CELLARC: An Oracle-Calibrated Benchmark for Few-Shot Rule Induction

Miroslav Lžičař [1]

Abstract

We introduce CELLARC, a controlled benchmark for few-shot local rule induction built from multicolor 1D cellular automata (CA). Each episode has five support pairs and one query serialized in ≤ 256 tokens, exposing explicit knobs for alphabet size k, radius r, rule family, Langton's λ, query coverage cov, and cell entropy H. The central goal is not to claim a broad measure of reasoning, but to create a small experimental system where evidence, ambiguity, and model failure can be separated. We release 95k training episodes plus two 1k test splits, evaluate symbolic, neural, recursive, and closed-model baselines, and add oracle-calibrated identifiability diagnostics. Under the unrestricted induced local-map class, query cells whose support windows are unseen are information-theoretically ambiguous, while seen windows are lookup-identifiable. We also implement structured posteriors for affine linear maps mod-k and one-step native totalistic/threshold families, exposing cells that are rule-identifiable even when lookup fails. This view explains why token accuracy alone can overstate progress and motivates exact-query, balanced, non-quiescent, and identifiability-conditioned metrics. Current results show that exact support-only structured solvers close the affine and native certifiable non-lookup strata, while seed-swept compact ICL baselines remain far below those certifying oracles. The released extrapolation split is difficulty-skewed rather than family-balanced, so we also provide generated family-balanced diagnostics; a lightweight learned family proposer plus exact verifier closes the generated affine diagnostic and partially closes native/family-balanced diagnostics.

This research was conducted independently, without the use of company resources. [1]Deep MedChem. Correspondence to: Miroslav Lžičař <lzicar2000@gmail.com>.

Accepted to the ICML 2026 CTB Workshop.

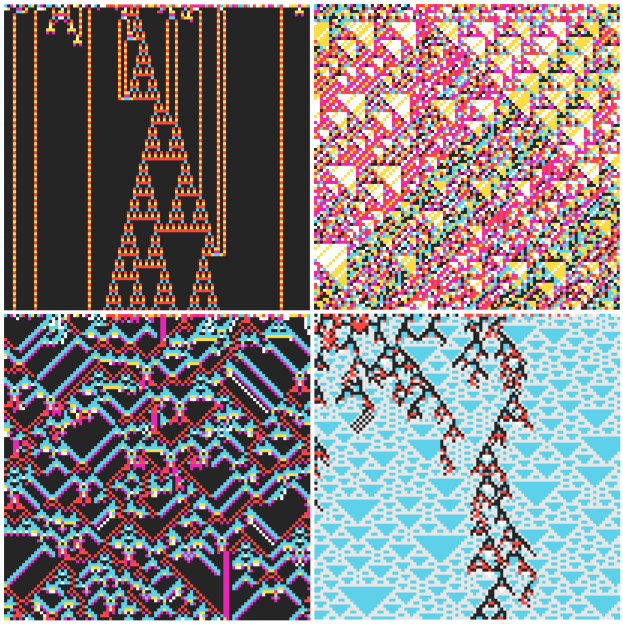

Figure 1. Illustrative CELLARC CA examples from diagnostic/test episodes. These examples show several rule families and dynamical regimes rather than summarizing the released TEST EXTRAPOLATION distribution. Top-left: outer–inner–totalistic, $\lambda = 0.310$, $H = 1.23$; top-right: linear-mod-k, $\lambda = 0.833$, $H = 2.57$; bottom-left: outer-totalistic, $\lambda = 0.500$, $H = 1.82$; bottom-right: totalistic, $\lambda = 0.808$, $H = 1.53$.

1. Introduction

The ARC-AGI Challenge (Kaggle, 2020) sparked broad interest in systematic generalization (Chollet, 2019), yet progress has been slower than expected (ARC Prize, 2024; Chollet et al., 2024). We see five recurring obstacles. *Creative bottleneck*: even with ARC-AGI-2's $\sim$1,000 curated tasks and automated instance augmentation (e.g., RE-ARC (Hodel, 2024)), inventing *new* rule families at scale remains labor- and expertise-intensive (Chollet et al., 2025). *Anthropocentric bias*: human-authored tasks encode incidental priors (symmetries, grid regularities, hand-crafted color patterns) that may not reflect the broader hypothesis space. *Compute barriers*: episodes can span thousands of tokens or require heavy program search, limiting rapid iteration with small models or heavier training regimes (e.g. in-context learning or meta-learning). *Saturation risk*: as leaderboards converge, new releases demand fresh handcrafted tasks, rais-

Resource	URL
Leaderboard	cellarc.mireklzicar.com
Dataset	github.com/mireklzicar/cellarc
Models	github.com/mireklzicar/cellarc_baselines
HF Dataset	mireklzicar/cellarc_100k
HF Meta	mireklzicar/cellarc_100k_meta

Table 1. CELLARC resources and code repositories. All code is licensed under Apache License 2.0; all data is licensed under CC BY 4.0.

ing cost and drift. *Difficulty caps*: tasks are intentionally human-solvable, good for interpretability, but a ceiling once models approach or exceed human performance.

In CELLARC, a synthetic benchmark that measures *local rule discovery efficiency* from compact supervision, tasks are generated from multicolor one-dimensional cellular automata (CA), giving essentially unbounded novel tasks with tunable complexity while decoupling the latent rule class from human priors (Langton, 1990). Each episode contains five support pairs and one query serialized in ≤ 256 tokens, enabling rapid iteration with small models and comparable evaluations across symbolic and neural approaches. Representative CA traces are shown in Figure 1, with qualitative grids in Figure 8 grounding the benchmark in concrete examples.

Design principles.

(G1) **Generative**: unlimited sampling of novel tasks.

(G2) **Controllable**: explicit knobs for complexity and difficulty via alphabet size k, radius r, rule family, Langton's λ, coverage cov, and cell entropy H.

(G3) **Budget-aware**: short sequences (≤ 256 per episode) enable training with lower memory and iterating fast.

(G4) **Non-anthropocentric**: structure arises from formal CA rules rather than human sketches.

(G5) **Transparent**: splits are measurable and reproducible, test splits exist both for interpolation (same distribution of CA as training) and extrapolation (out of distribution in terms of difficulty).

Key public assets (dataset, code, leaderboard, and mirrors) are summarized in Table 1.

2. Cellular Automata for Abstraction and Reasoning

A one-dimensional cellular automaton (CA) is specified by an alphabet of k states, a radius r, and a synchronous local rule $F : [k]^{2r+1} \to [k]$ applied in parallel to every cell (Hedlund, 1969). The model traces back to von Neumann's program on self-reproduction and computation in discrete media (von Neumann, 1966). CA exhibit a broad spectrum of dynamical regimes—from ordered to chaotic—characterized in classical analyses and monographs (Wolfram, 1983; 2002; Ilachinski, 2001). We use rule families with well-defined local semantics (totalistic, outer/inner-totalistic, linear-mod-k) (Eppstein, 2010; Martin et al., 1984) and steer dynamics with Langton's λ parameter to traverse order–chaos transitions (Langton, 1990).

Formal substrate. CA have long been used as formal models of local computation and pattern formation (Turing, 1990; Wolfram, 2002). We use them here only as an exactly inspectable substrate: CELLARC does not claim external-world fidelity, but uses CA because their local rules, symmetries, and induced hypothesis classes can be audited directly.

Why CA are a good substrate for benchmarking generalization. The following properties complement the design goals in §1 without relying on human-crafted priors:

1. **Calibrated complexity.** Varying k, r, rule family, and λ produces smooth difficulty gradations—from additive dynamics to chaos—while empirical cell entropy H provides a complementary, data-driven summary of regime complexity (Wolfram, 1983; Eppstein, 2010; Martin et al., 1984; Langton, 1990; Ilachinski, 2001).

2. **Finite local hypothesis class.** Each update depends only on a length-$2r+1$ window, yielding a compact, structured search space. This supports both symbolic solvers (e.g., de Bruijn analyses of window transitions) and neural models that can target the same locality bias (de Bruijn & Erdős, 1948; Sutner, 1991).

3. **Exact measurability.** Because the local rule is formal, we can *exactly* account for which windows appear in supervision (via de Bruijn constructions) and report $\mathrm{cov}/\lambda/H$, replacing opaque "difficulty" labels with reproducible diagnostics (de Bruijn & Erdős, 1948; Sutner, 1991).

4. **Unlimited instances under short contexts.** CA histories admit unlimited episodes while keeping supervision compact; by constraining $k \in \{2, \ldots, 6\}$ and $r \in \{1, 2, 3\}$, we span low- to high-entropy rules in token budgets suitable for efficiency-centric evaluation.

5. **Mathematical symmetry.** CA are translation-equivariant and time-homogeneous—the same local rule applies at every site and step (Hedlund, 1969). Many families admit additional symmetries (left–right reflection,

state relabeling, additive structure for linear mod-k rules) that induce equivalence classes over rules and traces, enabling targeted tests of systematic generalization (Martin et al., 1984; Eppstein, 2010; Sutner, 1991).

We leverage these properties in CELLARC: episodes are sampled across families and λ regimes; difficulty is controlled through window coverage cov and palette size k; and evaluation focuses on how quickly a model infers the hidden local rule under a fixed budget, not merely whether it eventually does (Chollet, 2019).

3. Related Work and Positioning

CELLARC is closest in spirit to ARC-style abstraction benchmarks, but it is deliberately narrower. ARC-AGI and RE-ARC emphasize human-authored or procedurally augmented visual abstraction tasks (Chollet, 2019; Kaggle, 2020; Hodel, 2024; Chollet et al., 2025); CELLARC instead isolates one capability–few-shot induction of a local rule–so ambiguity, support coverage, and exact-query failure can be measured directly. Program-synthesis systems such as DreamCoder also infer compact rules from examples, but they optimize over richer libraries and search procedures, whereas CELLARC fixes a local rule substrate to make identifiability auditable (Ellis et al., 2021). Classical CA work supplies the formal local-dynamics substrate (Hedlund, 1969; Wolfram, 1983; Langton, 1990; Ilachinski, 2001), while de Bruijn graph analyses motivate our lookup and coverage diagnostics (de Bruijn & Erdős, 1948; Sutner, 1991). The neural baselines connect to few-shot prompting (Brown et al., 2020), neural algorithmic reasoning (Veličković & Blundell, 2021), and sequence/automata-learning models (Graves et al., 2014; 2016; Kaiser & Sutskever, 2016; Delétang et al., 2023). Because procedurally generated benchmarks can reward shortcuts and generator-specific priors, we label access regimes, subset provenance, and limitations rather than presenting a single aggregate score as broad capability evidence (Geirhos et al., 2020). The contribution is therefore a benchmark and diagnostic harness, not a new general reasoning model or a replacement for broader ARC-style suites.

4. The CELLARC Benchmark

High-level dataset statistics appear in Table 2; we revisit the notation in Table 3.

4.1. Dataset Generation Pipeline

We follow the reproducible workflow documented in the dataset repository. The pipeline has four phases: (i) raw pool synthesis, (ii) filtering and metadata enrichment, (iii) coverage-aware resampling and split assignment, and (iv) packaging.

Metric	Value
Rule-space	
Alphabet size k (range)	2–6
Window length $C{=}2rt{+}1$ (range)	3–19
Radius r (range)	1–3
Steps t (range)	1–3
Neighborhood size $2r{+}1$ (range)	3–7
Episode size	
Training examples / episode	5
Median sample length	13
Median flattened episode length	156
Max flattened episode length	256

Table 2. Basic statistics of the CELLARC-100k dataset.

Raw pool synthesis. We sample multicolor one-dimensional CA across alphabet sizes $k \in \{2, \ldots, 6\}$ and radii $r \in \{1, 2, 3\}$. For each sampled rule we generate five support input–output pairs and one query, using a mixture of *cycle*, *unrolled*, and *hybrid* constructions. To span regimes from easy to challenging while discarding trivial cases, we draw a *raw support-span coverage* target uniformly in $[0.05, 0.95]$ and sample support spans accordingly over the de Bruijn cycle (fraction of cycle length covered by the union of support spans). This stage yields roughly 5.1×10^5 raw episodes. Pool diagnostics verify that the family mix and coverage histogram match the target before proceeding.

Filtering and enrichment. We downsample the raw pool to about 110k high-quality episodes while enforcing fingerprint-level deduplication, novelty of the query solution relative to supports, and a flattened supervision length of at most 256 tokens. Downsampling is balanced over a two-dimensional histogram of Langton's λ and observed coverage. A first sanity check removes degenerate absorbing rules (e.g., simultaneously very low λ and entropy). The surviving episodes are then replayed at a fixed width and horizon to compute enriched descriptors: Langton's λ, mean cell entropy H, mutual information, morphology statistics, and inline rule tables.

Coverage-aware resampling Next, to avoid prevalence of unsolvable episodes, we rewrite training windows so that the five supports collectively cover the query with query-weighted coverage $\mathrm{cov} \geq 0.5$ in the released data, while avoiding leakage (the exact query sequence is never copied into supports). Episodes that cannot satisfy the constraint within a bounded sampling budget are dropped, leaving roughly 95k–100k items. We then keep only the compliant episodes and partition them deterministically.

Splits and packaging. First of all the TEST EXTRAPOLATION split is populated with the most difficult CA: *lowest* cov and *highest* λ and H. By this construction, the released TEST EXTRAPOLATION split consists entirely of chaotic episodes (100% in the chaotic Langton-λ bin). The remainder are shuffled with a fixed seed to carve out TRAIN, VALIDATION, and TEST INTERPOLATION subsets of similar distribution. Fixed 100-episode test subsets are provided for API-metered closed-model diagnostics and quick inspection; they are not used as the primary benchmark score. To verify a result (or in case of saturation suspicion) it is possible to generate more CAs of corresponding families and parameters using the artifact generator. Both JSONL and Parquet views are released with fixed 100-episode subsets, checksums, CC BY 4.0 package licensing, and Croissant/RAI metadata. The public code repository keeps the generator, smoke checks, and local manifest verification path; the Hugging Face mirrors listed in Table 1 provide dataset and metadata access.

4.2. Difficulty Knobs

We expose explicit sampling and filtering knobs whose ranges are fixed by the pipeline above (see Table 3 for symbol definitions). Unless noted otherwise, knobs are set once per episode and enforced during filtering/splitting; replay width n used for entropy/statistics is fixed and not used as a knob.

