku through repeated graph messages (Palm et al., 2018); gated graph networks, MPNNs, and NeuroSAT provide broader recurrent formulations for relations and constraints (Li et al., 2016; Gilmer et al., 2017; Selsam et al., 2019). Easy-to-hard recurrent learning studies extrapolation and overthinking (Schwarzschild et al., 2021; Bansal et al., 2022), and looped Transformers can simulate graph and hypergraph algorithms (Back de Luca & Fountoulakis, 2024; Huang et al., 2025). This work establishes local iterative communication as a reasoning primitive. NRR organizes that communication through separate proposal, consensus, and disagreement states; controlled graph-network comparisons test the value of this organization.

Optimization and compatibility between local views. Factor graphs, belief propagation, and dual decomposition coordinate overlapping subproblems (Kschischang et al., 2001; Sontag et al., 2011). ADMM supplies alternating local updates and explicit dual variables (Boyd et al., 2011). Cellular sheaves represent compatibility between projected local states, and neural sheaf diffusion learns the corresponding maps (Curry, 2014; Hansen & Ghrist, 2019; 2021; Bodnar et al., 2022; Hanks et al., 2025). Sheaf-ADMM applies learned restrictions and primal–dual coordination to overlapping views (Seely et al., 2026). Building on that coordination framework, NRR makes local computation a shared, stateful recursive process and studies how recursion, disagreement memory, finite communication, and task supervision interact.

Algorithm unrolling, OptNet, differentiable convex layers, and SATNet likewise connect optimization structure with learning (Gregor & LeCun, 2010; Monga et al., 2021; Amos & Kolter, 2017; Agrawal et al., 2019; Wang et al., 2019). Plug-and-Play and Consensus Equilibrium accommodate learned operators without requiring an explicit energy (Venkatakrishnan et al., 2013; Buzzard et al., 2018). This perspective is relevant to RecurProx: its anchor is proximal-inspired, while its private state and supervised training define a learned dynamical update. Our analysis therefore distinguishes the implemented finite computation from the exact convex-optimization special case.

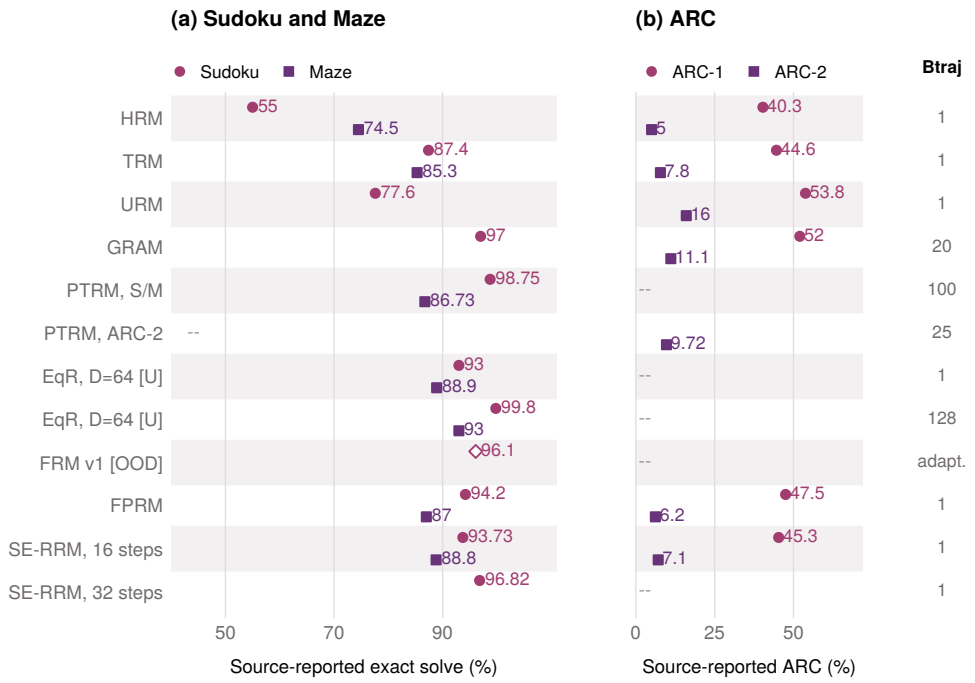
Latent interaction at other scales. Graph Machine learns edge mechanisms, and readout feedback steers recurrent models with prediction-dependent forces (Hou, 2026; Kamiya et al., 2026). Latent recurrence has also been studied in language models (Geiping et al., 2025; Zhu et al., 2025), by retrofitting pretrained models (McLeish et al., 2025), and with pretrained Qwen backbones for symbolic reasoning (Manzari et al., 2026). Blayney et al. (2026) analyze the cyclic latent dynamics and attention behavior of looped language models. RecursiveMAS, LatentMAS, and thought communication exchange hidden representations among language-model agents (Yang et al., 2026; Zou et al., 2025; Zheng et al., 2025). In NRR, the interacting units instead share one small predictor's weights and correspond to explicit local views of a structured output.

