ENERGY-GUIDED RECURSIVE MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Recursive models show promise on reasoning and language tasks, yet their test-time scaling lacks a principled criterion for selecting trajectories or determining recurrent depth. We introduce Energy-guided Recursive Model (ERM), which uses Hopfield-type memories of valid local and global structures to assign intrinsic energies to candidate trajectories. These energies guide candidate selection and suggest an effective range of recurrent depths, implying that deeper recurrence does not necessarily improve reasoning accuracy. They also enable sampling methods such as parallel tempering to improve exploration. For reasoning tasks, ERM achieves optimal solutions on Sudoku (98.97%), Pencil Puzzle Bench (PPBench, 88.04%) and Maze (99.30%), reaching the best accuracy in recursive modeling. On language modeling, ERM reduces RedPajama-V2 perplexity by 1.74% with marginal inference overhead. The results support energy guidance as a practical framework for improving test-time scaling in recursive models.

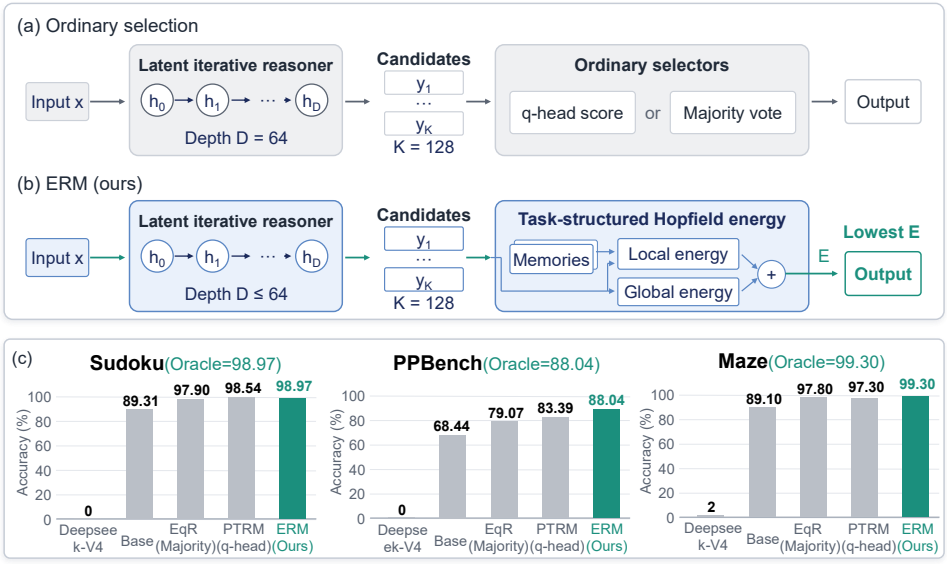


Figure 1: **Energy-guided recursive model improves on reasoning tasks.** (a) A reasoner generates K candidates at depth D; q-head scores or majority voting select the output. (b) ERM combines local Hopfield energies with a global constraint penalty and selects the lowest-energy candidate, without additional training. (c) Accuracy on three representative reasoning tasks. The ERM bars match their respective candidate-pool oracles.

1 INTRODUCTION

Test-time computation improves recursive models only when additional work produces better predictions, in both reasoning and language tasks. Autoregressive language models use longer chains of thought (Wei et al., 2022), multiple rationales (Wang et al., 2023), search (Yao et al., 2023), or learned verifiers (Cobbe et al., 2021; Lightman et al., 2024) to exploit additional inference budget (Snell et al., 2025). Recursive models generate candidate trajectories, but more trajectories or greater

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.

depth do not ensure better predictions. The shared problem is how to select useful candidates and decide when further recurrence helps.

Loop models reuse a shared operator within one forward computation, as in Neural GPUs (Kaiser & Sutskever, 2016), adaptive computation time (ACT) (Graves, 2016), Universal Transformers (Dehghani et al., 2019), and implicit or equilibrium networks (Geng et al., 2021; Bai et al., 2019). Recent transformers use recurrence for iterative algorithms, latent reasoning, and adaptive depth (Giannou et al., 2023; Yang et al., 2024; Saunshi et al., 2025; Geiping et al., 2025; Schöne et al., 2025; Bae et al., 2025; Zhu et al., 2025). HRM, TRM, EqR, and PTRM apply compact recurrent networks to structured reasoning (Wang et al., 2025; Jolicoeur-Martineau, 2025; Huang et al., 2026; Sghaier et al., 2026). Their trajectories can encounter spurious fixed points (Ren & Liu, 2026), making candidate selection consequential. Yet learned halting scores, confidence, and voting need not track prediction quality across recurrent steps.

Energy-based models assign scalar compatibility scores to configurations. Hopfield networks, Boltzmann machines, and energy-based learning define inference through low-energy states (Hopfield, 1982; Ackley et al., 1985; LeCun et al., 2006); dense associative memories and modern Hopfield networks connect retrieval to attention (Krotov & Hopfield, 2016; Ramsauer et al., 2021). Energy Transformer further designs representation updates that descend an attention-based energy (Hoover et al., 2023). We use Hopfield-type memories to define Hopfield energies over candidate predictions. Known rules supply these memories for reasoning, while observed continuations supply them for language modeling.

We introduce the **Energy-guided Recursive Model (ERM), a general framework for energy guidance in recursive reasoning and language modeling.** Hopfield energies guide selection, sampling, and recurrent depth without further neural training (Figure 1). Our main contributions are:

- **A general framework for energy guidance.** ERM uses Hopfield-type memories to define energies that improve recursive reasoning and language modeling at marginal overhead.
- **Oracle accuracy on reasoning tasks.** With $D \leq 64$ and $K = 128$, ERM achieves the candidate-pool oracles on Sudoku, PPBench, and Maze, suggesting optimal recurrent depth.
- **Improving looped language models.** ERM leverages Hopfield-type memories of observed continuations to reduce RedPajama-V2 perplexity by 1.74% without additional training.

2 Related Work

Latent iterative reasoning models reuse compact networks to solve structured tasks. HRM and TRM demonstrate this approach on Sudoku, Maze, and ARC-style problems (Wang et al., 2025; Jolicoeur-Martineau, 2025), while EqR and PTRM study attractor learning and stochastic exploration (Huang et al., 2026; Sghaier et al., 2026). Solve the Loop and fixed-point reasoners relate recurrence to convergence and adaptive stopping (Fein-Ashley & Rashidinejad, 2026; Movahedi et al., 2026). Lai et al. link slow reasoning to transient chaos near saddle points representing nearly correct solutions, with increasingly fractal basins on harder tasks (Lai et al., 2026). These studies explain the dynamics and limits of additional iterations. Adaptive computation time (ACT) learns a halting policy (Graves, 2016); ERM instead tests whether an explicit task energy can both select candidates and supply a stopping signal.

In language models, Chen et al. (2026) show that increasing model depth during pretraining and introducing boundary operators can improve compute-optimal scaling exponents. They study how architecture affects training efficiency, while ERM uses Hopfield energies to guide inference with fixed pretrained models.

Recent flow and diffusion models improve recurrent generation through denoising objectives. Flow Reasoning Models use self-conditioning and Fixed-Point Forcing to train on states produced by their own inference dynamics (Helbling et al., 2026). Drozdova et al. retain a hidden state across denoising steps and remove timestep conditioning, allowing a single trajectory to refine solutions without candidate selection (Drozdova et al., 2026). Looped flows train recurrent states with local denoising objectives and perform inference by probability-flow integration (Suleymanzade et al.,

2026). These methods improve the generator through training; ERM uses Hopfield energies to improve predictions from a fixed generator.

Energy-based models describe compatibility through a scalar energy. Classical Hopfield networks and Boltzmann machines associate low energy with preferred configurations (Hopfield, 1982; Ackley et al., 1985; LeCun et al., 2006), while modern Hopfield networks connect memory retrieval to attention (Ramsauer et al., 2021). Energy Transformer and IREM use energy minimization within representation updates or reasoning (Hoover et al., 2023; Du et al., 2022). Energy-Based Transformers (EBTs) learn an input–prediction energy and refine predictions through its gradients (Gladstone et al., 2026). Our language experiment keeps the EBT fixed and compares its learned scalar energy with Hopfield energies on the same candidates.

Verifier and reranking methods differ in the evidence used to judge predictions. Confidence and q-head selectors use model scores, while majority voting uses agreement among candidates (Sghaier et al., 2026). Learned verifiers can use additional outcome or process supervision (Cobbe et al., 2021; Lightman et al., 2024), and reinforcement learning can improve the generator itself through reward-based parameter updates (Guo et al., 2025). ERM requires neither a separately trained verifier nor post-training updates: its scores come from task memories or observed language continuations. This distinction motivates evaluating both the gain and the additional inference cost.

3 METHOD

3.1 PROBLEM SETUP

We study energy guidance for recursive models on reasoning and language tasks. Let x denote an input puzzle or text prefix, and let a recursive model with depth D produce K candidate outputs $\mathcal{Y}(x) = \{y_1, \ldots, y_K\}$ through independent rollouts, perturbations, or parallel-tempering chains. Each rollout provides a decoded prediction y_k over N positions and its predictive log-probabilities L_k. We write $E(y_k; x)$ as shorthand for a score that may also use L_k: Sudoku memories use these probabilities, while the other puzzle factors score decoded structures. Lower energy indicates greater task compatibility. For structured reasoning, the selected answer is

$$\hat{y} = y_{\hat{k}}, \qquad \hat{k} = \arg\min_{k \in \{1,\ldots,K\}} E(y_k; x). \tag{1}$$

This equation also defines the common baselines as energies: confidence uses negative mean token confidence, q-head uses the negative learned halting logit, and majority voting uses a negative agreement score. ERM constructs E from Hopfield-type memories and any global constraints. For language modeling, the same construction defines energies over tokens and combines candidate distributions, as described below. Appendix A.1 gives the independent-rollout selection algorithm; PT additionally changes candidate generation.