- **Alphabet size k.** Uniform over $[2, 6]$, balancing binary rules and higher-entropy palettes.
- **Radius r.** Most probability mass on $r{=}1$ (about 70%) with tails to $r{=}2$ and $r{=}3$.
- **Temporal horizon t and windowing C, h.** The prediction gap t is set by the episode; together with r it determines the centered supervision window $C{=}2rt{+}1$ and half-window $h{=}rt$. We cap the flattened supervision length to at most 256 tokens, indirectly bounding feasible (r, t) pairs.
- **Rule families.** Cycle, unrolled, and hybrid constructions encompass random, totalistic, outer/inner-totalistic, threshold, and linear-mod-k rules. Under the highest-λ/highest-entropy selection used for TEST EXTRAPOLATION, retained mass concentrates in Totalistic and Linear mod-k families.
- **Complexity and dynamics (λ, H).** Filtering and split assignment operate on Langton's activity λ and mean per-cell entropy H to cover ordered, edge-of-chaos, and chaotic regimes; TEST EXTRAPOLATION preferentially receives the highest-λ and highest-H items (in the released dataset this yields an *entirely* chaotic extrapolation split; 100% by λ bin).
- **Query-weighted coverage** cov**.** Supports are rewritten so the centered C-windows cover the query with score $\mathrm{cov} \geq 0.5$ in released data while avoiding leakage; TEST EXTRAPOLATION collects the lowest-coverage items.

Evaluation metrics. For continuity with prior runs, we still report token (Hamming) accuracy $\mathrm{Acc}_{\mathrm{tok}}$ over the query length. Because token accuracy can be inflated by frequent or quiescent states, the main result table also reports exact-query and balanced token accuracy, while the artifact exports exact-episode, non-quiescent, support-consistency, and identifiability-conditioned metrics. We report scores separately on *interpolation* and *extrapolation* test splits and summarize generalization with the interpolation–extrapolation gap Δ. Deterministic symbolic baselines and prediction-log subsets are accompanied by bootstrap intervals over episodes; seed-swept ICL rows report mean and standard deviation across three seeds, while unswept neural rows remain appendix/context diagnostics.

Proposition 4.1 (Lookup identifiability and unrestricted ambiguity)**.** *Fix an episode with support pairs S and an induced t-step local map $g : [k]^C \to [k]$, where $C = 2rt{+}1$. Assume the support constraints contain no conflict: every observed centered window has a unique observed label. For any query cell with centered window $w^\star$, if $w^\star$ appears in the support-window set, its label is identifiable by lookup. If $w^\star$ is absent, then for the unrestricted class of maps g every label in $[k]$ remains compatible with the support, and a uniform prior over unconstrained table entries gives Bayes token success probability $1/k$.*

Proof. Observed support windows impose equality constraints on the corresponding entries of the table for g. A seen query window reuses one constrained entry, so its label is the unique support label. An unseen query window indexes an unconstrained table entry; assigning any value in $[k]$ to that entry leaves all support constraints unchanged. Under the uniform independent prior on unconstrained entries, the posterior over that cell label is uniform. □

This proposition is intentionally about the unrestricted induced map. The affine and native-family posterior rows in Table 5 add a smaller hypothesis class; they can certify some unseen windows precisely because they assume more structure than unrestricted lookup.

5. Main Results Summary

Protocol. We evaluate symbolic, neural, recursive, and closed-model baselines under a shared five-support, one-query interface. The main comparison separates support-label methods that may inspect unlabeled query inputs, clean in-context neural models, and archived task-embedding di-

CA parameters

k Alphabet size (number of colors/states).

r Neighborhood radius; local rule $F : [k]^{2r+1} \to [k]$.

t Temporal horizon (steps between input and target).

C Supervision/window length for centered neighborhoods. $C = 2rt + 1$, with half-window h.

h Half-window size rt (number of cells on either side).

Complexity and Dynamics

λ Langton's activity with quiescent state q.

$$\lambda = \frac{1}{k^{2r+1}} \sum_{\mathbf{u} \in [k]^{2r+1}} \mathbf{1}\{F(\mathbf{u}) \neq q\}.$$

H Mean per-cell Shannon entropy (base 2) over a diagram $X \in [k]^{(t+1)\times n}$.

$$H = \frac{1}{n} \sum_{j=1}^{n} \Big(-\sum_{s=0}^{k-1} p_j(s) \log_2 p_j(s) \Big),$$ where $p_j(s)$ is the empirical state frequency at column j.

cov Query-weighted window coverage Q is the multiset of centered C-windows in the query, S the set of centered C-windows from the support inputs, where q(w) counts multiplicity in Q.

$$\text{cov} = \frac{\sum_w q(w)\, \mathbf{1}\{w \in S\}}{\sum_w q(w)}$$

Evaluation metrics

Acc_{tok} Token (Hamming) accuracy for query length L.

$$\text{Acc}_{\text{tok}} = \frac{1}{L} \sum_{i=1}^{L} \mathbf{1}\{\hat{y}_i = y_i\}.$$

Δ Interpolation-extrapolation gap

$$\Delta = \text{Acc}_{\text{interp}} - \text{Acc}_{\text{extra}}.$$

Table 3. Notation and metrics. q is the quiescent state (default 0); n is the spatial width used for H. Symbols in the text hyperlink back to this table.

agnostics whose held-out support rows can optimize per-episode state.

Term	Meaning
Raw span coverage	Generator-side target: fraction of the de Bruijn cycle deliberately exposed by support spans before conditioning on a query.
Observed support coverage	Unique support windows divided by the episode's de Bruijn cycle size; this is a global exposure statistic, not query-specific.
Query-weighted coverage	Fraction of query-centered windows that appear in support inputs, counting repeated query windows. This is the main coverage knob cov.
Lookup-identifiable	Query cells whose exact centered window appears in support with a unique target.
Ambiguous unrestricted	Query cells whose window is unseen; under an unrestricted local map the support imposes no label constraint.
Support conflict	A support window appears with multiple labels. In the audited release this count is zero on the reported test splits.
Support inconsistent	A unique support lookup disagrees with the query label. The audited cases are boundary/context cells from cropped spans, not core deterministic CA conflicts.

Table 4. Coverage and identifiability terminology used in diagnostics.

Model	Regime	N	Interpolation	Extrapolation	Δ	Params (M)
GPT-5 High	Closed LLM	100	62.3	**48.1**	14.2	–
Transformer + de Bruijn	Oracle Ensemble	100	**65.4**	35.5	29.9	9.7
Transformer	Task Embedding	100	61.2	28.1	33.1	9.7
Transformer	Task Embedding	1000	**58.0**	**32.4**	25.6	9.7
Transformer-ACT	Task Embedding	1000	50.8	27.2	23.6	8.9
TRM	Task Embedding	1000	48.7	26.4	22.3	9.5
HRM	Task Embedding	1000	50.8	28.2	22.6	11.2
LSTM	Task Embedding	1000	51.0	27.9	23.1	9.6
1D CNN	Task Embedding	1000	52.7	29.0	23.7	10.3
Transformer	In Context (ICL)	1000	**51.0**	28.3	22.7	9.7
Transformer-ACT	In Context (ICL)	1000	**51.0**	**30.3**	20.7	8.9
TRM	In Context (ICL)	1000	50.9	29.0	21.9	9.5
LSTM	In Context (ICL)	1000	50.8	27.5	23.3	9.6
HRM	In Context (ICL)	1000	50.9	30.1	20.8	11.2
1D CNN	In Context (ICL)	1000	46.7	18.6	28.1	10.3
de Bruijn	Symbolic	1000	**52.5**	**29.8**	22.7	≈0
Most Frequent	Symbolic	1000	50.4	28.2	22.2	≈0
Copycat	Symbolic	1000	32.7	18.0	14.7	≈0
Random	Symbolic	1000	29.7	17.6	12.0	≈0

Table 5. Main results on CELLARC (large models, ∼10M parameters). Per-token accuracy (%) is shown on the interpolation and extrapolation splits. Rows with N=100 use the same 100-task subsets as the closed LLM; rows with N=1000 evaluate the full 1k splits. The Oracle Ensemble row reports a per-episode maximum between Transformer-TE and de Bruijn accuracies and is not a deployable method. Symbolic baselines have no learned parameters.

Findings. Among trained full-split baselines, the strongest large model is the vanilla encoder-only Transformer with task embeddings, reaching 58.0% on TEST INTERPOLATION and 32.4% on TEST EXTRAPOLATION. The de Bruijn symbolic lookup baseline is strong for a no-training method (52.5/29.8), while most recursive and convolutional variants trail the Transformer on at least one split. On the fixed 100-episode diagnostic subset, GPT-5 High reaches 62.3/48.1%, and the non-deployable Transformer+de Bruijn oracle ensemble reaches 65.4/35.5%, highlighting complementary neural and symbolic failure modes. These results support the benchmark's intended role: separating lookup-solvable evidence, rule-identifiable evidence, and genuinely ambiguous cells.

Result provenance. Full-split symbolic rows are deterministic support-only evaluations. Neural rows summarize the large-preset training regime; in-context rows use the same benchmark splits while task-embedding rows are diagnostic because they carry per-episode state. The GPT-5 row is a closed-model diagnostic only: cached prompts/responses cover the fixed interpolation/extrapolation subset records and should not be treated as a reproducible full-benchmark leaderboard result.

Takeaway. CELLARC is not a broad reasoning score. It is a controlled, reproducible stress test for local rule induction under partial evidence, with metadata that identifies which failures are due to insufficient information and which reflect model inference errors. Detailed methods, full diagnostic tables, plots, and additional limitations are provided in the appendix.

Impact Statement

CELLARC is a synthetic benchmark intended to make local-rule induction failures easier to diagnose, not a proxy for broad intelligence or human reasoning ability. Misuse risks include overinterpreting a single score, treating closed-model subset diagnostics as stable leaderboard evidence, or optimizing models toward narrow CA regularities without checking transfer. The release includes exact metadata, uncertainty labels, and generator code to support reproducible audits. Training compact baselines is modest by contemporary standards, but API-based closed-model evaluation and large neural sweeps can still carry environmental and financial costs; the fixed smoke and 100-episode subsets are provided to reduce unnecessary reruns.

References

ARC Prize. Arc prize—leaderboard, 2024. URL https://arcprize.org/leaderboard.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

Choffrut, C. A characterization of subsequential functions. *Theoretical Computer Science*, 292(1):135–150, 2003.

Chollet, F. On the measure of intelligence. *arXiv*, 2019. doi: 10.48550/arXiv.1911.01547. URL http://arxiv.org/abs/1911.01547.

Chollet, F., Knoop, M., Kamradt, G., and Landers, B. Arc prize 2024: Technical report, 2024. URL https://arxiv.org/abs/2412.04604.

Chollet, F., Knoop, M., Kamradt, G., Landers, B., and Pinkard, H. Arc-agi-2: A new challenge for frontier ai reasoning systems, 2025. URL https://arxiv.org/abs/2505.11831.

Cook, M. Universality in elementary cellular automata. *Complex Systems*, 15(1):1–40, 2004.

de Bruijn, N. G. and Erdős, P. On a combinatorial problem. *Proceedings of the Section of Sciences of the Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam*, 51(10): 1277–1279, 1948.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, Ł. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1807.03819.

Delétang, G., Ruoss, A., Grau-Moya, J., Genewein, T., Wenliang, L. K., Catt, E., Cundy, C., Hutter, M., Legg, S., Veness, J., and Ortega, P. A. Neural networks and the chomsky hierarchy. In *International Conference on Learning Representations*, 2023. doi: 10.48550/arXiv.2207.02098. URL https://openreview.net/forum?id=WbxHAzkeQcn. ICLR 2023 notable top 25%.

Ellis, K., Wong, C., Nye, M., Sable-Meyer, M., Cary, L., Morales, L., Hewitt, L., Solar-Lezama, A., and Tenenbaum, J. B. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pp. 835–850, 2021. doi: 10.1145/3453483.3454080. URL https://doi.org/10.1145/3453483.3454080.

Eppstein, D. Growth and decay in life-like cellular automata. In Adamatzky, A. (ed.), *Game of Life Cellular Automata*, pp. 71–97. Springer London, 2010. ISBN 9781849962179. doi: 10.1007/978-1-84996-217-9_6. URL http://dx.doi.org/10.1007/978-1-84996-217-9_6.

Faldor, M. and Cully, A. CAX: Cellular automata accelerated in JAX. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=o2Igqm95SJ. ICLR.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2:665–673, 2020. doi: 10.1038/s42256-020-00257-z. URL https://doi.org/10.1038/s42256-020-00257-z.

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

Graves, A., Wayne, G., and Danihelka, I. Neural turing machines, 2014.

Graves, A., Wayne, G., et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538:471–476, 2016.

Hedlund, G. A. Endomorphisms and automorphisms of the shift dynamical system. *Mathematical Systems Theory*, 3(4):320–375, 1969. ISSN 1433-0490. doi: 10.1007/BF01691062. URL https://doi.org/10.1007/BF01691062.

Hodel, M. Addressing the abstraction and reasoning corpus via procedural example generation, 2024. URL https://arxiv.org/abs/2404.07353.

Hopcroft, J. E., Motwani, R., and Ullman, J. D. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 3 edition, 2006.

Ilachinski, A. *Cellular Automata: A Discrete Universe*. World Scientific, Singapore, 2001. doi: 10.1142/4702. URL https://www.worldscientific.com/doi/abs/10.1142/4702.

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

Kaggle. Abstraction and reasoning challenge, 2020. URL https://www.kaggle.com/competitions/abstraction-and-reasoning-challenge. Kaggle Competition.

Kaiser, Ł. and Sutskever, I. Neural gpus learn algorithms. In *International Conference on Learning Representations*, 2016.

Langton, C. G. Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D: Nonlinear Phenomena*, 42(1–3):12–37, 1990. doi: 10.1016/0167-2789(90)90064-V.

Martin, O., Odlyzko, A. M., and Wolfram, S. Algebraic properties of cellular automata. *Communications in Mathematical Physics*, 93(2):219–258, 1984. doi: 10.1007/BF01223745.

Mealy, G. H. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 34(5):1045–1079, 1955.

Mohri, M. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms, 2018a. URL https://arxiv.org/abs/1803.02999.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018b. URL https://arxiv.org/abs/1803.02999.