F.2 Comparison with Published Recursive Reasoners

The main text compares methods within a shared experimental setting. Published results answer a complementary question: where do these accuracies sit in the broader recursive-reasoning literature? Figure 14 reports the source values together with trajectory budgets; NRR's within-study scores appear in Table 1 and Figure 1b. NRR is competitive with compact single-trajectory approaches on constraint solving, while methods using other training, selection, or restart procedures achieve higher scores in some settings.

Methods represented in the published comparison. Hierarchical Reasoning Models (HRM; Wang et al., 2025) coordinate two recurrent modules operating at different time scales. TRM simplifies this organization using one small network for private and answer updates; EqR learns solution attractors (Section C.1). The Universal Reasoning Model (URM; Gao et al., 2025) uses a looped Transformer with convolution-enhanced gating and truncated backpropagation through loops. These methods refine a global latent representation rather than negotiating overlapping local proposals.

Generative Recursive reAsoning Models (GRAM; Baek et al., 2026) learn stochastic latent transitions and sample multiple trajectories, with a learned reward selector in the reported comparison. Probabilistic TRM (PTRM; Sghaier et al., 2026) adds noise to pretrained TRM rollouts at inference. Its quoted Sudoku result selects the highest Q-scoring rollout, whereas its Maze result selects the modal answer; the two tasks also use different depths. Flow Reasoning Models (FRM) refine a solution through learned flow updates. We cite version 1 (Helbling et al., 2026b) and version 3 (Helbling et al., 2026a) of the same preprint; Table 5 distinguishes their results and evaluation settings. Fixed-Point Reasoners (FPRM; Movahedi et al., 2026) study stable recurrent refinement and adaptive halting. Symbol-Equivariant Recurrent Reasoning Models (SE-RRM; Freinschlag et al., 2026) build symbol-permutation equivariance into recurrent solving; the ARC values are reported with the 16-step setting, while the 32-step row lists only its corresponding Sudoku result.



U: Maze-Unique. OOD: Sudoku-Shah training, Sudoku-Extreme evaluation.
ARC: source-specific protocols; URM uses pass@1. See the source table for budgets.

Figure 14: **Published context for recursive reasoning.** Scores follow the cited source configurations; NRR's within-study scores appear in Table 1. U marks Maze-Unique; the open diamond marks FRM v1's transfer from Sudoku-Shah training to Sudoku-Extreme evaluation. B_{traj} denotes trajectories per transformed input; PTRM's Sudoku/Maze and ARC-2 configurations are separate rows. Dashes mean no value is listed for that configuration. ARC metrics, source splits, selection rules, and FRM's adaptive verification/restart budget are distinguished in Table 5; the plot is not a common-budget ranking.

The inference unit matters particularly on ARC. NRR's benchmark score aggregates 1000 transformed inputs into two candidates. URM reports pass@1, while HRM and TRM use two-attempt protocols. EqR reports both single-trajectory and restart-aggregated results; its published $D = 64$ denotes 64 outer iterations per trajectory, using the source's notation rather than our input symbol D. These distinctions identify complementary approaches to additional computation; they do not form a common-budget ranking.

Table 5: **Sources and protocol distinctions for Figure 14.** Values are transcribed from the cited versions. Dataset splits, augmentation, candidate selection and training exposure remain source-specific; the plot is not a common-budget or harmonized-split ranking.

Source	Configuration and interpretation
HRM / TRM / URM / GRAM	Source-reported values (Wang et al., 2025; Jolicoeur-Martineau, 2025; Gao et al., 2025; Baek et al., 2026). HRM/TRM use two-attempt ARC evaluation; URM reports pass@1. TRM's Sudoku value uses the MLP variant, while Maze/ARC use attention. GRAM uses its learned reward selector.
PTRM v1, Table 3	Sudoku/Maze use $K = 100$, with $D = 64/16$ and Q-based/modal-answer selection, respectively. ARC-2 uses $K = 25$ per augmented input and $D = 16$, with highest-Q selection followed by augmentation voting: 8.47% pass@1 and 9.72% pass@2. The plot displays pass@2 (Sghaier et al., 2026).
EqR v1, Table 4	$D = 64$; one or 128 trajectories. The Maze values 88.9%/93.0% are on Maze-Unique, marked U, rather than Maze-Hard. The broader configuration uses restart aggregation (Huang et al., 2026).
FRM v1, Table 1	96.1% on Sudoku-Extreme after training on Sudoku-Shah: an OOD result with up to 8192 verification/restart rounds and a 128-step inner loop. The separate 99.2% result in Table 2 is on Sudoku-Shah (Helbling et al., 2026b). FRM v3 reports 99.5% on Sudoku-Extreme under its revised protocol (Helbling et al., 2026a).
FPRM v1, Table 1	ARC-1/2: 47.5%/6.2%, pass@2. Adaptive fixed-point stopping and the source's ARC dataset/split definitions apply; equivalence to the other rows' evaluation protocol is not assumed (Movahedi et al., 2026).
SE-RRM v1, Table 3	ARC-1/2: 45.3%/7.1%, pass@2, with the source's default 16-step setting. These ARC values are not assigned to the 32-step Sudoku row. The source reports using the released Sudoku test set for validation (Freinschlag et al., 2026).