3.2 ENERGY-GUIDED RECURSIVE MODEL

ERM is derived from the Modern Hopfield view of energy-based memory retrieval. In a Modern Hopfield network, a query state q is compared with a memory set $\mathcal{M}$, and retrieval lowers an energy when q is similar to at least one stored memory (Ramsauer et al., 2021). A compact form of this retrieval energy is

$$E_{\mathrm{MHN}}(q; \mathcal{M}) = \frac{1}{2}\|q\|^2 - \tau \log \sum_{m \in \mathcal{M}} \exp\left(\frac{\mathrm{sim}(q, m)}{\tau}\right), \tag{2}$$

where $\mathrm{sim}(q, m)$ is the similarity between the query and memory, τ is the retrieval temperature, and the squared-norm term regularizes the continuous query state. ERM uses this log-sum-exp construction to define Hopfield energies over candidate predictions: each factor j has an input-conditioned memory set $\mathcal{M}_j(x)$, the similarity is replaced by the negative distance $-\beta d_j(y_k, m)$, and constraints that are global rather than locally enumerable are written as $G(y_k, x)$:

$$E_{\mathrm{ERM}}(y_k; x) = \mu G(y_k, x) - \sum_{j=1}^{J} \tau \log \sum_{m \in \mathcal{M}_j(x)} \exp\left(-\frac{\beta d_j(y_k, m)}{\tau}\right) \tag{3}$$

Here $d_j(y_k, m)$ measures mismatch to memory m, β is the distance scale, and μ weights the global term. As τ approaches zero, each retrieval term approaches the minimum scaled distance to its local memory set. The factor sum therefore scores compatibility with enumerable local structures. The global penalty $G(y_k, x)$ accounts for additional constraints such as connectivity; it is omitted when no such constraints are specified. Reasoning uses memories of valid patterns, while language uses distributions over observed continuations.

3.3 Parallel Tempering over Candidate Energies

As an optional sampling extension, parallel tempering (PT) uses the same energy to define

$$\pi_T(y \mid x) \propto \exp\left(-\frac{E(y; x)}{T}\right), \tag{4}$$

where $T > 0$ controls exploration. Replicas at $T_1 < \cdots < T_R$ exchange states between neighboring temperatures, with acceptance probability

$$a = \min\left\{1, \exp\left[\left(\frac{1}{T_\rho} - \frac{1}{T_\eta}\right)(E_\rho - E_\eta)\right]\right\}. \tag{5}$$

Here E_ρ and E_η are the energies at temperatures T_ρ and T_η. We evaluate PT through selected and oracle accuracy, without assuming that the recurrent dynamics exactly sample π_T.

3.4 Reasoning Tasks

The three reasoning tasks use Equation 3, with task-specific memories, distances, and global constraints. In the shared-pool experiments, each selector uses the same generated candidates; in PT, the energy also guides replica exchanges during generation.

Sudoku instantiates $E_{\mathrm{ERM|Sudoku}}$ with row, column, and box memories. Each factor j is one row, column, or 3×3 box, and each memory $m \in \mathcal{M}_j(x)$ assigns one valid digit to each of the nine cells in that factor. Let $L_k[i, d]$ be rollout k's predictive log-probability for digit d at cell i, and let m_i be the digit in memory m. The Sudoku energy is the direct specialization

$$E_{\mathrm{ERM|Sudoku}}(y_k, L_k; x) = -\sum_j \tau \log \sum_{m \in \mathcal{M}_j(x)} \exp\left(\frac{1}{\tau}\sum_{i \in j} L_k[i, m_i]\right), \tag{6}$$

The sum over j covers rows, columns, and boxes. The memory distance is the negative log-probability assigned to a valid unit pattern, so this score uses soft predictions rather than only the decoded board. A log-permanent dynamic program evaluates the inner sum exactly; clue cells are clamped to their input digits.

PPBench instantiates $E_{\mathrm{ERM|PPBench}}$ with puzzle-type-specific local memories plus global distance potentials. For enumerable rules, each factor j stores legal binary or multi-class local patterns in $\mathcal{M}_j(x)$, and $d_j(y_k, m)$ is a normalized Hamming-style distance between the candidate pattern and a legal memory. Lightup uses numbered-clue exact-count memories and line-of-sight at-most-one-bulb memories; Tapa uses cyclic run-pattern memories around clues; Heyawake uses room-count and adjacency memories; Nurikabe uses no-2×2-sea memories and island/sea potentials; and the Sudoku subset reuses the Sudoku permutation memories. Connectivity and coverage rules are not faithfully enumerable as small local memories, so they enter through $G(y_k, x)$:

$$E_{\mathrm{ERM|PPBench}}(y_k; x) = 64\, G(y_k, x) - \sum_j \tau \log \sum_{m \in \mathcal{M}_j(x)} \exp\left(-\frac{128\, d_j(y_k, m)}{\tau}\right). \tag{7}$$

Here $\beta = 128$ and $\mu = 64$; no separate confidence score is added.

Maze instantiates $E_{\mathrm{ERM|Maze}}$ with local path memories and a global rule penalty. The local factors use the same $\mathcal{M}_j(x)$ notation as the other tasks: terminal factors store patterns in which the start or goal has path degree one, ordinary open-cell factors store either off-path patterns or on-path degree-two patterns, and wall factors enforce consistency with the input maze. Disconnected cycles and

fragments can satisfy local degree constraints, so $G(y_k, x)$ additionally penalizes violations of the global path rules. The resulting energy is

$$E_{\mathrm{ERM|Maze}}(y_k; x) = G(y_k, x) - 0.25 \sum_j \tau \log \sum_{m \in \mathcal{M}_j(x)} \exp\left(-\frac{d_j(y_k, m)}{\tau}\right). \tag{8}$$

The global rule penalty $G(y_k, x)$ checks format consistency with the input maze, start-goal connectivity, path degrees, cycles, and path length. The local retrieval weight is 0.25, with unit distance and global-penalty scales.

3.5 Language Modeling and Task Adaptation

The same framework applies to both reasoning and language by specifying the prediction variables, Hopfield-type memories, and any global constraints. Reasoning uses valid local patterns and global rules. Language has many acceptable continuations, so its Hopfield-type memories store token distributions rather than discrete valid assignments. Hopfield energies then define a normalized prediction distribution.

Language prediction. We apply the same factor construction in Equation 3 to token distributions. At position t, let $p_k(v)$ be the conditional probability of token v from trajectory k, and let $q_t(v)$ be a continuation distribution retrieved using the observed prefix $x_{<t}$. Define

$$p_0(v) = \frac{1}{K} \sum_{k=1}^{K} p_k(v), \qquad r(v) = (1 - g_t) p_0(v) + g_t q_t(v), \tag{9}$$

where $g_t \in [0, 1]$ controls the memory contribution. The two factors use $\mathcal{M}_1 = \{p_1, \ldots, p_K\}$ with uniform weights $1/K$, and $\mathcal{M}_2 = \{p_0, q_t\}$ with weights $(1 - g_t, g_t)$. For a memory distribution p_m, the mismatch is $d_j(v, m) = -\log p_m(v)$. We average the two weighted retrieval energies at unit temperature:

$$E_{\mathrm{LM}}(v) = -\frac{1}{2} \sum_{j=1}^{2} \log \sum_{m \in \mathcal{M}_j} w_{j,m} \exp[-d_j(v, m)] + \log Z_t, \tag{10}$$

where $Z_t = \sum_{u \in \mathcal{V}} \sqrt{p_0(u) r(u)}$ normalizes $\exp[-E_{\mathrm{LM}}(v)]$ over the vocabulary. The resulting token distribution is the normalized geometric mean of p_0 and r: the first factor retains model evidence, and the second combines it with retrieved continuations. At $g_t = 0$, both factors reduce to p_0. The language formulation uses weighted Hopfield-type memories at $\tau = \beta = 1$, averages two factors, and sets $\mu = 0$.

4 Experiments

4.1 Experimental Setup

We evaluate ERM on reasoning and language tasks with fixed pretrained models. Reasoning experiments use EqR with $D \leq 64$ and $K = 128$; the shared-pool comparison fixes $D = 64$, while depth and stopping studies use smaller budgets. Language experiments use an EBT with $K = 8$ (Section 4.4). The reasoning suite contains Sudoku-Lite, the five-type PPBench validation split derived from Pencil Puzzle Bench (Waugh, 2026), and the official EqR Maze-Unique setting. The Maze task uses the same dataset used by EqR: a uniquely solvable Maze-hard-1k variant synthesized from perfect 30×30 mazes by sampling start-goal pairs, filtering path lengths, and de-duplicating train/test splits (Huang et al., 2026). Reasoning comparisons report exact accuracy, requiring the full answer to match the target; appendix diagnostics also report per-position token accuracy. Appendix Table 13 summarizes the parameter and compute accounting: ERM adds no learned neural parameters and only adds task-energy scoring on top of the shared rollout budget. Language modeling uses token-weighted perplexity to assess the full prediction distribution.