Shazeer, N. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. URL https://arxiv.org/abs/2002.05202.

Su, J., Ahmed, M., Lu, Y., Pan, S., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Sutner, K. de bruijn graphs and linear cellular automata. *Complex Systems*, 5(1):19–30, 1991.

Turing, A. M. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

Turing, A. M. The chemical basis of morphogenesis. *Bulletin of Mathematical Biology*, 52(1):153–197, 1990. doi: 10.1007/BF02459572. Reprint of 1952 article in Philosophical Transactions of the Royal Society of London. Series B.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Veličković, P. and Blundell, C. Neural algorithmic reasoning. *Patterns*, 2(7):100273, 2021. doi: 10.1016/j.patter.2021.100273. URL https://arxiv.org/abs/2105.02761.

von Neumann, J. *Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana, IL, 1966. Edited by Arthur W. Burks.

Wang, G., Li, J., Sun, Y., Chen, X., Liu, C., Wu, Y., Lu, M., Song, S., and Abbasi-Yadkori, Y. Hierarchical reasoning model, 2025. URL https://arxiv.org/abs/2506.21734.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Wolfram, S. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55(3):601–644, July 1983. doi: 10.1103/RevModPhys.55.601. URL https://link.aps.org/doi/10.1103/RevModPhys.55.601.

Wolfram, S. *A New Kind of Science*. Wolfram Media, 2002. ISBN 1-57955-008-8.

Zhang, B. and Sennrich, R. Root mean square layer normalization. *arXiv preprint arXiv:1910.07467*, 2019. URL https://arxiv.org/abs/1910.07467.

Appendix

A. Additional Benchmark Details

A.1. Sampling Procedure

We generate each CELLARC episode by controlling which centered windows appear in supervision while preventing leakage of the query solution. The procedure mirrors the artifact generator and uses coverage-aware selection indexed by de Bruijn cycles (de Bruijn & Erdős, 1948; Sutner, 1991) (consistent with the Curtis–Hedlund–Lyndon characterization of 1D CA (Hedlund, 1969)). Unrolling CA is done using CAX, a library accelerating CA operations in JAX with optional GPU support.(Bradbury et al., 2018; Faldor & Cully, 2025)

Setup and notation. Fix an alphabet size k, radius r, prediction gap t, and thus a centered supervision window

$$C \;=\; 2\,r\,t+1, \qquad h \;=\; r\,t.$$

For a length-n 1D configuration $x \in [k]^n$ and index j, let $\mathrm{win}_C(x, j)$ be the length-C word centered at j with periodic boundary conditions. Given supports $S = \{(x_i, y_i)\}_{i=1}^{5}$ where $y_i = F^t(x_i)$, and a query $(x^\star, y^\star)$ with $y^\star = F^t(x^\star)$, define the multiset of query windows

$$\mathcal{Q} \;=\; \big\{\mathrm{win}_C(x^\star, j) \;:\; j = 1, \ldots, n\big\},$$

and the support window set

$$\mathcal{S} \;=\; \bigcup_{i=1}^{5} \big\{\mathrm{win}_C(x_i, j) \;:\; j = 1, \ldots, n\big\}.$$

Our query-weighted coverage is

$$\mathrm{cov} \;=\; \frac{1}{|\mathcal{Q}|} \sum_{w \in \mathcal{Q}} \mathbf{1}[\, w \in \mathcal{S}\,],$$

i.e., the fraction of query-centered windows that are seen in the supports (counting multiplicity in $\mathcal{Q}$). Leakage is avoided by forbidding any exact duplication of the query pair $(x^\star, y^\star)$ in the supports.

Construction modes. We use three interchangeable modes for drawing training/query rows: *cycle* uses a de Bruijn cycle (de Bruijn & Erdős, 1948) so every length-C word over $[k]$ appears once; *unrolled* draws i.i.d. initial conditions $x \sim \mathrm{Unif}([k]^n)$; *hybrid* mixes both (by default, 50:50). The cycle is *indexed* implicitly by positions on the order-C de Bruijn graph; we never materialize a length k^C word.

Inputs. The generator takes: ranges for k, r, t; a supervision budget (max flattened tokens, default 256); a mixture over rule families; a target coverage $\tau \in [0, 1]$; a construction mode; and a bounded resampling budget B (default B=64). A per-episode PRNG seed ensures determinism.

Procedure.

1. **Sample structural params.** Draw k from the configured range with a decreasing categorical prior (bias to smaller palettes); pick C under the supervision budget; choose (r, t) s.t. $2rt{+}1 = C$ with caps $r \leq r_{\max}$, $t \leq t_{\max}$.

2. **Instantiate a rule.** Sample a rule family from the mixture (totalistic, outer/inner-totalistic, threshold, linear-mod-k, or dense table), then draw a rule and record its Langton activity λ and quiescent state.

3. **Prepare sources of rows.**
   - *Cycle*: choose a starting index on the order-C de Bruijn cycle; the initial row x is the corresponding cyclic word; evolve for $t{+}1$ steps to obtain (x, y) with $y{=}F^t(x)$.
   - *Unrolled*: sample $x \sim \mathrm{Unif}([k]^n)$ and set $y{=}F^t(x)$.
   - *Hybrid*: mix the two sources by the configured probability.

4. **Pick supports to hit coverage.** Draw candidate rows and assemble five support pairs (x_i, y_i) while greedily maximizing cov toward the target τ. Stop when either $\mathrm{cov} \geq \tau$ or B draws are exhausted.

5. **Select and validate the query.** Draw $(x^\star, y^\star)$ using the same mode but withhold it from supports. Apply the leakage check after the query is fixed: if any support row duplicates the exact query pair, resample the query/support choice within the remaining budget. If cov computed against this query falls below the release threshold (e.g., ≥ 0.5 for training data), optionally resample up to the remaining budget.

6. **Compute metadata and checks.** Replay at fixed width n to collect descriptors (λ, mean cell entropy H, mutual information, morphology stats), compute the episode fingerprint (hash of rule table + spans + seed), and apply rejections (absorbing/degenerate rules; supervision length $>$ budget; cov outside split-specific bounds).

7. **Return the episode.**

Randomness. All random choices are made by a keyed PRNG; rerunning with the same seed reproduces bit-identical episodes. De Bruijn access is $O(1)$ per window via index arithmetic; episode construction is linear in the number of evolved cells ($O(n\,t)$) plus the bounded resampling loop ($\leq B$ proposals).

A.2. Identifiability Diagnostic

The key advantage of CELLARC over less controlled reasoning benchmarks is that the latent local hypothesis class is explicit. We therefore compute a first oracle-calibrated diagnostic for the unrestricted induced local-map class. Given support pairs S and centered windows of width C, a query cell is *lookup-identifiable* if its query window appears in the supports with a unique target. If the query window is unseen, then under an unrestricted local map the support imposes no constraint on that cell's output; with a uniform prior, the Bayes-optimal token success probability is $1/k$. We reserve *support conflict* for the case where the same support window appears with multiple labels.

For each episode we export the fraction of lookup-identifiable cells, the fraction ambiguous under the unrestricted map, and the unrestricted Bayes token/exact-query probabilities. This remains a conservative oracle for unconstrained local maps: it certifies irreducible ambiguity when the exact query window is unseen, while also identifying cells solved by pure lookup.

We then add structured posteriors that ask whether a smaller rule family certifies labels beyond lookup. For affine linear rules mod-k, we enumerate induced maps $f(w) = a^\top w + b \pmod{k}$ over the episode's effective window width and retain those consistent with the support windows; this is exact for multi-step linear episodes because the composed local map remains affine over the wider context. For native totalistic, outer-totalistic, outer-inner-totalistic, and threshold families, we export exact one-step posteriors over the corresponding feature maps. Multi-step native episodes are marked rollout-unsupported for this structured diagnostic because composition need not preserve the same one-step family. Empty or misspecified posteriors are treated as diagnostic mismatch rather than oracle evidence.

An audit of the reported splits found no true support-conflict cells and no core support-inconsistent cells. The nonzero support-inconsistent counts in the artifact are boundary/-context positions introduced by cropped spans with context padding: 351/11,099 query cells on TEST INTERPOLATION and 353/10,918 on TEST EXTRAPOLATION, all at boundaries. We therefore keep these cells visible in diagnostics while excluding them from claims about deterministic CA inconsistency.

A.3. Splits and Metadata

CELLARC includes 95k training episodes plus two 1k evaluation splits designed to probe different generalization capabilities. The TEST INTERPOLATION (interpolation) split mirrors the training distribution, testing whether models can maintain performance on unseen instances from the same difficulty regime. In contrast, the TEST EXTRAPOLATION (extrapolation) split deliberately stresses out-of-distribution generalization by selecting the most challenging episodes: those with lowest query-weighted coverage cov, highest Langton's λ, and highest cell entropy H. This

Split	Count
Train	95,317
Validation	1,000
Test Interpolation	1,000
Test Extrapolation	1,000
Test Interpolation 100	100
Test Extrapolation 100	100

Table 6. Dataset splits for the CELLARC-100k benchmark.

dual-split design isolates two distinct forms of generalization—performance under distributional similarity versus robustness to systematically harder, lower-supervision cases.

In the released dataset, the extrapolation split is 100% chaotic (by λ bin) with median statistics $\mathrm{cov} \approx 0.50$, $\lambda \approx 0.888$, and $H \approx 2.26$, while the interpolation split closely matches the training distribution across all complexity proxies. Across splits, the mean flattened episode length is $\sim$142 tokens (max 252), consistent with the ≤ 256-token budget.

The artifact materializes two diagnostic layers. First, it includes deterministic conditioned slices from the released metadata: lookup-solvable, ambiguous-under-unrestricted-map, linear-identifiable-not-lookup, native-identifiable-not-lookup, coverage-bin, rollout-hard, and family-specific subsets. Second, it includes a fresh generated CELLARC-ED suite with fixed generator profiles for ambiguous, lookup-solvable, linear-identifiable-not-lookup, native-identifiable-not-lookup, rollout/composition-hard, and family-balanced OOD conditions. The generated local release contains 300 episodes per condition, and a regenerated 120-episode smoke suite is retained for stability checks.

The distribution of CA rule families varies substantially across splits (see Figure 2). Training data exhibits balanced family composition with Random ($\sim$25.5%), Totalistic ($\sim$24.1%), and several other families well-represented. The interpolation split maintains similar proportions, whereas the extrapolation split concentrates in Totalistic (86.8%) and Linear mod(k) (9.4%) families due to the highest-λ/highest-entropy filtering. Consequently, extrapolation scores primarily reflect generalization to these two families in the chaotic regime. Figure 3 shows the corresponding distribution of dynamical regimes, confirming that TEST EXTRAPOLATION eliminates ordered and edge-of-chaos episodes to focus entirely on chaotic CA. For detailed split-wise statistics, Figure 4 presents violin plots of λ, H, and cov, visually confirming that TEST INTERPOLATION matches TRAIN/VALIDATION while TEST EXTRAPOLATION skews toward the intended difficulty regime.

100-episode subsets for LLM evaluation. To enable cost-controlled evaluation of large closed models, we provide

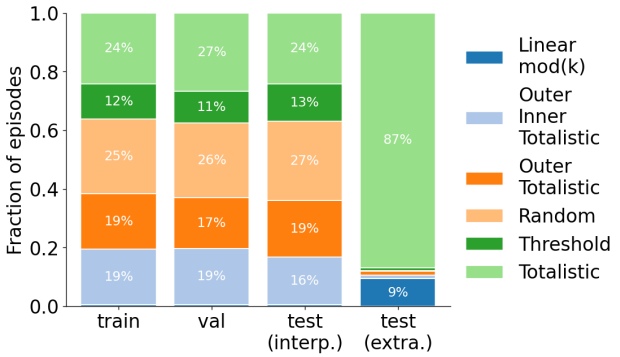

Figure 2. Distribution of CA rule families across training, validation, and test splits. The dataset exhibits diverse family composition (train: Random $\sim$25.5%, Totalistic $\sim$24.1%, Outer Inner Totalistic $\sim$18.9%, Outer Totalistic $\sim$18.9%, Threshold $\sim$12.1%, Linear mod(k) $\sim$0.6%). Notably, the extrapolation split is concentrated in Totalistic (86.8%) and Linear mod(k) (9.4%) families due to the highest-λ/highest-entropy filtering (see text).

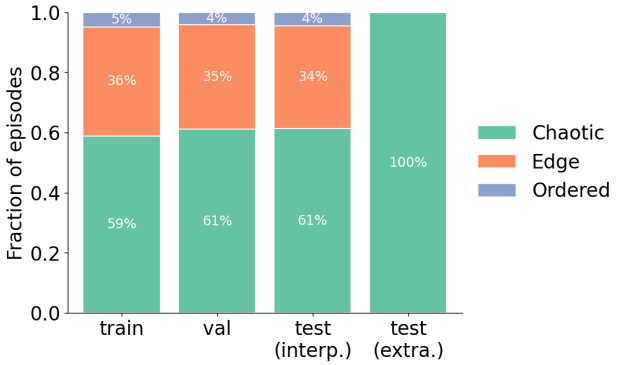

Figure 3. Distribution of Langton's λ regimes across splits. The dataset is dominated by Chaotic ($\lambda > 0.5$; 59.4%) and Edge-of-chaos ($0.4 < \lambda \leq 0.5$; 35.8%) rules, with a small fraction of Ordered ($\lambda \leq 0.4$; 4.8%) dynamics. The extrapolation split now consists entirely of chaotic episodes (100%), removing edge-of-chaos and ordered rules to stress generalization to the highest-entropy CA.

fixed 100-episode diagnostic subsets from each test split (TEST INTERPOLATION 100 and TEST EXTRAPOLATION 100). These subsets are useful for prompt iteration and cached closed-model comparisons, but they have higher variance and should not be read as stable family-level estimates or replacements for the full 1k splits.

Saturation Protection To confirm a score, avoid potential saturation, memorization or leakage risk in external APIs, researchers can always generate additional episodes of matching difficulty using the artifact generator.

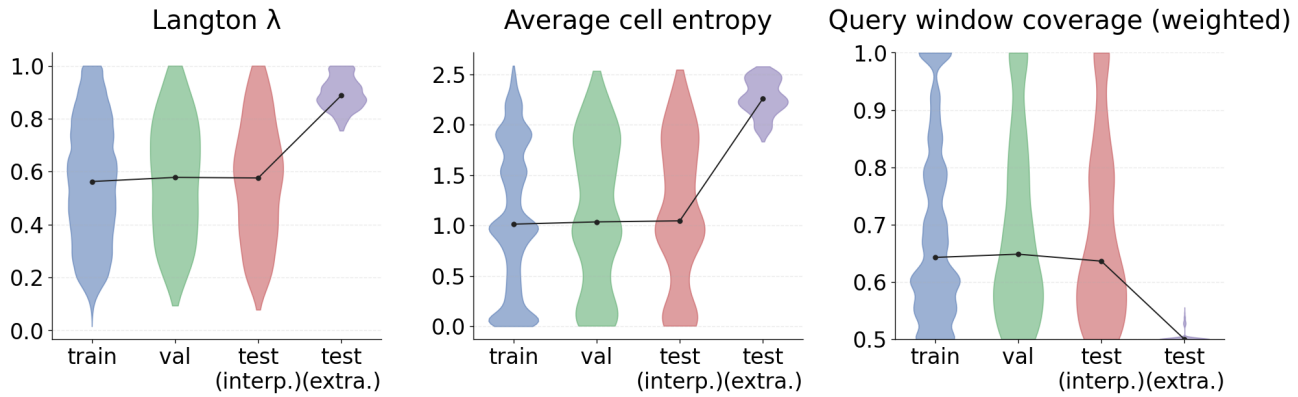

Figure 4. Distribution of Langton's λ, mean cell entropy H, and query-weighted coverage cov across splits. Each panel shows a violin plot for TRAIN, VALIDATION, TEST INTERPOLATION, and TEST EXTRAPOLATION. TEST INTERPOLATION closely matches TRAIN/VALIDATION across all three statistics, while the TEST EXTRAPOLATION split is deliberately shifted toward higher λ, higher H, and lower cov.

B. Methods and Baselines

We implemented symbolic, LLM, convolutional, recursive reasoning, and neural CA baselines within a single training/evaluation harness. Unless stated otherwise, neural models emit per-token logits over the output palette and are trained with cross-entropy. Preset definitions and parameter counts for every architecture live in Appendix J. All models are instantiated in small, medium, and large variants (approx. 100k, 1M, and 10M parameters). Because methods differ in what information they can access at evaluation time, we explicitly distinguish baseline, inductive, support-label/query-input, transductive, and oracle regimes.

B.1. Episode Representation and Training Interfaces

Canonical episode. Each episode is a structured record (flattening is optional and only used for sequence-only training):

```
{
  "id": "...",
  "train": [
    { "input": [0, 1, ...],
      "output": [2, 0, ...] },
    { "input": [5, 4, ...],
      "output": [3, 3, ...] },
    ...
  ],
  "query": [3, 0, ...],
  "solution": [4, 5, ...]
}
```

Unless stated otherwise, symbols lie in $\{0, 1, \ldots, k-1\}$ (typically $k \in [2, 6]$). Sequence lengths are variable. The supervision budget of ≤ 256 counts integer symbols across all fields of an episode; metadata keys and fixed markers are not counted. We expose the episode to the model in one of two ways that align with our regimes.

(i) Single I/O pair (Task Embeddings; TE). Training proceeds on individual labeled support pairs (x, y) from registered episodes. All support pairs from the same episode share a learned *task embedding* e_{ep} that is injected into the model input at every time step as a broadcast bias (for sequence models, we optionally realize e_{ep} as a single learned prefix token; this prefix is not part of the dataset or the supervision budget). In the audited implementation, validation/test episode identifiers are registered before model construction and the embedding-training iterable can include their support rows; query labels are used for evaluation loss/-metrics, not for default gradient updates. We therefore label current TE scores as *transductive TE diagnostics*, not as directly comparable inductive few-shot inference.

(ii) Flattened episode (In-Context Learning; ICL). Here the full episode is serialized into a single token stream and fed in one shot; no task embeddings are used. The vocabulary is $\{0, \ldots, k-1\}$ plus six markers `I`, `O`, `Q`, `T`, `M`, `E`. Whitespace separates tokens and, for $k \leq 6$, each integer is a single token. We serialize five supports followed by the query:

$$(\texttt{I}\ x_1\ \texttt{O}\ y_1),\ \ldots,\ (\texttt{I}\ x_5\ \texttt{O}\ y_5),\ \texttt{Q}\ x^\star,\ \texttt{T},\ \underbrace{\texttt{M}, \ldots, \texttt{M}}_{|y^\star|},\ \texttt{E}.$$

During training, loss is applied only on the masked query positions after replacing `M` with the ground-truth symbols (teacher forcing). At inference, the model replaces `M` by its predictions until $|y^\star|$ symbols are produced (optionally followed by `E`). This interface naturally handles unseen tasks without additional training, at the cost of higher sequence length and memory.

Method	Sup. labels	Query in	Test opt.	Ep. ID	Regime
Random/Copycat	No	No	No	No	Baseline
Most-frequent	Yes	No	No	No	Support-only statistic
de Bruijn	Yes	Yes	No	No	Support lookup
Posterior selector	Yes	Yes	No	No	Support labels + query input
Proposal+Verifier	Yes	Yes	No	No	Support labels + query input
Learned proposer+verifier	Yes	Yes	No	No	Learned support proposal
Transformer-ICL	Yes	Yes	No	No	Inductive ICL
Current TE	Yes	Yes	Yes	Yes	Transductive TE
Oracle selector	Yes	Yes	No	Uses scores	Oracle

Table 7. Information-access regimes. "Unlabeled query input" means the method may inspect the query sequence but never query labels. "Test-time opt." means optimizing method-specific state for held-out episodes; current task embeddings optimize per-episode vectors from held-out support rows and are therefore diagnostic rather than a clean inductive baseline.

B.2. Symbolic Baselines

- **Copycat** echoes the query input as output.
- **Most-frequent** predicts uniform sequence of digits consisting of the most common color observed in the episode.
- **Random** samples each output color uniformly based on the episode's observed vocabulary.
- **De Bruijn local map** learns the induced local rule $F^t : [k]^C \to [k]$ from the five supports by counting centered $C{=}2rt{+}1$ query windows; at test time it applies majority back-off for unseen windows (de Bruijn & Erdős, 1948; Sutner, 1991).
- **Posterior selector** keeps exact de Bruijn lookup on seen query windows, and for unseen windows selects the most confident support-consistent posterior mode among affine linear, totalistic, outer-totalistic, outer-inner-totalistic, and threshold hypotheses. It never reads query labels.
- **Proposal+Verifier** fits the same family proposals from support only, chooses a global support-consistent family by query-side posterior concentration with a small description-length penalty, preserves exact lookup on seen windows, and falls back to the posterior selector otherwise.
- **Learned family proposer+verifier** trains a small support-only ExtraTrees classifier over support-derived and support-posterior features to propose one compact family label, then routes that proposal through the same exact verifier. Accepted rows are support-consistent by construction; rejected rows are reported separately, with a de Bruijn fallback only in a clearly labeled hybrid ablation.

Oracle Ensemble (row-wise max). To probe complementarity we report an *Oracle Ensemble* score that, for each episode, selects the higher per-episode accuracy between the Transformer-TE and de Bruijn solvers. This is a non-realizable upper bound (it assumes a perfect selector) but it quantifies headroom that a learned or metadata-driven selector could recover.

B.3. Transformer (vanilla)

Encoder-only Transformer (Vaswani et al., 2017): one-hot token inputs, sin/cos positions, ReLU feed-forward, attention via PyTorch SDPA; outputs per-token logits (no teacher forcing in reported numbers).

B.4. LSTM (RNN)

LSTM baseline using an encoder-only variant without teacher forcing. Task embeddings, when enabled, are added as an input-space bias to all time steps; they are disabled in ICL runs. Our RNN setup is guided by prior evaluations across the Chomsky hierarchy (Delétang et al., 2023).

B.5. 1D CNN

The 1D CNN embeds tokens and applies a stack of dilated Conv1d+GELU blocks with padding to preserve length and optional residuals, then projects to logits. Task embeddings (if enabled) are added as a per-time-step bias after the token embedding.

B.6. Neural Cellular Automata (NCA)

Our 1D NCA embeds tokens, maps to an internal channel state, iterates depth-wise convolutions (groups=channels) to produce a multi-kernel perception tensor, and updates via an MLP. We apply cell dropout during updates and keep sequence length fixed (odd kernel sizes). Implementation is based on recent NCA literature.(Faldor & Cully, 2025)

B.7. Recursive-reasoning models with ACT

TRM and a Transformer-ACT ablation follow Jolicoeur-Martineau (2025), while HRM follows Wang et al. (2025). Shared features: Rotary position embeddings (RoPE) (Su et al., 2024), RMSNorm (Zhang & Sennrich, 2019), SwiGLU MLP (Shazeer, 2020), and an Adaptive Computation Time (ACT) outer loop (Graves, 2016; Dehghani et al., 2019) with max 16 steps (exploration prob. 0.1 during training), plus a Q-head for halting. The ACT loss uses a stabilized cross-entropy ("stablemax") for the language-modeling term. At inference: HRM/TRM run at max steps for batch uniformity; Transformer-ACT evaluates with ACT

disabled unless explicitly enabled.

Note on task embeddings in recursive models. Unlike the vanilla Transformer/CNN/LSTM, recursive models use task embeddings as prefix tokens: a per-episode vector is sliced into L_p=1 token and concatenated before the sequence.

B.8. LLM Baselines

We evaluate closed LLMs via in-context prompting (no gradient updates) (Brown et al., 2020). The model reported in Table 5 as *GPT-5 High* is `gpt-5-2025-08-07` with reasoning effort set to `high` using the OpenAI Responses API. This row is a dated diagnostic subset, not a full reproducibility claim. The cached prediction log was produced on 2025-11-09 UTC and the per-task accuracy exports on 2025-11-10 UTC. Configuration follows the public ARC-AGI benchmarking presets used by our script: Responses API, streaming enabled, `max_output_tokens=200000`, `request_timeout=3600`, `max_retries=2`, retry backoff 2.0, synchronous worker pool 10, no JSON mode, provider-default temperature/top-p, and no explicit chain-of-thought request. We accept free-form answers and extract integer sequences with a regex parser that chooses an exact-length sequence when available and otherwise clips to the expected length; all 199 cached scored records parse to the expected length, and one interpolation request has no cached response and is omitted rather than imputed. Cached prompts and responses live under `outputs/llm/gpt5/` in the baseline-output release path; the compact release artifact carries summaries and reproduction instructions rather than the full closed-model log. An exact prompt instance is shown in Appendix I.

B.9. Training Protocol

We use AdamW for non-recursive models and a dedicated recursive trainer (same batch size when memory allows; ACT-specific learning-rate settings including a higher LR for task embeddings). Evaluation uses per-token accuracy on the query; the max sequence length is 272 to accommodate markup/prefix tokens while all serialized episodes remain within the $\leq$256-token supervision budget. Most of the models were trained either on a single A100 GPU or in distributed regime (4x H100 GPUs or 3x A100). All models are trained for 50 epochs (one epoch covers the whole training set) with batch size 768. Batch size of 192 with 4x gradient accumulation was used for large variants of recursive models.

C. Experimental Setup

Optimizers and schedules. Non-recursive baselines (Transformer, CNN, RNN, NCA) use AdamW with optional linear warmup and otherwise constant learning rate (no cosine decay by default). Recursive models (TRM/HRM/Transformer-ACT) follow the TinyRecursiveModels training recipe (Jolicoeur-Martineau, 2025): we use an AdamATan2 optimizer with a cosine schedule and linear warmup for the embedding regime, and a fixed learning rate in the in-context regime. For task embeddings with recursive models, we apply a separate optimizer with a higher learning rate to the prefix-embedding table. We do not use weight decay unless specified in configs.

Losses and clipping. Sequence models optimize standard cross-entropy over output tokens. Recursive models optimize a stabilized "stablemax" cross-entropy for language modeling and binary cross-entropies for the halting heads (halt/continue); exact-match, per-token accuracy, and Q-halt accuracy are logged. Gradient clipping is disabled by default (configs set `clip_grad_norm: null`).

Budgets and presets. We provide small/medium/large neural presets (see Appendix J for the complete specification). Table 5 reports the main summary across symbolic, neural, and closed-model representatives; Appendix tables report medium ($\sim$1M) and small ($\sim$0.1M) neural sweeps. Batch size is 768 unless constrained by memory. Episodes are serialized once so the full context (markup, supports, query) contains at most 256 data-symbol tokens; the sequence-length cap is 272 to allow fixed markers and prefix tokens.

Compute and result ledger. The paper-facing ledger is split by access regime. Symbolic rows are deterministic CPU passes over the released splits with 200-sample episode bootstrap intervals and no trainable weights. The clean ICL rows were rerun for seeds 1337, 2027, and 2028 on an NVIDIA H100 NVL; manifest logs record mean training time of about 72 minutes per Transformer seed and 30 minutes per CNN seed, with prediction export taking about 4 seconds per seed. Archived TE and other neural rows are single point estimates from A100/H100-class 50-epoch runs and are labeled diagnostics. Closed-model rows use the fixed GPT-5 cache described above. Generated diagnostic suites use fixed generator seeds, 300 episodes per condition, and local exact verifiers. The artifact mirrors these paths in `SOLVER_RESULTS.md` and `REPRODUCE.md`.

Metrics. We report per-token accuracy on the query sequence, exact-match accuracy, and adaptive-computation steps (if applicable). Symbolic methods lack trainable weights, so we also log coverage statistics to interpret their performance.

Evaluation splits. Interpolation mirrors training; extrapolation enforces lower query-weighted coverage (mean 0.50

vs. 0.68), lower observed support coverage (mean 0.22 vs. 0.29), and higher λ/H to stress generalization beyond observed local patterns. We additionally publish metadata for custom slices (e.g., high-k or high-entropy subsets).