For reasoning, we compare ERM with four non-oracle selectors. The one-rollout baseline uses a single EqR prediction. Whole-sequence majority chooses the most frequent complete answer among the K candidates when applicable, while cell-majority voting is additionally reported for PPBench

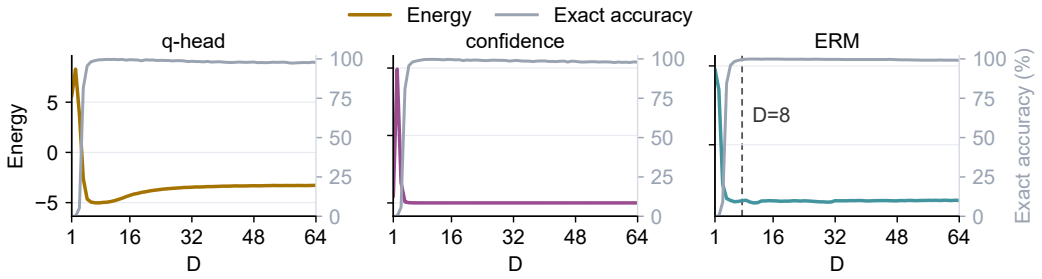


Figure 2: **Maze selector dynamics over recurrent depth.** Solid curves show mean selected energy; gray curves show exact accuracy (right axes) for $D = 1$–64 and $K = 128$ Maze rollouts. Confidence uses negative mean valid-token maximum log-probability; q-head and confidence scores quickly saturate. ERM uses the Maze energy in Equation 3. The vertical dashed line marks $D = 8$, where ERM reaches 99.60% exact accuracy.

Table 1: **Shared-candidate selection at $D = 64$, $K = 128$.** All methods select from the same candidate pool for each task. Base denotes the one-rollout baseline, Conf. denotes confidence selection, the oracle is the best possible exact accuracy inside that pool, and the ERM gap is oracle minus ERM.

Task	Base	Majority	q-head	Conf.	ERM	Oracle	Gap
Sudoku	89.31	97.90	98.54	98.58	98.83	98.83	0.00
PPBench	68.44	79.07	83.39	80.73	88.04	88.04	0.00
Maze	89.10	97.80	97.30	97.80	99.30	99.30	0.00

and Maze diagnostics. The q-head selector chooses the candidate with the largest learned EqR halting score, and confidence chooses the candidate with the largest mean valid-token confidence. The candidate oracle uses the target to report whether any of the K candidates is exactly correct, defining the selection upper bound for that pool. Because Maze exposes a particularly sharp distinction between local confidence and global path compatibility, Figure 2 also shows how the selector energy and exact accuracy evolve over recurrent depth.

Section 4.5 tests whether the energy dynamics in Figure 2 can guide early stopping.

4.2 Oracle Accuracy on Reasoning Tasks

On Sudoku, ERM selects all $2024/2048$ correct answers available in the shared pool (98.83%), five more than confidence selection. The remaining errors arise from candidate generation; PT adds three further solves (Table 2).

PPBench shows the largest selection gain: ERM solves $265/301$ puzzles, matching the oracle and exceeding q-head by 14 solves. Its local memories and global penalties distinguish valid candidates across heterogeneous puzzle rules.

On Maze, ERM reaches the 99.30% oracle, compared with 97.80% for majority and confidence and 97.30% for q-head. Thus ERM recovers every correct answer available in this pool. At $D = 16$, both ERM and the oracle reach 99.90% (Table 4); deeper recurrence therefore does not guarantee better candidate coverage.

4.3 Sampling Extension: Parallel Tempering

PT-ERM raises Sudoku accuracy from 98.83% to 98.97% ($2027/2048$), adding three solves, and preserves the PPBench and Maze results of 88.04% and 99.30% (Figure 3; Table 2). Selected accuracy matches each PT pool's oracle. The main gains come from selection; the three-solve PT improvement is modest, and its robustness across random seeds remains untested.

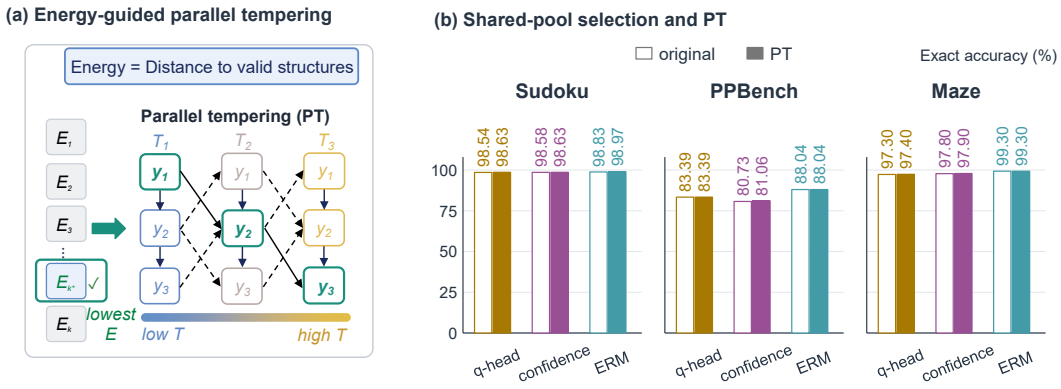


Figure 3: **Energy-guided parallel tempering and selection.** (a) The task energy guides exchanges between neighboring temperatures and selects the lowest-energy candidate. Vertical arrows show refinement; diagonal arrows show exchange proposals accepted by Equation 5. Teal states illustrate a replica moving across temperatures. (b) Hollow bars show shared-pool selection; solid bars show PT with the same scoring energy. All values are exact accuracy (%).

Table 2: **Parallel tempering with different energies.** Original is the accuracy of the same selector on the shared EqR candidate pool, selected accuracy is the final answer chosen after PT sampling, and oracle accuracy is pass@K within the PT-generated candidates.

Task	PT energy	Original	Selected	Oracle	Swap rate	Count
Sudoku	q-head	98.54	98.63	98.68	0.7964	2020/2048
Sudoku	Confidence	98.58	98.63	98.63	0.9548	2020/2048
Sudoku	ERM	98.83	**98.97**	98.97	0.9010	2027/2048
PPBench	q-head	83.39	83.39	87.38	0.8948	251/301
PPBench	Confidence	80.73	81.06	88.04	0.9939	244/301
PPBench	ERM	88.04	**88.04**	88.04	0.9378	265/301
Maze	q-head	97.30	97.40	99.20	0.8669	974/1000
Maze	Confidence	97.80	97.90	99.30	0.9952	979/1000
Maze	ERM	99.30	**99.30**	99.30	0.9471	993/1000

4.4 Improved Language Modeling

Language modeling tests the same energy-guidance principle using Hopfield-type memories of observed continuations. Figure 4(a) shows its integration with a pretrained EBT. We use an S2 EBT with six blocks and 44.86M parameters, trained for 105,000 steps, and evaluate $K = 8$ shared trajectories on a frozen RedPajama-V2 proxy of 33,000 documents and 7,397,883 target tokens (Gladstone et al., 2026; Weber et al., 2024). The continuation memory combines training-source suffix counts with earlier observations in the current document. We report token-weighted perplexity, $\exp(\sum_d \mathrm{NLL}_d / \sum_d n_d)$, where n_d counts scored tokens; Appendix B.2 gives the evaluation settings.

Table 3: **Language modeling with a fixed EBT.** All methods use the same $K = 8$ candidates on the 33k proxy. Lower perplexity is better. ERM uses Equation 10; the learned scalar energy selects one trajectory.

Method	Perplexity	Reduction (%)
First candidate	44.189	—
Candidate average	44.147	0.094
EBT scalar-energy selection	44.180	0.020
ERM Hopfield energies	**43.419**	**1.742**

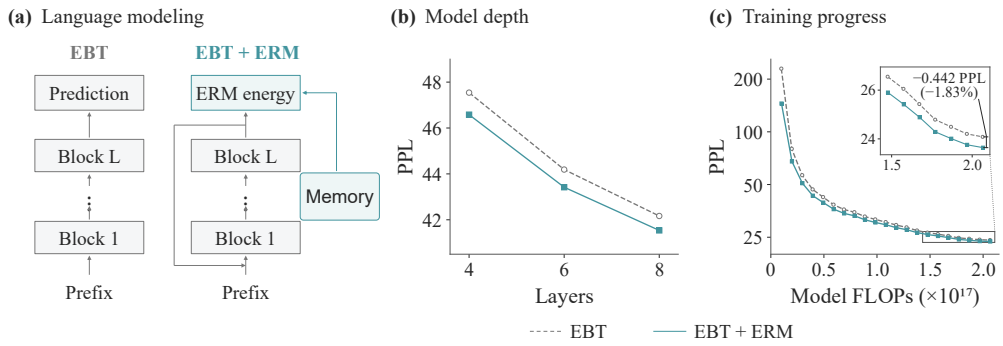


Figure 4: **ERM for language modeling.** (a) EBT backbone and ERM inference with continuation memory. (b) Perplexity (PPL) of 4-, 6-, and 8-block models on the 33k-document proxy after 105,000 training steps. (c) Six-block PPL across 21 checkpoints on a fixed 256-document slice, against pretraining plus candidate-generation FLOPs (one EBT candidate; eight ERM candidates). The main PPL axis is logarithmic; the linear-scale inset shows steps 75,000–105,000 and the final-checkpoint PPL reduction. EBT denotes the first candidate; EBT + ERM uses Equation 10 with $K = 8$.

ERM reduces perplexity from 44.189 to 43.419 (Table 3), a 1.742% improvement (paired-document bootstrap 95% CI: 1.711–1.773%), and improves on candidate averaging by 1.649%. Selecting the trajectory with the lowest summed EBT scalar energy reduces perplexity by 0.020%. Hopfield energies also lower PPL from 47.541 to 46.581 for four blocks and from 42.166 to 41.538 for eight blocks (Figure 4(b)). These comparisons show consistent gains across the three backbone depths with fixed pretrained parameters.

Across 21 checkpoints (steps 5,000–105,000), we hold the memory, hyperparameters, and 256-document slice fixed. Mean next-token energy decreases at every interval, from 4.977 to 3.163 nats; the corresponding PPL falls from 145.033 to 23.641 (Figure 4(c)).

ERM adds no optimizer steps or trainable parameters and uses the same eight trajectories as candidate averaging. Its additional cost is memory construction, lookup, and normalization. The full 33k comparison takes 2.83 hours on one RTX 4090; Appendix B.2 gives the timing scope and development cost.

4.5 Energy-Based Early Stopping

The energy dynamics in Figure 2 suggest that task compatibility can guide recurrent depth as well as final selection. On Maze, we monitor the batch mean of the lowest candidate energy for each input and stop when its relative change stays below 0.01 for two consecutive checks, starting at step four. With $K = 128$ PT replicas and a maximum depth of 64, all 63 batches stop by this criterion at depths 6–11, averaging 7.60 steps, with $996/1000 = 99.60\%$ exact accuracy (Table 4). Appendix B.8 gives the precise criterion.

Task energy supplies a batch-level stopping signal without additional ACT training (Graves, 2016). The runs use different pools, and reduced depth does not by itself establish lower wall-clock cost.

Table 4: **Maze accuracy with reduced recurrent depth.** All runs use $K = 128$ and the same 1000-example test split. The early-stopping depth is averaged over 63 batches; the adaptive run uses PT and is not a shared-pool comparison with the fixed-depth runs.

Inference	Max. depth	Mean stop	Exact (%)	Count
ERM, fixed depth	64	—	99.30	993/1000
ERM, fixed depth	16	—	99.90	999/1000
ERM + PT, energy stopping	64	7.60	99.60	996/1000

5 DISCUSSION

ERM uses one principle across reasoning and language: Hopfield-type memories define Hopfield energies that assess candidate predictions. Reasoning uses valid patterns and global constraints; language uses observed continuations. This framework recovers candidate-pool oracle accuracy on reasoning tasks and improves language perplexity without further training. On Maze, the energies also guide stopping, reaching 99.60% accuracy at a mean depth of 7.6. More recurrence is therefore not always better.

Recurrent computation is also being explored in billion-parameter language models, including recurrent-depth transformers and Ouro's looped models (Geiping et al., 2025; Zhu et al., 2025). Hopfield energies could help combine their predictions or guide stopping when energy tracks prediction quality. Our fixed-EBT experiment demonstrates the language formulation; transfer to larger recurrent language models remains untested.

5.1 LIMITATIONS

The framework is general, but its Hopfield-type memories and constraints depend on the task. Our reasoning comparisons do not isolate the local terms of Hopfield energies from a discrete rule checker or global penalties. Tasks with unknown rules require a way to infer suitable memories and constraints. Language modeling is evaluated on one corpus proxy.

Our language study isolates inference gains on a fixed iterative EBT, rather than comparing architectures with the nonrecurrent Transformer++ baseline (Gladstone et al., 2026). It therefore does not establish an advantage over Transformer++; that comparison requires matched training data and compute. Finally, stable energy does not certify correctness, and the stopping experiment does not directly compare against a trained ACT controller.

6 CONCLUSION

We presented ERM, a framework for energy guidance in recursive reasoning and language models. Hopfield-type memories define Hopfield energies for candidate selection, prediction, sampling, and depth control. ERM matches candidate-pool oracle accuracy on Sudoku, PPBench, and Maze, and reduces language-model perplexity by 1.74% without further training. Maze stopping and the modest Sudoku gain from PT show how the same energies can guide computation. The results support energy guidance as a practical principle for test-time scaling, with gains depending on candidate quality and memory design.

AI USE STATEMENT

Generative AI tools were used only to assist with writing and information search. They were not used for method development, experiment design, implementation, data processing, or result interpretation. The authors are responsible for verifying AI-assisted text and retrieved information and for the final content of this work.

ETHICS STATEMENT

This work evaluates ERM on structured puzzle benchmarks and a RedPajama-V2 language-modeling proxy, without human-subject experiments. Web-derived text may contain biased, harmful, or sensitive content; our evaluation does not establish privacy, fairness, or deployment-safety guarantees.

REPRODUCIBILITY STATEMENT

Section 3 defines the energies and inference procedure, and Section 4.1 describes the evaluation setup. Appendix A.1 provides inference pseudocode; Appendix B.2 details the language protocol and memory construction; and Appendix B.8 specifies the stopping criterion. Appendix C reports parameter and compute accounting.

REFERENCES

David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. *Cogn. Sci.*, 9(1):147–169, 1985. URL https://doi.org/10.1016/S0364-0213(85)80012-4.

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. In *Adv. Neural Inf. Process. Syst.*, 2025. URL https://openreview.net/forum?id=QuqsEIVWIG.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Adv. Neural Inf. Process. Syst.*, volume 32, 2019. URL https://papers.nips.cc/paper/8358-deep-equilibrium-models.

Zixi Chen, Akshay Vegesna, Samip Dahal, and Andrew Gordon Wilson. How model growth, recursion, and boundary operators influence scaling exponents. *arXiv preprint arXiv:2609.19107*, 2026. doi: 10.48550/arXiv.2609.19107. URL https://arxiv.org/abs/2609.19107.

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

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *Proc. Int. Conf. Learn. Represent.*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

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. *arXiv preprint arXiv:2609.01449*, 2026. URL https://arxiv.org/abs/2609.01449.

Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Learning iterative reasoning through energy minimization. In *Proc. Int. Conf. Mach. Learn.*, volume 162, pp. 5570–5582, 2022. URL https://proceedings.mlr.press/v162/du22d.html. Code: https://github.com/yilundu/irem_code_release.

Jacob Fein-Ashley and Paria Rashidinejad. Solve the loop: Attractor models for language and reasoning. *arXiv preprint arXiv:2605.12466*, 2026. doi: 10.48550/arXiv.2605.12466. URL https://doi.org/10.48550/ARXIV.2605.12466.

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. In *Adv. Neural Inf. Process. Syst.*, 2025. URL https://openreview.net/forum?id=S3GhJooWIC.

Zhengyang Geng, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. On training implicit models. In *Adv. Neural Inf. Process. Syst.*, volume 34, pp. 24247–24260, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/cb8da6767461f2812ae4290eac7cbc42-Abstract.html.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *Proc. Int. Conf. Mach. Learn.*, 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

Alexi Gladstone, Ganesh Nanduru, Md Mofijul Islam, Peixuan Han, Hyeonjeong Ha, Aman Chadha, Yilun Du, Heng Ji, Jundong Li, and Tariq Iqbal. Energy-based transformers are scalable learners and thinkers. In *Proc. Int. Conf. Learn. Represent.*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/e19a65fd53b6f9a88b354da98813465d-Abstract-Conference.html.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016. URL https://arxiv.org/abs/1603.08983.

Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

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:2606.29150*, 2026. URL https://arxiv.org/abs/2606.29150.

Benjamin Hoover, Yuchen Liang, Bao Pham, Rameswar Panda, Hendrik Strobelt, Duen Horng Chau, Mohammed J. Zaki, and Dmitry Krotov. Energy transformer. In *Adv. Neural Inf. Process. Syst.*, volume 36, pp. 27532–27559, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/57a9b97477b67936298489e3c1417b0a-Abstract-Conference.html.

John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. U.S.A.*, 79(8):2554–2558, 1982. URL https://doi.org/10.1073/pnas.79.8.2554.

Benhao Huang, Zhengyang Geng, and Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning. In *Proc. Int. Conf. Mach. Learn.*, 2026. URL https://openreview.net/forum?id=lh95PnOlpM.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025. URL https://arxiv.org/abs/2510.04871.

Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. In *Proc. Int. Conf. Learn. Represent.*, 2016. URL https://arxiv.org/abs/1511.08228.

Dmitry Krotov and John J. Hopfield. Dense associative memory for pattern recognition. In *Adv. Neural Inf. Process. Syst.*, volume 29, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/eaae339c4d89fc102edd9dbdb6a28915-Abstract.html.

Jeffrey Lai, Anthony Bao, John Quinn, and William Gilpin. Fractal basins trap latent reasoning. *arXiv preprint arXiv:2609.04963*, 2026. URL https://arxiv.org/abs/2609.04963.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. In *Predicting Structured Data*, pp. 191–246. MIT Press, 2006. URL http://yann.lecun.com/exdb/publis/pdf/lecun-06.pdf.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *Proc. Int. Conf. Learn. Represent.*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

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. *arXiv preprint arXiv:2606.18206*, 2026. doi: 10.48550/arXiv.2606.18206. URL https://arxiv.org/abs/2606.18206.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, David Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need. In *Proc. Int. Conf. Learn. Represent.*, 2021. URL https://openreview.net/forum?id=tL89RnzIiCd.

Zirui Ren and Ziming Liu. Are your reasoning models reasoning or guessing? a mechanistic analysis of hierarchical reasoning models. *arXiv preprint arXiv:2601.10679*, 2026. doi: 10.48550/arXiv.2601.10679. URL https://arxiv.org/abs/2601.10679.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *Proc. Int. Conf. Learn. Represent.*, 2025. URL https://openreview.net/forum?id=din0lGfZFd.

Mark Schöne, Babak Rahmani, Heiner Kremer, Fabian Falck, Hitesh Ballani, and Jannes Gladrow. Implicit language models are rnns: Balancing parallelization and expressivity. In *Proc. Int. Conf. Mach. Learn.*, 2025. URL https://openreview.net/forum?id=5EbiopWH6e.