D. Results

Overall. The strongest full-split symbolic rows are posterior selection and the support-only Proposal+Verifier, reaching 88.0/88.2% token accuracy on TEST INTERPOLATION/TEST EXTRAPOLATION for posterior selection and 87.9/88.2% for Proposal+Verifier. Exact-query accuracy improves to 38.0/31.8% for posterior selection and 37.9/31.8% for Proposal+Verifier, versus 17.1/0.8% for de Bruijn lookup with majority backoff. The gain comes from aligning symbolic learners to centered support windows and applying structured support-consistent families on uncovered query windows; core support-consistency errors are zero in the corrected runs. Boundary/context cells are reported separately and excluded from deterministic-conflict claims. This still does not solve CELLARC: exact-query accuracy remains below 40%, and the strongest exact solvers rely on hand-coded family posteriors. In three-seed clean ICL sweeps, Transformer reaches 51.0 ± 0.1/ 28.8 ± 0.3% token accuracy on TEST INTERPOLATION/TEST EXTRAPOLATION, while CNN reaches 46.6 ± 0.3/ 19.1 ± 0.4%, far below the support-only structured solvers.

Neural architectures. The matched three-seed ICL sweep supports a cautious architecture comparison between the two clean rows: the vanilla Transformer is consistently above CNN on both splits, by about 4.4 token points on interpolation and 9.7 points on extrapolation. Seed variation is small relative to this gap (Transformer extrapolation std. 0.31 points; CNN extrapolation std. 0.42 points). Other neural architectures and TE rows remain context diagnostics rather than seed-robust rankings. In archived TE point estimates, additive task embeddings raise some vanilla architectures (e.g., Transformer+TE: 58.0/32.4; CNN+TE: 52.7/29.0; LSTM+TE: 51.0/27.9), while recursive reasoning models remain competitive but below the vanilla Transformer in those archived runs (e.g., HRM+TE: 50.8/28.2; TRM+TE: 48.7/26.4; Transformer-ACT+TE: 50.8/27.2). Results at medium and small scales follow the same pattern (see Table 14, Table 15).

NCA baseline. Neural cellular automata (NCA) in medium/large presets ran out of memory when unrolled, and the small preset plateaued below 25% token accuracy. NCA excel at forward simulation of a fixed rule; here the task is inverse rule induction over varying rules (see Appendix K; Appendix Figures 9–10).

Task embeddings vs. in-context (ICL). The clean ICL rows are now seed-swept, but the TE rows remain transductive diagnostics. In archived full-split point estimates, TE often raises vanilla Transformer/CNN/LSTM: Transformer (large) moves from the seed-swept ICL mean 51.0/28.8 to a TE point estimate of 58.0/32.4; CNN moves from 46.6/19.1 to 52.7/29.0. This gain should be interpreted as the value of per-episode support adaptation, not as evidence that the model can infer an unseen task from context alone. For recursive models, TE is a single prefix token and slightly hurts generalization: TRM (large) improves from 48.7/26.4 (TE) to 50.9/29.0 (ICL); HRM (large) from 50.8/28.2 to 50.9/30.1; Transformer-ACT (large) from 50.8/27.2 to 51.0/30.3. Appendix Figures 9–10 visualize train/validation dynamics across sizes and regimes.

Symbolic vs. neural baselines The de Bruijn local map is a strong non-learned baseline on CELLARC. It memorizes the induced local map where centered support windows cover the query, so its accuracy rises steeply with support coverage (see §A.1) and reaches 100% on lookup-identifiable cells. It cannot extrapolate beyond exact lookup: on full TEST EXTRAPOLATION linear-identifiable-but-not-lookup cells it reaches 18.8%, and on native-identifiable-but-not-lookup cells it reaches 29.7%. Posterior selection and Proposal+Verifier close both certifiable strata at 100% by using structured support-consistent hypotheses. These exact rows are hand-coded support-only baselines. The learned family proposer in Table 9 is a first amortized diagnostic: it learns to propose a family from support-derived features, then relies on the same verifier to reject inconsistent proposals. The comparatively high score of *Most-frequent* (50.4/28.2) indicates non-uniform output palettes and quiescent tendencies in several rule families; it remains a sanity check for genuine rule induction.

Learned support-only proposer. The learned proposer closes the generated affine diagnostic (100% family accuracy and 100% accepted exact-query accuracy), reaches 75% family accuracy and 76% accepted exact-query accuracy on generated native-identifiable episodes, and remains support-consistent after verification (0% support-consistency error on accepted rows). On the full released extrapolation split it accepts 99.8% of episodes and obtains 88.2% accepted token accuracy, while hitting 100% token accuracy on the released linear- and native-identifiable non-lookup strata. This does not replace exact posterior enumeration: the features include support-posterior summaries, and hard rollout-composition episodes reveal the limitation, with only 21.7% acceptance. It does, however, show that a compact learned proposal layer can recover useful parts of the structured family signal under an exact verifier.

Closed LLMs vs. neural baselines On the prediction-log subset, the closed model (*GPT-5 High*) attains 62.3% interpolation and 48.1% extrapolation, exceeding Transformer-TE on extrapolation but trailing the corrected symbolic solvers in aggregate token accuracy. This is a dated subset diagnostic, not a full-split or reproducible leaderboard claim. The more informative comparison is identifiability-conditioned: on extrapolation-100 cells that are linearly identifiable but not lookup-identifiable, GPT-5 reaches 59.0%, Transformer-TE reaches 11.5%, de Bruijn reaches 13.1%, and the structured support-only symbolic rows reach 100% (Table 8). The intended diagnostic is therefore sharper: pure lookup and compact neural models fail on certifiable cells, GPT-5 partially recovers them in this cached run, exact hand-coded structured posteriors close them, and the learned proposer+verifier begins to amortize this family choice while remaining verifier-gated.

Support budget. Table 10 and Figure 6 evaluate de Bruijn lookup under progressively larger support prefixes on matched generated episodes. Each row for $K{=}1, 2, 3, 5, 8, 10$ uses the first K supports from the same underlying 10-support episodes, so the curve measures additional evidence rather than a distribution change. Token accuracy, exact-query accuracy, lookup-identifiable coverage, and Bayes token accuracy all rise with K, but exact-query accuracy remains only 50.8% on matched interpolation and 50.0% on matched extrapolation at $K{=}10$.

Oracle Ensemble. On the same prediction-log subset, an Oracle Ensemble that selects per episode between Transformer-TE and de Bruijn attains 79.0% on interpolation and 61.1% on extrapolation. Because it assumes a perfect selector this is an upper bound, yet it highlights complementary failure modes: de Bruijn dominates covered cells, while neural/LLM methods can sometimes recover structure on cells outside lookup coverage.

Small vs. medium vs. large models Capacity helps, but with diminishing returns. Comparing Transformer presets, moving from $\sim$0.1M to $\sim$1M parameters yields modest gains, while $\sim$10M parameters delivers the clearest jump in the archived full-split TE sweep (50.8/29.6 at small, 51.5/28.8 at medium, and 58.0/32.4 at large; see Table 15, Table 14). The interpolation–extrapolation gap Δ slightly widens with scale for Transformer-TE ($21.2 \rightarrow 22.6 \rightarrow 25.6$), indicating that extra capacity primarily boosts fit on the training-like distribution while yielding smaller relative gains on out-of-distribution, low-coverage cases. CNNs benefit disproportionately from TE at all scales (ICL $\ll$ TE), while LSTMs track Transformers but remain several points behind. Recursive models (TRM/HRM/Transformer-ACT) scale stably in ICL but tend to overfit with TE, yielding smaller improvements from small$\rightarrow$medium$\rightarrow$large than the vanilla Transformer.

Overfitting Appendix Figure 10 reveals that recursive models (TRM/HRM/Transformer-ACT) steadily drive training loss down while validation loss bottoms out early and rises, mirroring the divergence between training and validation accuracy in Figure 9. Together these curves show pronounced overfitting despite similar asymptotic training accuracy.

Takeaways. Among the matched clean ICL rows, the vanilla Transformer outperforms the CNN but remains far below structured support-only solvers. Task embeddings can help vanilla architectures as transductive diagnostics, but can become shortcut tokens in recursive models, where an ICL regime leads to better generalization. Qualitative grids in Figure 8 contextualize these behaviors on concrete episodes covering easy through unsolved cases.

E. Discussion

What CELLARC actually measures. CELLARC is best viewed as a controlled test of *rule discovery efficiency under partial local information*. Each episode compresses to ≤ 256 tokens, the hypothesis class is finite and explicitly local (§2), and the primary source of difficulty is how much of the query's centered windows are *covered* by supports (§A.1). This isolates a clean question: given five supports that reveal only a subset of length-C neighborhoods, can a model infer the hidden local rule F well enough to label the query?

Drivers of difficulty. Across symbolic and neural methods, per-token accuracy rises with query-weighted coverage cov and declines as dynamics become more active/entropic (higher Langton's λ and H). These effects persist within families and across sizes (see Table 13, Figure 11). Intuitively, higher cov shrinks the unobserved slice of the local map and makes majority/back-off strategies viable, whereas high activity/entropy increases the diversity of neighborhoods that matter, amplifying the penalty for unseen windows. This supports our design choice to define TEST EXTRAPOLATION by *jointly* lowering cov and raising λ/H.

Inductive biases that help. In seed-swept clean ICL runs, a vanilla encoder-only Transformer is the strongest compact inductive architecture among the two matched rows in Table 5, outperforming CNN on both interpolation and extrapolation with low seed variance. In the transductive TE setting, additive TE supplies a low-cost per-episode state and raises scores for some architectures, but those rows remain diagnostic rather than clean inductive baselines. Recursive models with ACT (TRM/HRM/Transformer-ACT) provide extra compute but no clear advantage on strictly local CA dynamics; when TE is injected as a prefix token,

Model	Split	Stratum	Cells	Acc.	95% CI	Gap
GPT-5 High	extrapolation-100	linear-ident. not lookup	61	59.0	[31.6, 83.6]	41.0
Transformer TE	extrapolation-100	linear-ident. not lookup	61	11.5	[3.6, 21.1]	88.5
Posterior Selector	extrapolation-100	linear-ident. not lookup	61	100.0	[100.0, 100.0]	0.0
Proposal+Verifier	extrapolation-100	linear-ident. not lookup	61	100.0	[100.0, 100.0]	0.0
de Bruijn	extrapolation-100	linear-ident. not lookup	61	13.1	[7.8, 19.3]	86.9
Posterior Selector	extrapolation	linear-ident. not lookup	516	100.0	[100.0, 100.0]	0.0
Proposal+Verifier	extrapolation	linear-ident. not lookup	516	100.0	[100.0, 100.0]	0.0
de Bruijn	extrapolation	linear-ident. not lookup	516	18.8	[15.3, 22.6]	81.2
Posterior Selector	extrapolation-100	native-ident. not lookup	368	100.0	[100.0, 100.0]	0.0
Proposal+Verifier	extrapolation-100	native-ident. not lookup	368	100.0	[100.0, 100.0]	0.0
de Bruijn	extrapolation-100	native-ident. not lookup	368	31.8	[26.7, 37.3]	68.2
Posterior Selector	extrapolation	native-ident. not lookup	3826	100.0	[100.0, 100.0]	0.0
Proposal+Verifier	extrapolation	native-ident. not lookup	3826	100.0	[100.0, 100.0]	0.0
de Bruijn	extrapolation	native-ident. not lookup	3826	29.7	[28.0, 31.5]	70.3

Table 8. Oracle-gap slices from the Phase-2 diagnostic export. "Gap" is the difference between the certifying oracle accuracy (100%) and model accuracy on cells identified by the named stratum. The headline slice is linearly identifiable but not lookup-identifiable cells, where the affine mod-k posterior certifies the answer even though de Bruijn lookup has not seen the exact query window. Native-identifiable rows use the exact one-step structured posterior for the recorded native family.

Evaluation condition	Episodes	Family acc.	Accept	Reject	Accepted token	Accepted exact	Stratum token
Generated linear-ident. not lookup	300	100.0	100.0	0.0	100.0	100.0	100.0
Generated native-ident. not lookup	300	75.0	100.0	0.0	94.3	76.0	99.8
Generated family-balanced OOD	300	54.0	85.3	14.7	93.5	75.8	93.5
Generated rollout-composition hard	300	38.3	21.7	78.3	100.0	100.0	100.0
Released extrapolation	1000	98.8	99.8	0.2	88.2	31.9	88.2
Released extrapolation, linear stratum	1000	98.8	99.8	0.2	88.2	31.9	100.0
Released extrapolation, native stratum	1000	98.8	99.8	0.2	88.2	31.9	100.0

Table 9. Learned support-only family proposer diagnostic. All values except episode counts are percentages. The proposer predicts a compact family label from support-derived features; accepted predictions pass an exact support verifier, so accepted support-consistency error is zero. "Stratum token" restricts scoring to the named diagnostic stratum where applicable; the released-split stratum rows keep Episodes as the parent split size while scoring only cells in the named stratum. Hybrid de Bruijn fallback rows are omitted here and kept as artifact ablations.

they show shortcut overfitting signals (Figure 9, Figure 10).

Neural–symbolic complementarity. The de Bruijn local map is a strong no-training baseline whose accuracy closely tracks cov. Where supports nearly tile the query, symbolic lookup is hard to beat; where coverage is low, lookup has no way to extrapolate. The structured posterior rows show that some unseen-window labels are still certifiable from support alone under smaller rule classes, closing affine and native one-step oracle strata that lookup misses. The dated GPT-5 diagnostic recovers many more affine cells than Transformer-TE and de Bruijn on the extrapolation subset, but it remains below the exact structured posterior. This turns the benchmark from a score table into a diagnostic: a model should not be blamed for ambiguity under the unrestricted map, but failures on lookup- or rule-identifiable cells are genuine inference failures.