Amin Sghaier, Ali Parviz, and Alexia Jolicoeur-Martineau. Probabilistic tiny recursive model. *arXiv preprint arXiv:2605.19943*, 2026. URL https://arxiv.org/abs/2605.19943.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. In *Proc. Int. Conf. Learn. Represent.*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Ayhan Suleymanzade, Chanhyuk Lee, Floor Eijkelboom, Nicholas M. Boffi, İsmail İlkan Ceylan, and Jinwoo Kim. Thinking with looped flows. *arXiv preprint arXiv:2609.11801*, 2026. URL https://arxiv.org/abs/2609.11801.

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. URL https://arxiv.org/abs/2506.21734.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *Proc. Int. Conf. Learn. Represent.*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Justin Waugh. Pencil puzzle bench: A benchmark for multi-step verifiable reasoning. *arXiv preprint arXiv:2603.02119*, 2026. URL https://arxiv.org/abs/2603.02119.

Maurice Weber, Daniel Y. Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Ré, Irina Rish, and Ce Zhang. RedPajama: an open dataset for training large language models. In *Adv. Neural Inf. Process. Syst.*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d34497330b1fd6530f7afd86d0df9f76-Abstract-Datasets_and_Benchmarks_Track.html.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Adv. Neural Inf. Process. Syst.*, volume 35, pp. 24824–24837, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *Proc. Int. Conf. Learn. Represent.*, 2024. URL https://openreview.net/forum?id=HHbRxoDTxE.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Adv. Neural Inf. Process. Syst.*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract-Conference.html.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025. URL https://arxiv.org/abs/2510.25741.

A NOTATION TABLE

Table 5: **Notation used in the main text.**

Symbol	Meaning
x	Input puzzle or observed text prefix.
q	Query state in the Modern Hopfield retrieval energy.
$\|q\|^2$	Squared Euclidean norm of the Modern Hopfield query state.
D	Recurrent depth; reasoning experiments use $D \leq 64$.
K	Number of candidate rollouts or sampled trajectories.
$\mathcal{Y}(x)$	Candidate pool produced for input x.
y	Generic candidate output variable used when defining a distribution.
y_k	The k-th candidate structured output.
k	Candidate index inside the pool $\mathcal{Y}(x)$.
$\hat{k}$, $\hat{y}$	Selected candidate index and selected structured answer.
N	Number of output positions or cells.
$E(y_k; x)$	Candidate energy; shorthand includes L_k when required. Lower is better.
E_ρ, E_η	Energies of two PT replicas proposed for swapping.
E_{MHN}	Modern Hopfield retrieval energy in Equation 2.
E_{ERM}	ERM scoring function for Hopfield energies in Equation 3.
$E_{\mathrm{ERM\|Sudoku}}$, $E_{\mathrm{ERM\|PPBench}}$, $E_{\mathrm{ERM\|Maze}}$	Task-specific instantiations of E_{ERM}.
J	Number of task factors used by ERM.
j	Factor index, such as a Sudoku row or a PPBench clue neighborhood.
$\mathcal{M}$	Generic memory set used in the Modern Hopfield retrieval energy.
$\mathcal{M}_j(x)$	Input-conditioned memory set for factor j.
m	One stored memory pattern in either $\mathcal{M}$ or $\mathcal{M}_j(x)$.
$d_j(y_k, m)$	Factor mismatch, using L_k for Sudoku; for language, $d_j(v, m) = -\log p_m(v)$.
τ	Soft-retrieval temperature for the memory log-sum-exp.
$G(y_k, x)$	Global rule-violation score for constraints not enumerated locally.
μ	Weight of the global rule-violation term.
T	Sampling temperature in the tempered distribution.
$\pi_T(y \mid x)$	Tempered distribution over candidate outputs at temperature T.
$T_1, \ldots, T_R$	Parallel-tempering replica temperatures.
T_ρ, T_η	Temperatures of two PT replicas proposed for swapping.
R	Number of PT replicas.
ρ, η	Replica indices used in the PT swap rule.
a	Swap acceptance probability between two PT replicas.
i, d	Output position and digit/token index.
L_k, $L_k[i, d]$	Predictive log-probability tensor from rollout k and its entry at position i, digit/token d.
m_i	Digit assigned by Sudoku memory m to cell i.
sim	Generic similarity function used in Modern Hopfield retrieval.
β	Scale parameter for memory similarity or distance.
t, $x_{<t}$	Token position and observed prefix in language prediction.
v, u, $\mathcal{V}$	Candidate token, summation token, and vocabulary.

Symbol	Meaning
$p_k(v)$, $p_0(v)$	Candidate token distribution and uniform candidate average.
$q_t(v)$, g_t, $r(v)$	Continuation memory, memory gate, and interpolated distribution.
$w_{j,m}$, $p_m(v)$	Memory weight and token distribution for memory m in factor j.
E_{LM}, Z_t	Language energy and its exact normalization constant.
n_d, NLL_d	Scored token count and summed negative log-likelihood for document d.

A.1 ERM Inference Pseudocode

The procedure below describes independent rollouts followed by ERM selection; it does not include the optional PT exchanges. EqR supplies the stochastic depth–breadth candidate pool, following the randomized-initialization and repeated-update structure in Algorithm 1 of Huang et al. (2026). ERM leaves these neural updates unchanged and selects candidates using Hopfield energies from Equation 3.

Algorithm 1: ERM selection from independent rollouts

Input: input x; EqR update f_θ; depth D; candidates K;
task memories $\{\mathcal{M}_j(x)\}_{j=1}^{J}$; global score G;
task scoring parameters, including τ, β, and μ.
Output: selected structured answer $\hat{y}$.
for $k = 1, \ldots, K$ **do**
  initialize the EqR carry for candidate k independently.
  for $t = 1, \ldots, D$ **do**
    update the carry with f_θ conditioned on x.
  end for
  decode y_k and retain predictive log-probabilities L_k.
end for
for $k = 1, \ldots, K$ **do**
  score y_k using the task memories, global score, and parameters;
  for Sudoku, use L_k in Equation 6.
end for
$\hat{k} \leftarrow \arg\min_{k \in \{1,\ldots,K\}} E_{\mathrm{ERM}}(y_k; x)$; $\hat{y} \leftarrow y_{\hat{k}}$.
return $\hat{y}$.

Here x is the input instance, f_θ is the trained EqR recurrent update, D is the number of recurrent steps, and K is the number of independent rollouts. The indices k and t denote the candidate and recurrent step, y_k is the final candidate from rollout k, $\mathcal{M}_j(x)$ is the valid memory set for factor j among J task factors, $G(y_k, x)$ measures global rule violations, E_{ERM} is the ERM energy, and $\hat{k}, \hat{y}$ are the selected index and answer.

B Additional Experimental Details

B.1 EqR+ET Sudoku Motivation

The EqR+ET Sudoku experiment compares selectors on a checkpoint trained with an Energy Transformer architecture. ET energy and confidence both solve $1574/2048 = 76.86\%$ of puzzles, compared with $1351/2048 = 65.97\%$ for one rollout. PT-confidence solves $1555/2048 = 75.93\%$. These results motivate evaluating the scoring rule and sampling procedure separately. The main experiments use task memories with fixed pretrained generators and report selected and oracle accuracy for each candidate pool.

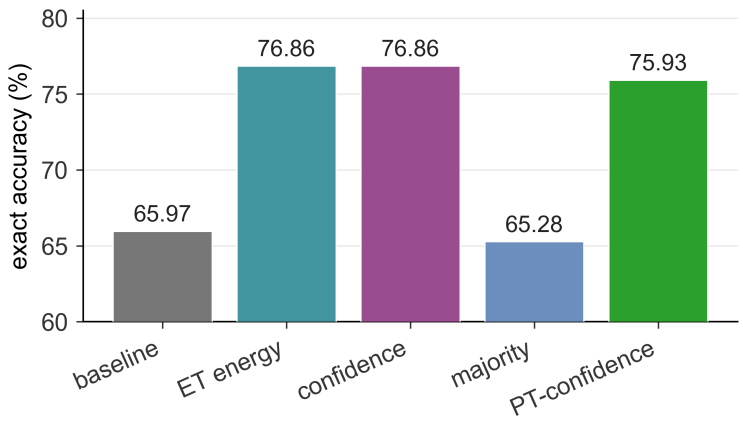


Figure 5: **Training with an Energy Transformer architecture.** This appendix experiment uses the EqR+ET Sudoku checkpoint with $D = 64$, $K = 16$, and PT temperatures from 0.001 to 1.0. ET energy and confidence both improve over the one-rollout baseline, motivating the main inference-time ERM design.

Table 6: **EqR+ET Sudoku appendix result.** Exact and token accuracy at $D = 64$, $K = 16$; PT-confidence uses confidence for sampling and final selection.

Method	Exact	Token	Count
baseline	65.97	87.79	1351/2048
ET energy	76.86	91.20	1574/2048
confidence	76.86	91.25	1574/2048
majority	65.28	88.13	1337/2048
PT-confidence	75.93	90.99	1555/2048

B.2 Language Protocol and Cost

The S2 EBT has six independently parameterized blocks, width 384, six attention heads, and a 256-token context. It was pretrained with batch size 32 for 105,000 optimizer steps; the recorded duration is 62,960 seconds on an RTX PRO 6000. The language comparison uses the final checkpoint of this training run and a frozen proxy drawn from shards separate from training; the same proxy supports development and evaluation. Inference uses $K = 8$, noise 0.5, seed 33000, and batch size four. Confidence intervals quantify document-sampling uncertainty within this proxy using 3,000 paired bootstrap resamples of token-weighted NLL.

Continuation tables use 24,000 documents from 13 training sources, context lengths one to four, and up to 32 continuations per context. Document-local suffixes of lengths one to eight use only previously observed tokens, and the resulting memory keeps at most 192 continuations per position. The retained memory mass defines a gate capped at 0.95 and scaled by 0.6, selected on 256 calibration documents in a 1,280-document development run. The sparse correction includes an exact normalization over the vocabulary; no current or future target enters the energy. Its computation adds no gradient update or learned verifier.

The depth comparison evaluates 4-, 6-, and 8-block EBTs, each trained for 105,000 steps, on the full 33k proxy. The checkpoint sweep uses 256 documents (57,628 tokens), $K = 8$, and fixed continuation tables and gate. In Figure 4(c), pretraining costs $C(s) = s \times 32 \times 256 \times 4.5 \times (6 \times 2{,}457{,}600 + 38{,}597{,}376)$ FLOPs at optimizer step s, using batch size 32, context length 256, and 4.5 forward-equivalent training passes. The two inner terms count block and vocabulary-projection FLOPs. Two refinement steps plus a final forward pass cost seven forward equivalents per candidate on the doubled sequence, giving $I = 57{,}628 \times 7 \times (2 \times 6 \times 2{,}457{,}600 + 38{,}597{,}376) = 2.747 \times 10^{13}$ FLOPs. We plot $C(s) + I$ for EBT and $C(s) + 8I$ for ERM; first-candidate PPL comes from the shared $K = 8$ evaluation, with its one-candidate cost assigned analytically. Model FLOPs exclude

memory construction and lookup. Target tokens enter only the evaluation metric. The recorded development evaluation takes 390 seconds (6.5 minutes). This duration covers candidate generation and all comparison branches together. Additional language ablations and gain distributions are reported in Figures 6 and 8.

B.3 Language Ablations and Gains

Figure 6 tests the two factors in Equation 10. Using only the model–memory mixture $r = (1 - g_t)p_0 + g_t q_t$ gives 46.215 PPL, versus 43.419 for normalized $\sqrt{p_0 r}$, supporting the separate model-distribution factor. Uniform and energy-softmax candidate weights differ by 0.0048 PPL; we use uniform weights for simplicity.

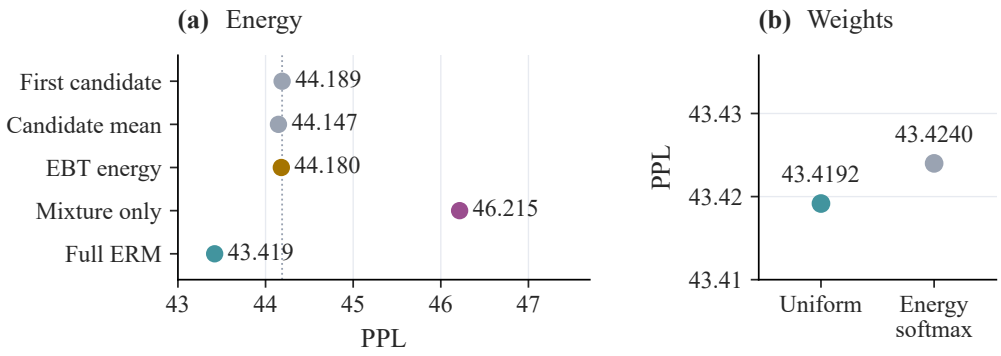


Figure 6: **Language energy ablation.** Six-block EBT with $K = 8$ on the 33,000-document proxy. (a) Token-weighted PPL; lower is better. The dotted line marks the first candidate; full ERM combines p_0 and r. (b) Candidate weighting with the same checkpoint, memory, and evaluation documents; the PPL axis is enlarged.

Memory strength and form. Figure 7 examines the continuation memory using the frozen six-block checkpoint and $K = 8$. The development run contains 256 calibration documents and 1,024 validation documents; the figure reports token-weighted validation PPL. In this run, p_0 uses energy-softmax candidate weights shared across all memory settings. We scale the retained-memory gate by 0.1, 0.3, or 0.6 and compare the normalized two-factor correction $\sqrt{p_0 r}$ with the mixture r alone. The two-factor correction improves on candidate averaging at all three strengths, reaching 25.772 PPL at 0.6, compared with 26.288 for candidate averaging and 27.152 for the mixture alone. Retaining the model-distribution factor thus supports a larger memory contribution.

We also compare suffix, skip-frame, and continuous-context memories with the same two-factor correction. Suffix memory combines training continuations with earlier occurrences in the document. Skip-frame memory matches pairs of earlier tokens separated by a gap; continuous-context memory retrieves continuations from similar earlier context vectors. All three use observed prefixes. Suffix memory gives the lowest PPL at strengths 0.3 and 0.6, with the best setting among the compared memories at 0.6. This agrees with the setting selected on the calibration documents and used in the full-proxy evaluation.

Candidate selection versus probability correction. The language extension changes the predictive distribution rather than selecting a single rollout. To examine this distinction, Table 7 compares memory-based selectors with the probability correction in Equation 10. The experiments use the frozen six-block S2 checkpoint at step 105,000, $K = 8$, noise 0.5, and seed 33000 on the same 33,000-document proxy (7,397,883 target tokens). PPL is computed as the exponential of total NLL divided by the number of target tokens.

The single-factor selector scores each candidate by soft retrieval against a lexical memory distribution and sums these scores over the document. The extended selector adds document and cache memories, factor weights, reliability gates, score standardization across candidates, and a relation-memory term. Both select one complete candidate distribution per document. We also report the lowest-PPL token selector among the tested variants: it combines lexical, document, and cache

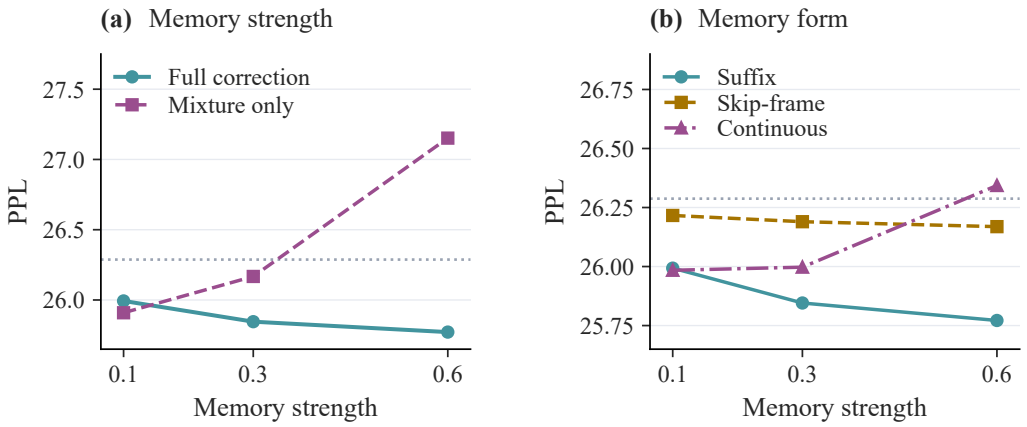


Figure 7: **Memory strength and form.** Six-block EBT, $K = 8$, and 1,024 validation documents from the development run, with energy-softmax candidate weights. (a) Two-factor correction (full correction) and mixture only, both using suffix memory. (b) Memory forms with the two-factor correction. The dotted line marks uniform candidate averaging; lower PPL is better.

factors with weights 1, 2, and 0.05, then selects a candidate separately at each position. These selectors apply different memory scores to the same candidate pool. Target tokens are used only for evaluation, except in the explicitly marked oracle diagnostic.

Table 7: **Candidate selection and probability correction on the language proxy.** Frozen six-block S2 EBT, $K = 8$; lower PPL is better. Document selection retains one candidate for the entire document; token selection may switch candidates across positions. The oracle is a diagnostic based on target NLL.

Method	PPL ↓
First candidate	44.188958
Uniform candidate average	44.147292
Single-factor memory, document selection	44.183465
Extended memory, document selection	44.182072
Weighted memory, token selection	44.126243
Document-selection oracle	44.003611
ERM probability correction (Equation 10)	**43.419165**

The extended document selector reduces PPL by 0.0014 relative to the single-factor selector. Token selection reaches 44.126243 PPL, and probability correction further reduces it to 43.419165, a 1.74% reduction from the first candidate. Combining model predictions with continuation memory provides the largest gain in this comparison.

The document-selection oracle minimizes total target NLL over the K candidate distributions separately for each document. Its 44.003611 PPL bounds document-level selection from that pool, but does not bound methods that construct a new distribution. ERM assigns probabilities through normalized $\sqrt{p_0 r}$ and can therefore improve on this oracle without using target tokens in its correction. This distinction explains why the language formulation goes beyond the answer selection used for structured reasoning.

ERM improves on candidate averaging at all 21 checkpoints (Figure 8(a)). At the final checkpoint, it improves on the first candidate for 76.15% of the 33,000 documents, showing that the aggregate gain extends across most documents (Figure 8(b)).