Closed LLMs and amortized search. On fixed 100-episode diagnostic subsets, the dated GPT-5 High diagnostic surpasses trained compact models especially on the extrapolation subset and on low-coverage/high-entropy episodes. Because this row is subset-only, API-dependent, and much more expensive than the open baselines, we treat it as suggestive evidence that broad pretraining can amortize some *search over rule families*, not as a stable leaderboard result. The learned proposer+verifier provides a smaller open diagnostic in the same direction: it amortizes support-only family proposals but still depends on exact verification and support-posterior features. Matching this closed-model diag-

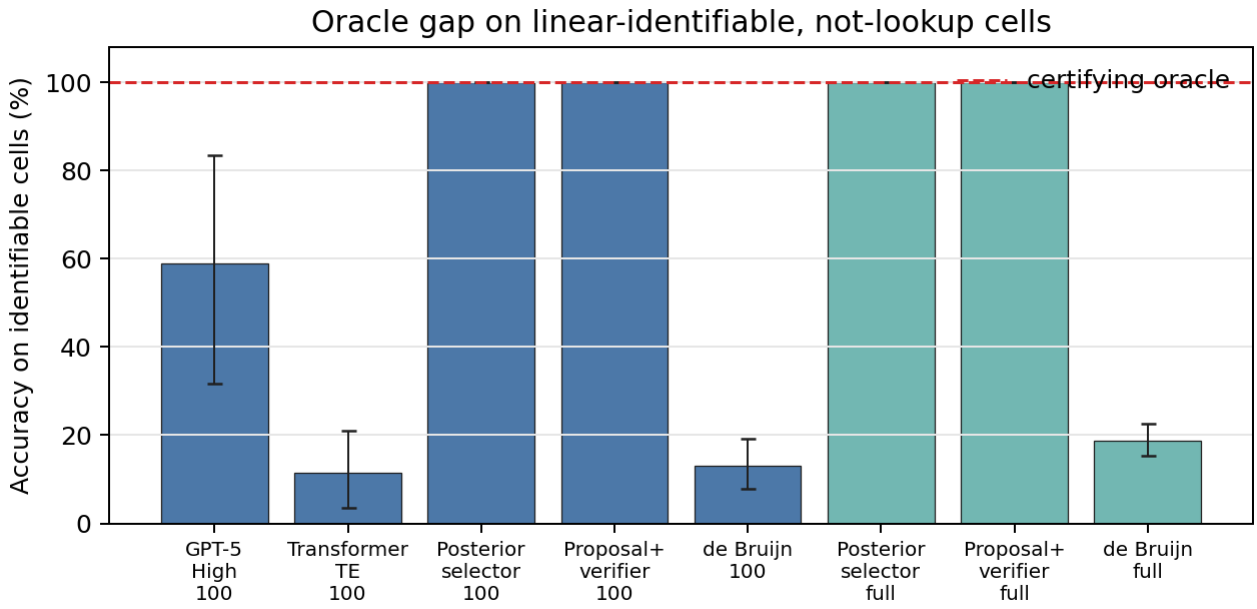

Figure 5. Accuracy on oracle-defined extrapolation-100 strata. The largest gap appears on linear-identifiable-but-not-lookup cells for compact Transformer-TE and de Bruijn lookup; the structured support-only posterior rows close this affine certifying oracle.

Split	K	N	Token	Exact-query	Lookup	Bayes
matched interp.	1	120	60.3	24.2	25.5	53.1
matched interp.	2	120	66.7	30.8	42.9	63.2
matched interp.	3	120	71.2	32.5	49.1	66.9
matched interp.	5	120	75.8	37.5	59.4	72.4
matched interp.	8	120	81.7	44.2	73.2	81.3
matched interp.	10	120	85.6	50.8	77.7	83.6
matched extrap.	1	120	59.2	23.3	19.7	49.9
matched extrap.	2	120	64.3	27.5	35.7	58.8
matched extrap.	3	120	69.3	31.7	44.2	63.2
matched extrap.	5	120	75.8	36.7	56.6	70.0
matched extrap.	8	120	81.5	46.7	68.9	76.5
matched extrap.	10	120	83.3	50.0	74.3	79.5

Table 10. Matched-prefix support-budget curve for de Bruijn lookup. Every row uses the same generated 10-support episodes within its split and evaluates the first K supports, avoiding the previous unmatched released-vs-K10 comparison.

nostic with compact, fully open models remains a concrete challenge that CELLARC makes cheap to iterate on.

Relation to ARC. ARC-AGI tasks conflate heterogeneous skills. CELLARC complements ARC by removing confounds: supervision and scoring are exact, the local hypothesis class is explicit, and difficulty is tunable. In this sense, CELLARC is a microscope for one capability—*learning local rules from scant evidence*—that repeatedly appears inside harder reasoning pipelines.

Practical benchmarking advantage. Because episodes are short, labeled with transparent metadata, and easy to regenerate in any form and in unlimited amount, CELLARC supports *comparative* evaluation across many training regimes that are impractical on ARC-AGI (ICL vs. task embeddings, TE vs. ACT depth, symbolic vs. neural vs. mixtures, compute scaling at fixed token budgets). This enables rapid ablations and fairer efficiency studies while keeping results reproducible.

F. Limitations and Future Work

Scope of dynamics. Episodes derive from one-dimensional CA with periodic boundaries, palettes $k \in [2, 6]$, and short horizons chosen to respect a 256-token budget. This deliberately narrows the hypothesis space and under-represents long-range spatial composition and multi-scale structure common in 2D settings.

Distributional shape of splits. The released TEST EXTRAPOLATION is dominated by Totalistic and Linear mod-k

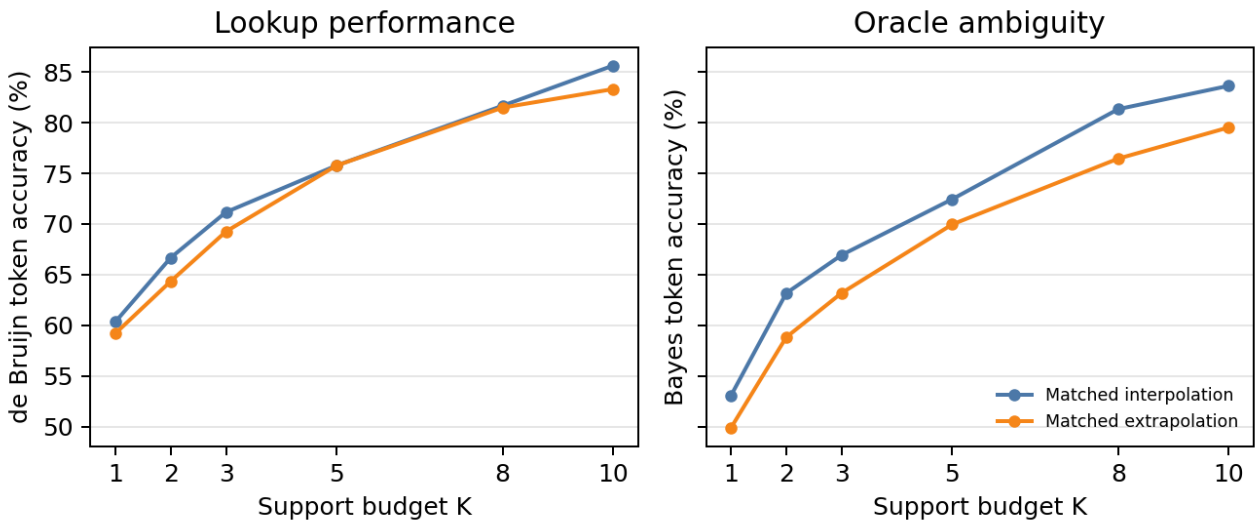

Figure 6. Matched support-budget diagnostic curve. Additional support improves token and exact-query accuracy and increases lookup-identifiable coverage, but exact-query accuracy remains far from solved at $K{=}10$.

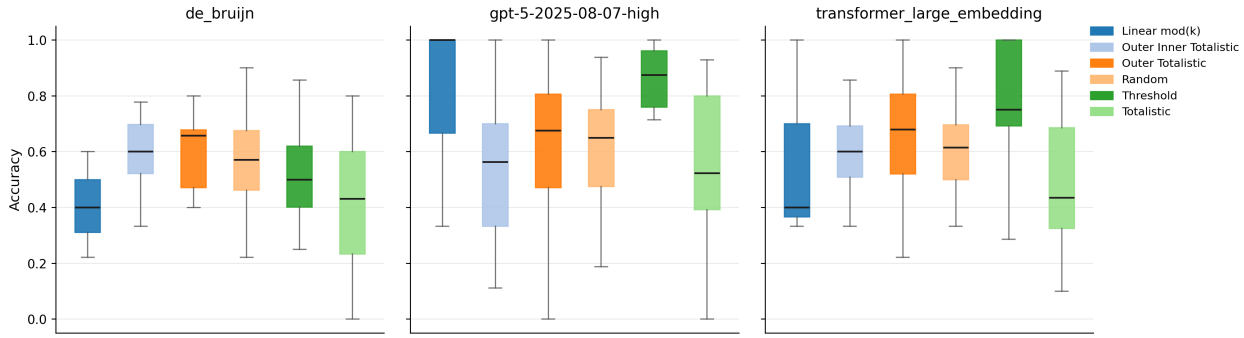

Figure 7. Per-family accuracy on the TEST INTERPOLATION 100 subset for three representative approaches (de Bruijn baseline, Transformer with task embeddings, and GPT-5 High). Box plots summarize per-task accuracies within each CA family. Improvements of GPT-5 over the symbolic baseline are largely driven by high accuracy on Linear mod-k and Threshold rules. The Transformer shows similar gains on Threshold but lags GPT-5 on Linear mod-k, suggesting useful inductive biases or inference strategies not captured by compact open models and motivating further analysis.

rules and is 100% chaotic by the Langton-λ bin. This sharpens the stress test but couples difficulty to a specific family mix. The generated CELLARC-ED suite now includes a 300-episode family-balanced OOD diagnostic, but it remains a diagnostic complement rather than a replacement for broader family-balanced benchmark releases.

Single-query supervision. Each episode contains one query. This is ideal for tight budgets and clean scoring, but it does not test continual inference within a task. Multi-query variants would probe longer-horizon credit assignment.

Metrics and objectives. Per-token accuracy is convenient but coarse. Exact-query, balanced, non-quiescent, support-consistency, and identifiability-conditioned metrics are now implemented in the artifact and used in the Phase-2 diagnostic tables. Further metrics—local-table recovery rate, equivariance tests under reflection/relabeling, and *program-consistency* scores computed from the induced local rule—would separate additional kinds of generalization.

Training and evaluation budgets. Our training and evaluation budgets were intentionally limited to emphasize efficiency and comparability. A more comprehensive hyperparameter sweep and longer training could strengthen several baselines, particularly the recursive models. The released benchmark episodes contain five support pairs; the artifact now includes a matched-prefix $K{=}10$ diagnostic set for support-budget analysis, but those rows are a generated

Figure 8. Qualitative grid of unrolled CELLARC episodes from test splits. The grid spans tasks of varying apparent regularity: note that more structured, low-entropy patterns tend to be easier across models. Notably, CA in cell A4 is an example where *all* models achieve 0% accuracy, while cells A1, B1, D1, B2, B3, and C3 have at least two models solving the episode exactly. This human-perceived regularity aligns with metadata trends: higher cov associates with higher accuracy, while higher λ and H tend to reduce accuracy (see Table 13).

Split	Family	Model	N	Token	Exact	Coverage
extrapolation-100	linear mod-k	GPT-5 High	11	61.3	9.1	50.0
extrapolation-100	linear mod-k	Transformer TE	11	16.1	0.0	50.0
extrapolation-100	linear mod-k	de Bruijn	11	51.6	0.0	50.0
extrapolation-100	linear mod-k	Posterior Selector	11	94.4	54.5	50.0
extrapolation-100	outer-inner totalistic	GPT-5 High	2	35.0	0.0	50.0
extrapolation-100	outer-inner totalistic	Transformer TE	2	35.0	0.0	50.0
extrapolation-100	outer-inner totalistic	de Bruijn	2	55.0	0.0	50.0
extrapolation-100	outer-inner totalistic	Posterior Selector	2	70.0	0.0	50.0
extrapolation-100	random	GPT-5 High	1	58.3	0.0	50.0
extrapolation-100	random	Transformer TE	1	16.7	0.0	50.0
extrapolation-100	random	de Bruijn	1	66.7	0.0	50.0
extrapolation-100	random	Posterior Selector	1	66.7	0.0	50.0
extrapolation-100	totalistic	GPT-5 High	86	46.4	1.2	50.4
extrapolation-100	totalistic	Transformer TE	86	29.5	0.0	50.4
extrapolation-100	totalistic	de Bruijn	86	62.3	1.2	50.4
extrapolation-100	totalistic	Posterior Selector	86	88.4	31.4	50.4
interpolation-100	linear mod-k	GPT-5 High	3	75.0	66.7	58.9
interpolation-100	linear mod-k	Transformer TE	3	50.0	33.3	58.9
interpolation-100	linear mod-k	de Bruijn	3	62.5	0.0	58.9
interpolation-100	linear mod-k	Posterior Selector	3	100.0	100.0	58.9
interpolation-100	outer-inner totalistic	GPT-5 High	20	55.0	15.0	72.8
interpolation-100	outer-inner totalistic	Transformer TE	20	57.5	5.0	72.8
interpolation-100	outer-inner totalistic	de Bruijn	21	78.6	19.0	71.7
interpolation-100	outer-inner totalistic	Posterior Selector	21	82.5	28.6	71.7
interpolation-100	outer totalistic	GPT-5 High	16	63.0	6.2	69.2
interpolation-100	outer totalistic	Transformer TE	16	63.0	6.2	69.2
interpolation-100	outer totalistic	de Bruijn	16	78.4	31.2	69.2
interpolation-100	outer totalistic	Posterior Selector	16	86.4	43.8	69.2
interpolation-100	random	GPT-5 High	27	61.8	0.0	66.9
interpolation-100	random	Transformer TE	27	59.9	0.0	66.9
interpolation-100	random	de Bruijn	27	80.1	7.4	66.9
interpolation-100	random	Posterior Selector	27	80.7	11.1	66.9
interpolation-100	threshold	GPT-5 High	15	82.6	26.7	68.2
interpolation-100	threshold	Transformer TE	15	81.2	33.3	68.2
interpolation-100	threshold	de Bruijn	15	81.2	20.0	68.2
interpolation-100	threshold	Posterior Selector	15	96.4	66.7	68.2
interpolation-100	totalistic	GPT-5 High	18	53.8	0.0	60.9
interpolation-100	totalistic	Transformer TE	18	48.4	0.0	60.9
interpolation-100	totalistic	de Bruijn	18	75.8	5.6	60.9
interpolation-100	totalistic	Posterior Selector	18	93.0	38.9	60.9

Table 11. Family-conditional subset metrics joined with coverage and exact-query accuracy. Counts differ slightly between GPT-5/Transformer and symbolic rows because one interpolation prediction record is absent from the closed-model/Transformer logs; tiny extrapolation family slices (N=1–11 outside Totalistic) are reported for transparency and should not be read as stable family-level estimates.

evidence-scaling suite rather than the primary five-support benchmark. We encourage readers to challenge these baselines and submit improved results to the public leaderboard.