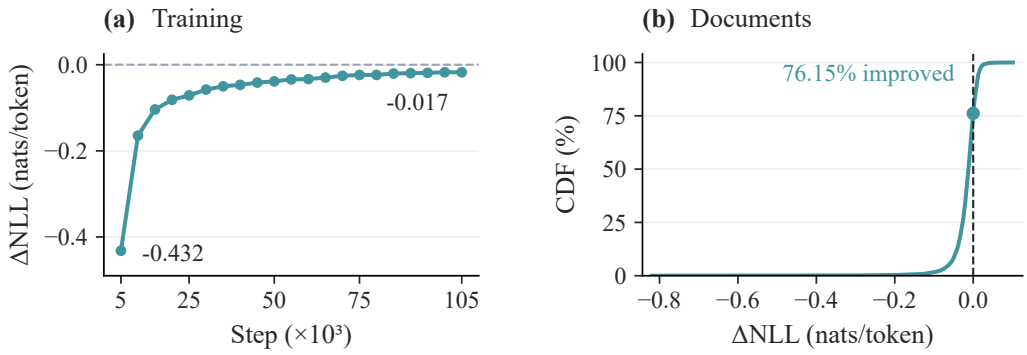


Figure 8: **Language gains across training and documents.** (a) Token-weighted ΔNLL (ERM minus candidate average) on a fixed 256-document slice. (b) Cumulative distribution of per-document ΔNLL (ERM minus first candidate) on the full 33,000-document proxy, with equal document weights. Negative values indicate improvement. Both panels use $K = 8$ and fixed memory and gate.

Gains across document lengths. Figure 9 groups the 33,000 proxy documents by the number of evaluated target tokens, up to the 256-token context limit. Each group uses total NLL divided by its total token count to compute PPL. With the fixed suffix memory and uniform candidate weights used in the main experiment, ERM reduces PPL relative to candidate averaging by 1.33%, 1.54%, 1.69%, and 1.65% across the four groups. All four paired-bootstrap intervals remain above zero, showing gains across the evaluated document lengths.

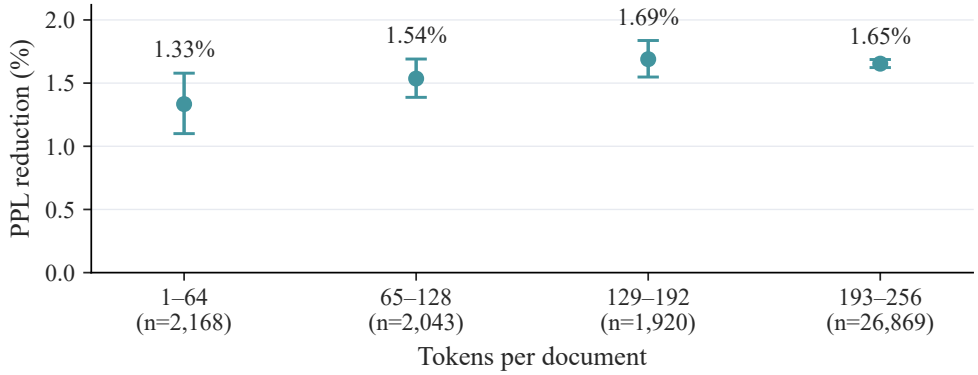


Figure 9: **Language gains by document length.** Relative reduction in token-weighted PPL from candidate averaging to ERM on the 33k proxy, with $K = 8$. Length is the number of evaluated target tokens; n is the number of documents. Error bars show 95% confidence intervals from 3,000 paired document bootstrap resamples within each group.

B.4 Language Correction with a Shared-Weight Recurrent EBT

We also evaluate a frozen S1 EBT trained for 105,000 steps, with four parameterized blocks reused over four cycles per energy evaluation. Table 8 uses $K = 8$, noise 0.5, seed 33000, and batch size four on the same 33,000-document proxy. All methods use the same candidate outputs. The internal-energy selector chooses the candidate with the lowest summed model energy over the document.

For this recurrent checkpoint, we use an adaptive mixture of memory-corrected predictions. It mixes six normalized predictions: a fixed memory-corrected baseline and five variants using suffix, skip-

frame, or continuous-context memories. Mixture weights depend on exponentially discounted prediction losses at earlier observed positions, with update scale 0.3 and decay 0.98; a 2% prior mixture prevents weight collapse. The weights reset for each document and use neither the current nor future target. Model parameters remain frozen.

Table 8: **Language correction on a shared-weight recurrent EBT.** Four blocks reused for four cycles, $K = 8$, and the full 33k proxy. The adaptive correction is a supplementary variant of the language method.

Method	PPL ↓
First candidate	51.100984
Uniform candidate average	50.841181
Internal-energy document selection	51.121273
Adaptive memory correction	**49.426066**

Adaptive correction lowers PPL by 3.28% relative to the first candidate and gives the lowest PPL among the compared methods. Memory correction therefore also improves prediction on this shared-weight recurrent checkpoint.

B.5 Energy-Guided Selection after ET Distillation

We test whether a learned ET correction improves candidate generation and provides an internal selection energy. The teacher is a 75,000-step EqR checkpoint with exponential-moving-average (EMA) weights. The student freezes its backbone and output heads, adding a shared ET module with 1.57 million trainable parameters. After each reasoning-module call, the intermediate state h receives $g[\mathrm{ET}_\phi(h) - h]$ before the damped update. ET takes one negative-gradient step on its attention and Hopfield energies; its projection and memory weights, step size, and gate g are trained. This experiment evaluates energy-based selection on the resulting student generator.

Training uses 10,000 augmentations of 1,000 puzzles, batch size 32, and AdamW with learning rate 5×10^{-5}. Token, halting, distillation, energy-descent, and energy-ranking losses have weights 1, 0.5, 1, 0.1, and 0.05. Distillation matches teacher predictions at the same recurrent depth using KL divergence at temperature 2, ramped over depths 12–16 with full weight at terminal steps. Descent penalizes energy increases; ranking favors lower energy for lower-NLL examples across a training batch.

We evaluate the step-2,000 EMA checkpoint from a 3,000-step run (decay 0.999). Validation/test use 2,048/4,096 disjoint puzzles from the official test pool. A validation sweep at $D = 16$ selects $g = 0.418\%$, reparameterized as $g = 0.005 \tanh(a)$ with a 0.5% bound. With paired random initial states and zero recurrent noise, enabling this gate raises single-trajectory test solves from 3226 to 3253 out of 4096.

For $K = 16$, teacher and student use matched random seeds (Figure 10(b,c)). ERM, ET internal energy, and summed EqR–ET internal energy recover all 3914 solves in the student pool, versus 3916 available to the teacher. Mean trajectory accuracy is 78.810% for the student and 79.065% for the teacher. Energy selection therefore recovers all available student solutions, while teacher and student have similar candidate coverage. These results use the same student training run as the gate experiment.

B.6 Task-Specific ERM Examples

Sudoku ERM performs exact soft retrieval over the valid memories of each structural unit. The row, column, and box memory sets follow the task's known permutation rules, with clues fixed to the input digits. A log-permanent dynamic program sums the probability assigned to every valid assignment within each unit. Summing the resulting retrieval energies gives the Sudoku score in Equation 6.

PPBench ERM uses a local-global decomposition for its different puzzle families. Local factors cover enumerable rules such as clue counts, forbidden adjacencies, no-2×2 patterns, and run-length

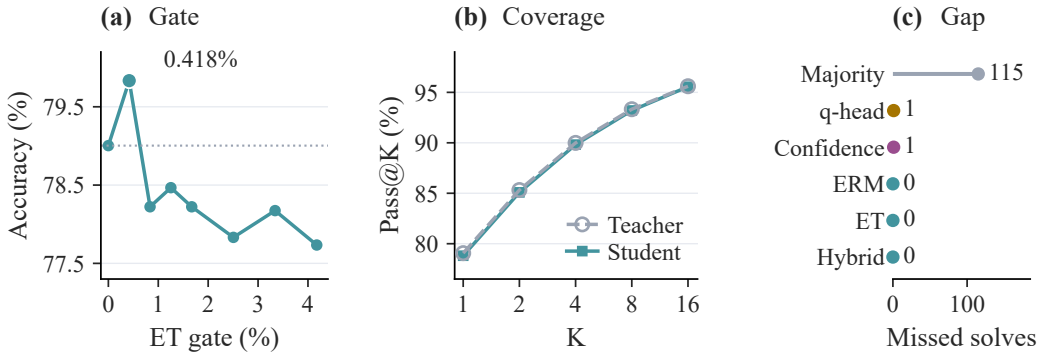


Figure 10: **Selection after ET distillation ($D = 16$).** (a) Validation exact accuracy versus effective gate; the dotted line marks the gate-disabled baseline. (b) Pass@K estimated from 16-candidate pools on 4,096 test puzzles. (c) Student solves missed relative to its oracle of 3914; majority uses whole-sequence voting, and ET/Hybrid use internal energies. Panels (b,c) use random latent initializations, zero recurrent noise, and seed 20260902; selectors share each model's candidate pool.

memories. Global energy terms account for connectivity and coverage. No separate confidence score is added to these task energies.

Maze ERM combines local path-degree memories with global path checks. The local factors score allowed patterns around terminal, open, and wall cells. The global term checks format, connectivity, degree, cycle, and length violations in the proposed path. These checks use the input maze and candidate assignment, without constructing a reference solution. Together, the terms evaluate the local and global constraints of a valid start-to-goal path.

B.7 Local and Global Energy Ablations

We evaluate the local and global terms of ERM on the full Maze-Unique test set (1,000 puzzles) and the PPBench held-out split (301 puzzles). Each task uses a frozen EMA checkpoint, at training step 135,000 for Maze and 85,800 for PPBench. We generate $K = 128$ independent candidates with maximum depth $D = 64$, batch size 16, and seed 0, using the original recurrent updates, random initial states, and noise scale 0.01. All selectors within a task use the same candidate pool. Energy selection is applied after generation, with ties resolved by the first candidate index. The separately generated single-trajectory baseline is also reported; it is evaluated before the shared pool, with batch sizes 128 for Maze and 512 for PPBench.

Full ERM retains both energy terms; the two ablations retain only the local or global term with its original weight. Maze uses local weight 0.25 and $\tau = 1$. Its global energy includes format, connectivity, degree, cycle, and path-length terms, as in Equation 8. PPBench uses $\beta = 128$, $\tau = 1$, and global weight 64. Removed factors are set to zero before the original per-puzzle average, preserving the normalization. Token-domain constraints belong to the local term. The PPBench Sudoku subset uses local permutation memories without a global term, so global-only selection takes the first candidate on that subset.

Tables 9 and 10 report exact accuracy, token accuracy over evaluated positions, and solved-puzzle counts. Whole-sequence majority selects the most frequent complete prediction. The oracle selects the first exact solution when one is present, otherwise the first candidate; its token accuracy follows this choice. Reference answers are used for metrics and the oracle only.

Maze. Full ERM and the global term each recover all 993 solutions available in the pool. Local-only selection solves 980, while q-head and confidence solve 979 and 978. Adding the global term to the local energy recovers 13 additional puzzles and retains all local-only solves. The global path energy thus supplies the decisive ranking information in this Maze evaluation.

The control selectors compare alternative rule scores and numerical precision. Unweighted violation count sums format errors, degree violations, disconnected nodes, a missing-goal indicator, and cycles. First-valid selection takes the first candidate satisfying all path rules, or the first candidate if none is valid; validity-then-confidence uses confidence within the valid set, or across all candidates if none is valid. Both checks use the proposed path and the input maze. The confidence control adds 0.001 times the mean token log-confidence to the negative full energy, and the float64 control combines the same global and local scores in double precision. Each control also solves 993 puzzles. Adding confidence leaves the selected candidate indices unchanged; double-precision combination preserves exact and token accuracy.

Table 9: **Maze energy ablations and selection controls.** Full 1,000-puzzle test set, $D = 64$, $K = 128$, seed 0. All rows except the separate baseline share the same candidates. Full ERM uses Equation 8 without a confidence term.

Method	Exact (%)	Token (%)	Count
Baseline (separate rollout)	89.60	99.117	896/1000
First candidate	91.20	99.280	912/1000
q-head	97.90	99.991	979/1000
Confidence	97.80	99.991	978/1000
Whole-sequence majority	97.70	99.933	977/1000
ERM (full energy)	**99.30**	99.987	993/1000
Global only	99.30	99.987	993/1000
Local only	98.00	99.987	980/1000
ERM + confidence	99.30	99.987	993/1000
ERM (float64 combination)	99.30	99.987	993/1000
Unweighted violation count	99.30	99.991	993/1000
First valid candidate	99.30	99.996	993/1000
Validity then confidence	99.30	99.993	993/1000
Oracle	99.30	99.996	993/1000

PPBench. Full ERM solves $265/301 = 88.04\%$, compared with 218 for global-only and 250 for local-only selection (Table 10). It retains every puzzle solved by either ablation and adds 47 solves relative to global-only and 15 relative to local-only selection. The gains are 15.61 and 4.98 percentage points, with 95% paired-bootstrap intervals of $[11.63, 19.93]$ and $[2.66, 7.64]$ from 5,000 resamples of the 301 puzzles in this run. Combining local and global energies therefore improves selection on PPBench.

The global-plus-domain control restores the token-domain factor while keeping other local factors removed. It gives the same 218 solves as global-only selection. The discrete rule score evaluates both local and global rules and solves the same 265 puzzles as full ERM. The candidate-pool oracle contains 266 solutions. Together, these controls support evaluating the complete set of task constraints.

Table 11 shows the contribution of each energy term across puzzle families. On Nurikabe, the global term recovers all 46 available solutions, compared with 39 for local-only selection. Local factors add two solves each on Heyawake and Lightup and six on Tapa relative to global-only selection. Among these four families with both local and global constraints, Tapa has the lowest candidate-pool oracle accuracy ($19/30 = 63.33\%$) and the largest gain from adding local factors. Full ERM raises its solve count from 13 (global only) and 15 (local only) to 19. These results support global energy for overall structural consistency and local memories for finer constraints, with a larger local contribution on the more challenging Tapa task. Combining the two terms gives a complete task energy that accounts for both levels of structure.

Table 10: **PPBench energy ablations and selection controls.** Full 301-puzzle held-out split, $D = 64$, $K = 128$, seed 0. All rows except the separate baseline share the same candidates. Global + domain restores only the token-domain factor; the discrete rule score includes both local and global rules.

Method	Exact (%)	Token (%)	Count
Baseline (separate rollout)	68.77	90.667	207/301
First candidate	67.44	90.376	203/301
q-head	82.72	94.952	249/301
Confidence	79.73	94.211	240/301
Whole-sequence majority	77.74	93.591	234/301
ERM (full energy)	**88.04**	96.330	265/301
Global only	72.43	91.691	218/301
Local only	83.06	95.416	250/301
Global + domain	72.43	91.691	218/301
Discrete rule score	88.04	96.541	265/301
Oracle	88.37	96.077	266/301

Table 11: **PPBench solved puzzles by type.** Counts from the same candidate pool as Table 10; column headers give the number of evaluated puzzles.

Method	Nurikabe (53)	Heyawake (72)	Lightup (51)	Tapa (30)	Sudoku (95)
First candidate	37	64	46	11	45
q-head	41	68	50	16	74
Confidence	40	67	49	12	72
Whole-sequence majority	39	67	47	11	70
ERM (full energy)	**46**	**68**	**50**	**19**	**82**
Global only	46	66	48	13	45
Local only	39	67	47	15	82
Global + domain	46	66	48	13	45
Discrete rule score	46	68	50	19	82
Oracle	46	69	50	19	82

B.8 Maze Depth Reduction and ERM Energy Stopping

The Maze energy curve in Figure 2 motivates evaluating smaller recurrent budgets. Table 12 reports $D = 16$ and $K = 128$ on the same 1000-example test split. ERM selects $999/1000 = 99.90\%$ exact solutions and matches this pool's oracle, while q-head and confidence select $994/1000 = 99.40\%$. The smaller depth therefore yields a pool with high solution coverage that ERM fully recovers.

Table 12: **Maze shared-candidate selection at $D = 16$, $K = 128$.** Exact and token accuracy are reported on the official 1000-example Maze-Unique test split.

Method	Exact	Token	Count
Baseline	81.40	98.20	814/1000
q-head	99.40	100.00	994/1000
Confidence	99.40	99.99	994/1000
ERM	**99.90**	99.99	999/1000
Oracle	**99.90**	99.99	999/1000

For the adaptive run in Table 4, PT maintains $K = 128$ replicas per input. At step t, the stopping statistic is the batch mean of the minimum candidate energy,

$$s_t = \frac{1}{B}\sum_{b=1}^{B} \min_k E(y_{b,k}^{(t)}; x_b), \qquad \delta_t = \frac{|s_t - s_{t-1}|}{\max(|s_{t-1}|, 10^{-6})}.$$

Checks begin at $t = 4$; the whole batch stops after two consecutive checks with $\delta_t < 0.01$, or at the maximum depth 64. The test set is processed in 63 batches of at most 16 examples. All batches stop by energy stability, with an unweighted mean depth of 7.603 (standard deviation 1.148, range 6–11). Exact accuracy is 99.60% and token accuracy is 99.95%. The checkpoint and its existing halting head are retained. Energy stability determines stopping for every batch, without training an additional controller. The reported compute reduction counts recurrent steps in the combined PT and stopping procedure; each step includes intermediate decoding and energy checks.

C Parameter and Compute Accounting

We use a simple rollout-based compute count in Table 13. Here D is the recurrent depth per trajectory, K is the number of parallel candidates or restarts, and the number of model evaluations is $D \times K$. Equivalent layers multiply model evaluations by the per-step equivalent-layer count used in TRM/EqR: 42 for the Sudoku/PPBench-style backbones and 15 for the small Maze backbone. ERM keeps the same neural rollout budget as the backbone and adds no learned neural parameters. The table counts neural model evaluations; memory retrieval, global-constraint evaluation, and repeated scoring for PT or stopping are additional operations. The last column records the final selector.

Table 13: **Parameter and compute accounting.** Model evaluations count recurrent outer-step evaluations; equivalent layers follow the TRM/EqR accounting.

Model/task	Params (M)	D	K	Model evals.	Eq. layers	Extra selector
TRM Sudoku	5.00	16	1	16	672	none
TRM attention	7.00	16	1	16	672	none
PTRM Sudoku	5.00	64	100	6,400	268,800	q-head
PTRM Maze	7.00	16	100	1,600	67,200	q-head
PTRM PPBench	7.00	48	100	4,800	201,600	q-head
EqR Sudoku	5.03	64	128	8,192	344,064	majority vote
EqR Maze	2.64	64	128	8,192	122,880	majority vote
ERM Sudoku	**5.03**	64	128	8,192	344,064	**Hopfield energies**
ERM PPBench	**7.12**	64	128	8,192	344,064	**Hopfield energies**
ERM Maze	**2.64**	64	128	8,192	122,880	**Hopfield energies**

D Code Release

We release the evaluation code used for the results in this paper as part of the supplementary material. It contains the selector harness for Sudoku-Lite, PPBench, and Maze-Unique, which covers shared-candidate and parallel-tempering selection for every method compared in Table 1, and the language routines for the four methods in Table 3 together with the suffix continuation memory builder. Each script takes a trained checkpoint and its dataset by command-line path and runs without further training.