Future work. We prioritize: (i) 2D multicolor CELLARC with matched metadata and coverage controls; (ii) noise robustness (token corruptions, occlusions) and boundary-condition diversity; (iii) multi-query and curriculum generators that sweep cov, k, and r over time.

Beyond CA: more expressive automata. To address expressiveness limits and cover a broader slice of the program-synthesis landscape, future benchmarks can instantiate tasks from richer machine models: *Mealy machines* (sequential transducers with output on transitions) for streaming I/O regular functions (Mealy, 1955); *subsequential (deterministic) transducers* for a large class of rational functions and practical string-to-string mappings (Choffrut, 2003; Mohri, 1997); and *Turing-complete* substrates (e.g., small universal machines, or universal CA such as Rule 110) to stress algorithm learning and memory use (Turing, 1936; Cook, 2004; Hopcroft et al., 2006). These families would probe capabilities absent from 1D CA—e.g., stateful streaming, alignment, and unbounded computation—and connect directly to neural program-induction work (NTM/DNC, Neural GPU, neural algorithmic reasoning) (Graves et al., 2014; 2016; Kaiser & Sutskever, 2016; Veličković & Blundell, 2021).

Meta-learning. Because episodes are short and structurally homogeneous, CELLARC is well-suited for gradient-based meta-learning and fast adaptation (MAML, FO-MAML, Reptile) (Finn et al., 2017; Nichol et al., 2018a;b). Episodic sampling can serve as tasks, enabling controlled studies of inner-loop step counts, adaptation horizons t, and how meta-initializations trade off with in-context strategies.

Dynamics and interactivity. CAs are inherently dynamical. This suggests extending the benchmark along a time/interaction axis: multi-step interactive episodes where models can query/perturb inputs or choose rollouts, con-

necting to emerging interactive ARC-style settings (e.g., the dynamic/minigame direction sometimes referred to as "ARC-AGI-3"). Such variants would let us test whether agents can *actively* acquire the local rule under a fixed observation budget.

Neural cellular automata baselines. Motivated by recent successes of neural CA in pattern formation and morphogenesis tasks (Faldor & Cully, 2025), we implemented one-dimensional NCA baselines analogous to the approach described in Appendix B. These models caused OOM errors in both large and medium parameter setting and failed to achieve competitive performance on the small-scale setting, obtaining per-token accuracies below 25% on the interpolation split. We hypothesize that the difficulty arises from two factors: (i) NCA architectures excel at forward simulation of *known* rules but struggle with the inverse problem of inferring latent rules from sparse supervision, and (ii) the episodic structure of CellARC—where each episode instantiates a distinct CA rule—conflicts with the NCA inductive bias toward learning a single universal update function. Both in-context learning variants and task-embedding augmentation failed to bridge this gap. We therefore omit NCA from the main results table but include this negative result to guide future architectural exploration.

Autoregressive and CoT model variants. In the repository with model baselines we have tried or implemented autoregressive variants for LSTM, Transformer and Recursive Models, but due to the added complexity, training cost and nonsignificant gains on small and medium model sizes, we have discontinued this path. CoT wasn't tried for similar reason. Authors of this benchmark do not exclude that more such baselines will be added in future to reflect trends in recent literature.(Wei et al., 2022)

G. Conclusion

Many reasoning pipelines require systems to acquire a new task from minimal supervision. CELLARC isolates a narrow, measurable version of that problem: *inferring local rules from compact, partial observations*. By grounding task generation in cellular automata rather than human intuition, we obtain a benchmark that is simultaneously controllable, generative, efficient, and transparent.

Concretely, CELLARC supplies unlimited episodes whose difficulty is *explicitly* governed by alphabet size, radius, rule family, Langton's λ, query-weighted coverage, and cell entropy. Episodes serialize to ≤ 256 tokens with five supports and a single query, enabling fast iteration and apples-to-apples comparisons across symbolic and neural approaches. This design reframes performance as *rule-discovery efficiency under partial local information*, with metadata that makes difficulty measurable rather than anecdotal.

Empirically, core-aligned structured posterior selection is the strongest full-split token-accuracy baseline, and the support-only Proposal+Verifier essentially ties it. This strength is also a warning: token accuracy alone can make partial lookup look like reasoning, while exact-query and identifiability-conditioned views reveal which cells are genuinely solved. On cells that are linearly or natively identifiable but not lookup-identifiable, de Bruijn and seed-swept compact ICL models remain far below the certifying oracle, the dated GPT-5 diagnostic closes part of the affine gap, and the exact structured support-only rows close the gap by construction. A learned family proposer plus exact verifier partially amortizes this structure: it closes generated affine episodes, achieves 100% token accuracy on released linear/-native identifiable strata after acceptance, and exposes hard rollout-composition cases through rejection. Capacity scaling helps, but the interpolation–extrapolation gap persists for compact neural runs; TE can aid vanilla Transformers while acting as a shortcut for recursive models, where in-context training generalizes better.

CELLARC is not a complete measure of "reasoning." Its hypothesis class is one-dimensional and local, the extrapolation split is intentionally skewed toward chaotic totalistic dynamics, and scoring is dominated by per-token accuracy under single-query supervision. These constraints are deliberate: they make the benchmark cheap, reproducible, and sharp enough to expose inductive biases and trade-offs. They also point to concrete next steps: 2D variants, multi-query or interactive settings to test active acquisition; richer automata (subsequence transducers, turing machines, universal CA) to probe stateful computation and nonlocal algorithms; noise and boundary-condition robustness; and metrics that directly assess recovery of the latent local rule (e.g., program-consistency and equivariance tests).

We provide public data hosting, code, diagnostics, Croissant/RAI metadata, and a checksummed local release package for rigorous, efficiency-centric studies–neural, symbolic, and neuro-symbolic. In our view, CELLARC complements ARC-AGI by serving as a microscope for one recurrent capability inside harder tasks: learning compact rules from scant evidence. We invite the community to use its controllable knobs and rich metadata to build better selectors, move beyond verifier-gated learned proposals toward fully learned open posteriors, and match closed-model diagnostic behavior under low coverage with fully open systems.

H. Additional Material

I. LLM Prompt Example

We evaluate closed LLMs with a concise few-shot prompt that presents five input–output training pairs followed by

a test input. The model is instructed to return only the final output sequence as space-separated integers. An exact prompt instance is listed below.

Listing 1. Example LLM prompt used for CELLARC evaluation. The model is instructed via few-shot pairs and asked to output only the final sequence.

```
User Prompt:
Predict the output sequence based on the
    provided examples.

Example 1:

Input:
3 4 1 3 5 1 4 2 1 4 3 1 4
Output:
1 0 1 1 0 1 0 2 1 0 1 1 4

Example 2:

Input:
2 2 5 2 3 3 2 3 4 2 3 5 2
Output:
2 2 0 2 1 1 2 1 4 2 1 5 2

Example 3:

Input:
5 3 1 5 4 1 5 5 2 2 2 3 2
Output:
0 1 1 5 4 1 5 5 2 0 2 1 2

Example 4:

Input:
4 4 5 4 5 5 5 0 0 0 1 0 0
Output:
4 4 5 4 5 5 5 0 0 0 0 0 0

Example 5:

Input:
0 4 0 0 5 0 1 1 0 1 2 0 1
Output:
0 0 0 0 0 0 0 0 0 0 0 0 1

Below is the test input sequence. Predict
    the corresponding output sequence.
Your final answer should just be the text
    output sequence itself.

Input:
2 4 0 1 0 1 0 1 3 5 2 5 5
```

J. Model Sizing Reference

Preset philosophy. The baseline-workspace Hydra configurations, archived as release attachments rather than bundled in the compact code artifact, scale width, depth, and embedding size while holding the input serialization, optimizer, and training schedule fixed. The resulting presets target ∼0.1M, ∼1M, and ∼10M trainable parameters, respectively, so that researchers can sweep model capacity without editing multiple files. Sequence length, token vocabulary, and batch sizes remain constant, which keeps the data pipeline and wall-clock comparisons consistent across presets.

Parameter accounting. Table 12 lists the parameter counts realized by the 50-epoch task-embedding runs in the `cellarc100k_50e_{embedding,in_context}` projects (run IDs shown in the main results tables). In-context runs share the same checkpoints, so the totals apply to both regimes; disabling task embeddings only removes the small per-episode bias vectors ($< 0.5\%$ of the budget).

Transformers (vanilla and autoregressive). The encoder-only transformer uses embedding dimensions $E = \{80, 256, 448\}$, layers $L = \{2, 2, 4\}$, heads $H = \{4, 8, 8\}$, dropout 0.1, and widening factors $\{2, 2, 4\}$ for the small/medium/large presets, respectively. The `transformer_ar` variant keeps the depth but reduces the embedding to $56/160/320$ and the widening factors to $1/2/4$ so that teacher forcing remains stable; it was used for ablations but not for the headline results.

LSTM / RNN. The `rnn` preset is a single-layer LSTM whose hidden size scales as $160/512/1536$, paired with token embeddings that match the task-embedding dimension. The teacher-forcing variant (`rnn_ar`) halves the hidden size ($80/256/768$) and ties input/output projections to reduce the softmax cost while keeping the same parameter budgets shown in Table 12.

1D CNN. CNN presets embed tokens into $64/160/256$ channels (small/medium/large), then stack $2/4/5$ dilated residual Conv1d blocks with hidden channels $160/320/896$ and dropout 0.1. Task embeddings are injected as an additive bias after token embedding so that enabling TE does not change the convolutional kernel count.

1D Neural CA. The neural CA baseline converts tokens to channel states with $80/192/384$ channels, applies 2 perception kernels of width 3, iterates for $48/64/96$ steps, and updates a hidden MLP of size $320/1536/6144$. Only the small preset (110k parameters) was trained to completion; the larger presets exceed the memory budget when unrolled for 96 steps, so they remain configuration stubs.

Tiny Recursive Model (TRM). TRM presets use hidden sizes $64/192/608$, $4/4/8$ heads, expansion factors $1.8/2.3/4$, and fixed H/L cycle counts (H=3, L=6 with L-layers $= 2$). Task embeddings are encoded as a single prefix token (`task_emb_len` $= 1$), and the ACT loss follows the stabilized cross-entropy configuration archived with the baseline release attachments.

Architecture	Trainable parameters (M)		
	Small	Medium	Large
Transformer (encoder-only)	0.11	1.06	9.67
Transformer-ACT	0.13	0.91	8.86
Tiny Recursive Model (TRM)	0.13	0.89	9.52
Hierarchical Reasoning Model (HRM)	0.17	0.89	11.15
LSTM / RNN	0.12	1.10	9.58
1D CNN	0.11	1.09	10.35
1D Neural CA	0.11	—	—

Table 12. Trainable parameter counts (rounded to two decimals, in millions) for every architecture and preset. Counts are taken from the 50-epoch task-embedding runs; in-context experiments reuse the same weights. Dashes indicate configurations that were specified but not trained to convergence (see text).

Hierarchical Reasoning Model (HRM). HRM scales hidden sizes $24/96/320$ with $4/4/8$ heads, four H-layers and four L-layers, and two outer/inner cycles. The expansion factor grows from 1.1 (small) to 2.0 (medium) and 4.0 (large). As with TRM, TE is injected through a single prefix token and the halting schedule caps episodes at 16 steps.

Transformer-ACT. The ACT-enabled transformer uses hidden sizes $64/160/384$, heads $4/4/8$, and H-layers $2/4/6$ with expansion $1.8/2.0/2.5$. All presets share the same halting hyperparameters (`halt_max_steps` $= 16$, exploration probability 0.1), so the parameter growth comes purely from wider attention blocks.

Symbolic solvers. The de Bruijn local-map baseline learns explicit lookup tables over the induced centered window of width $C{=}2rt{+}1$ and therefore has no trainable weights; its memory footprint scales with k^C rather than parameter count.

K. Training Dynamics

L. Design Space Exploration

During the development of CellARC, we explored several alternative design choices that ultimately did not yield improvements over the reported configuration. We document these decisions here to inform future work and prevent redundant exploration.

Split construction via Langton's λ only. We initially constructed the extrapolation split by directly stratifying episodes according to Langton's λ parameter (Langton, 1990). Surprisingly, this approach exhibited low correlation with model generalization difficulty. Empirical analysis revealed that query-weighted coverage cov–the fraction of query windows observed in support examples–was a substantially stronger predictor of model performance than Langton's λ alone. We systematically evaluated cov thresholds spanning $[0.0, 1.0]$ (exact coverage), and $[0.5, 1.0]$ (partial coverage). The unrestricted $[0.0, 1.0]$ range produced majority of episodes that proved intractable for all tested architectures, yielding near-random performance even after extended training. Conversely, episodes with exact coverage (1.0) were trivially solved by symbolic solvers as well as neural networks, incentivizing memorization. The intermediate range $[0.5, 1.0]$ achieved the desired balance: episodes remained challenging yet tractable, allowing differentiation among baseline architectures while preserving headroom for future improvements. Finally we used a combination of λ, cov and cell entropy H.

Table 13 highlights how query coverage, Langton's λ, and mean cell entropy H correlate with per-token accuracy for symbolic, neural, and closed models, quantifying the intuition that high coverage helps while chaotic regimes suppress accuracy.

Figure 11 complements the aggregate view by plotting per-task scatter, showing that de Bruijn collapses sharply once cov dips below $\sim$0.6 whereas Transformer-TE and GPT-5 High degrade more gracefully but still suffer as λ and H increase.

Extended vocabulary sizes. We investigated alphabet sizes spanning $k \in [2, 10]$ to assess whether richer color palettes would yield more challenging generalization regimes. This extension caused the explicit rule table representations required for certain symbolic solvers and metadata computation to exceed available memory. Specifically, for radius $r{=}3$ and $k{=}10$ colors, the dense rule table demands $k^{2r+1} = 10^7$ entries, each requiring storage of output states and intermediate computation buffers. Given that our scientific transparency goals (section 1, G5) mandate publishing complete rule tables and Langton's λ statistics for all episodes, we restricted the palette to $k \in [2, 6]$, ensuring that all metadata remain computable within standard memory constraints while still spanning ordered, edge-of-chaos, and chaotic dynamical regimes.

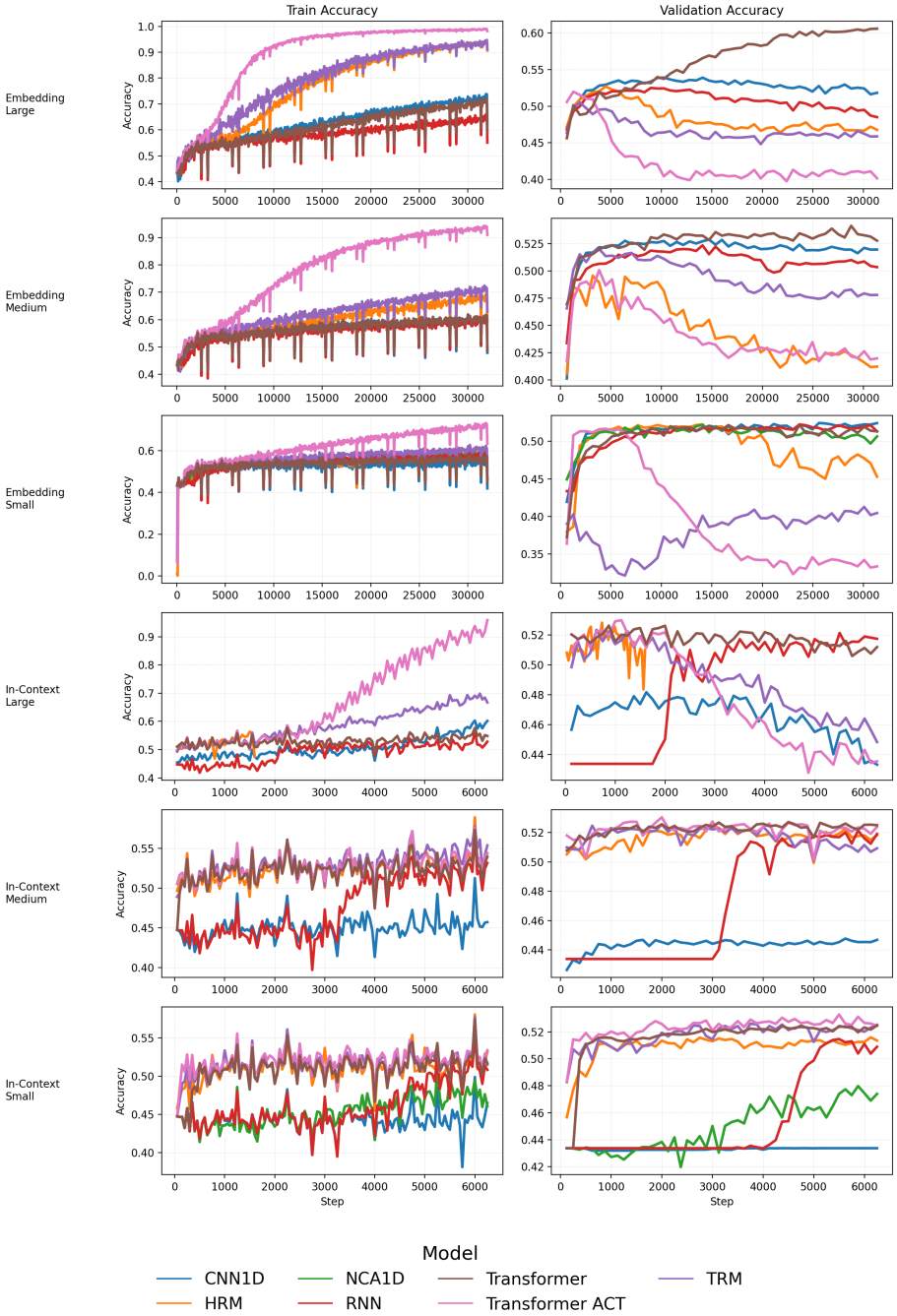

Figure 9. Training and validation accuracy by family, size, and regime. With task embeddings (top three rows), recursive models (TRM/HRM/Transformer-ACT) quickly reach high training accuracy but validation accuracy often peaks early and then degrades, indicating overfitting to the prefix task code. Vanilla Transformer continues to improve and achieves the highest validation accuracy. In the in-context regime (bottom rows), the gap narrows and Transformer-ACT/HRM/TRM stabilize, while CNN1D benefits less from pure ICL.

M. Results for Small and Medium Size Models

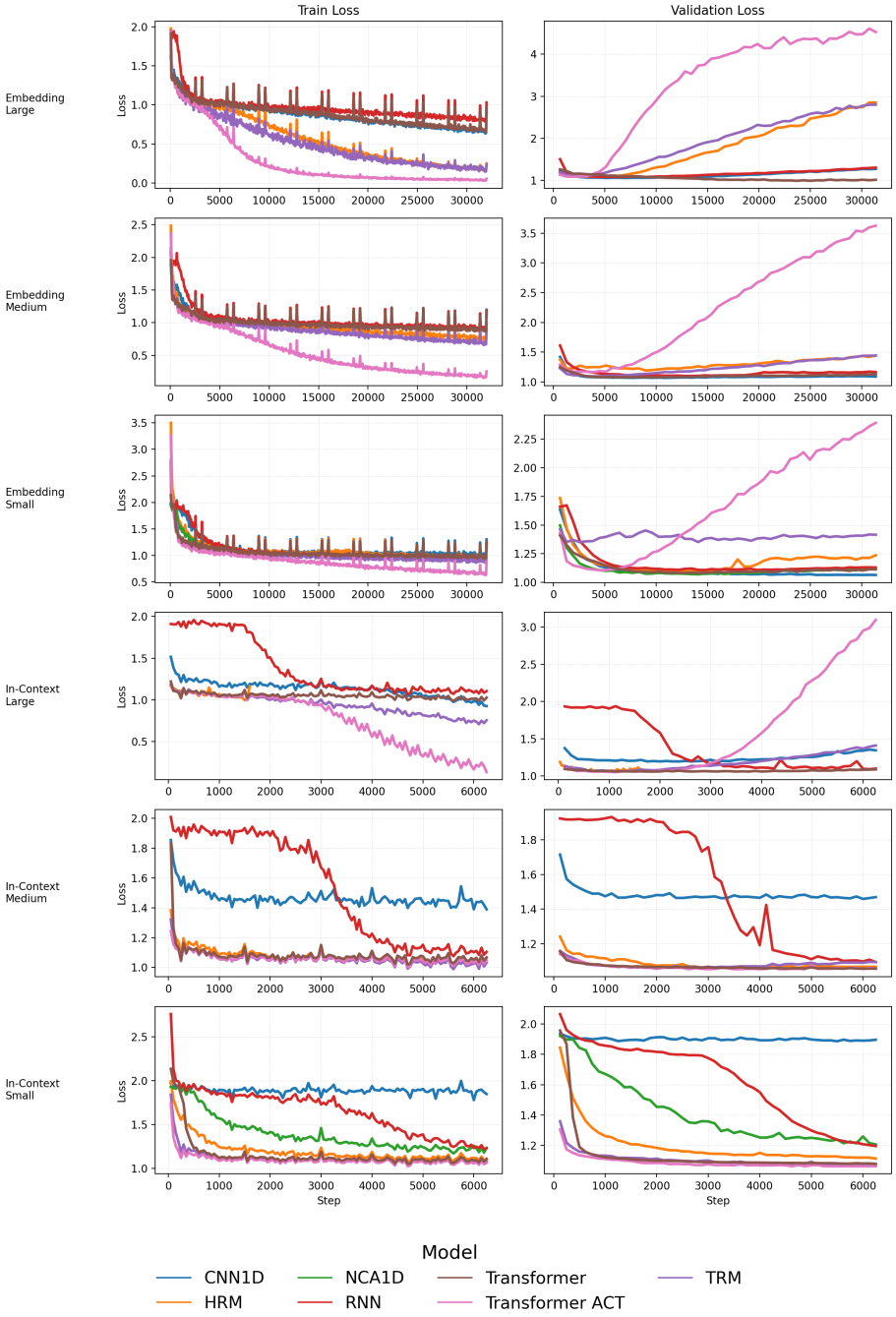

Figure 10. Training and validation loss. Under task embeddings, recursive models show characteristic overfitting signatures: training loss decreases steadily while validation loss bottoms out early then rises. Vanilla Transformer exhibits smoother train–val behavior with lower validation loss. In in-context runs, validation loss remains stable across families, and Transformer-ACT attains competitive curves without task tokens.

Metric	Pearson r			Spearman ρ		
	de Bruijn	Transformer (TE)	GPT-5 High	de Bruijn	Transformer (TE)	GPT-5 High
cov	0.41	0.53	0.35	0.50	0.56	0.29
λ	-0.56	-0.53	-0.29	-0.47	-0.49	-0.26
H	-0.63	-0.69	-0.35	-0.67	-0.72	-0.34

Table 13. Correlation between per-token accuracy and dataset complexity proxies across models on the extrapolation split ($n = 199$ tasks). Higher cov positively correlates with accuracy, while higher λ and cell entropy H correlate negatively. Values are Pearson r and Spearman ρ.

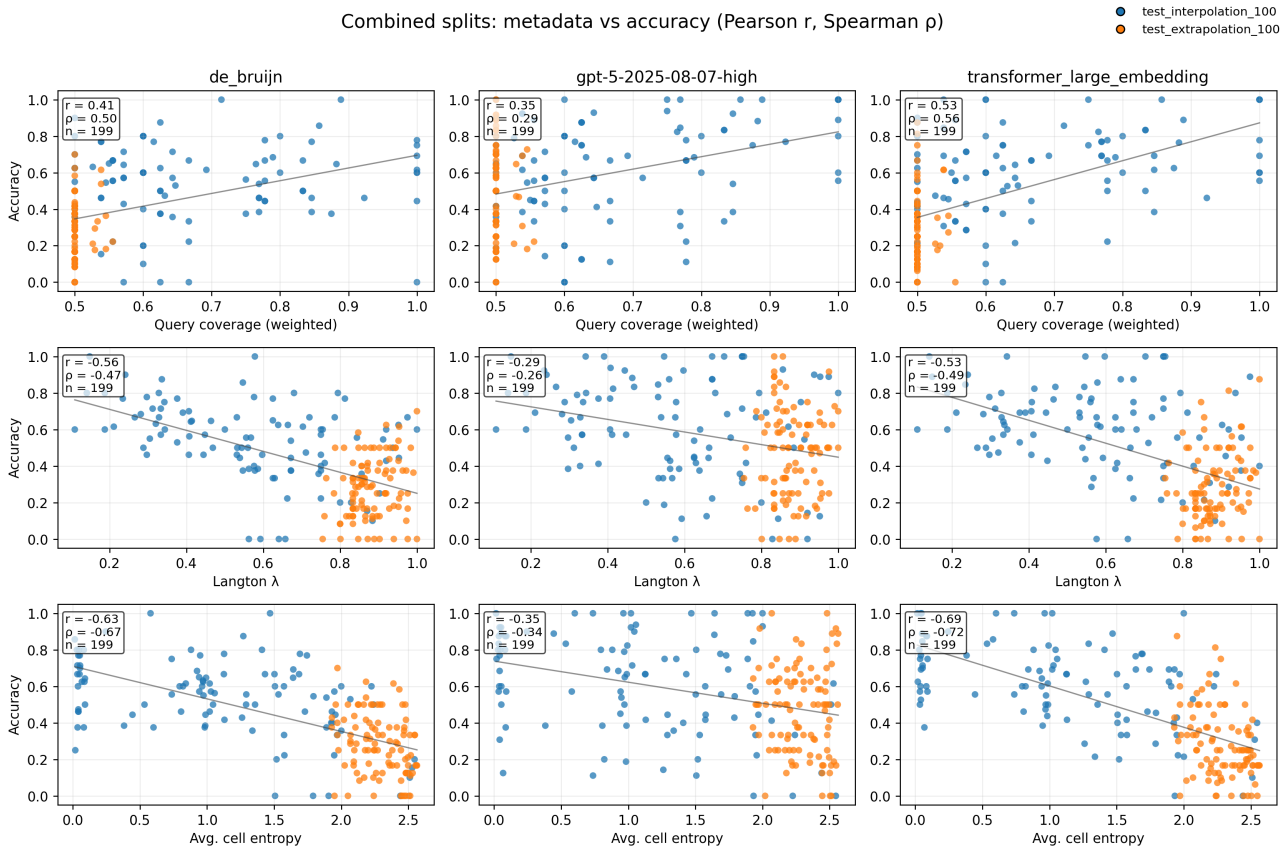

Figure 11. Scatter plots of per-task accuracy versus complexity proxies for three representative models (de Bruijn, Transformer with task embeddings, GPT-5 High). Each column shows accuracy against query-window coverage cov, Langton's λ, and empirical cell entropy H. Points are colored by split: blue for TEST INTERPOLATION 100 and orange for TEST EXTRAPOLATION 100. Both Pearson r and Spearman ρ correlations are shown, complementing the aggregated statistics in Table 13.

Model	Regime	Interpolation	Extrapolation	Δ	Params (M)
Transformer-ACT	In Context (ICL)	50.9	30.0	20.9	0.91
Tiny Recursive	In Context (ICL)	51.0	28.8	22.3	0.89
Transformer	In Context (ICL)	51.0	29.5	21.5	1.06
LSTM	In Context (ICL)	50.6	27.5	23.1	1.10
1D CNN	In Context (ICL)	43.6	13.0	30.6	1.09
HRM	In Context (ICL)	51.2	29.1	22.1	0.89
Transformer	Task Embedding	51.5	28.8	22.6	1.06
1D CNN	Task Embedding	51.6	28.4	23.2	1.09
HRM	Task Embedding	48.9	27.1	21.8	0.89
LSTM	Task Embedding	50.8	28.0	22.8	1.10
Transformer-ACT	Task Embedding	48.9	27.2	21.7	0.91
Tiny Recursive	Task Embedding	50.2	28.9	21.3	0.89
Proposal+Verifier	Symbolic	**87.9**	**88.2**	-0.3	$\approx$0
Posterior Selector	Symbolic	**88.0**	**88.2**	-0.2	$\approx$0
de Bruijn	Symbolic	80.4	60.6	19.8	$\approx$0
Most Frequent	Symbolic	50.4	28.2	22.2	$\approx$0
Copycat	Symbolic	32.7	18.0	14.7	$\approx$0
Random	Symbolic	29.7	17.6	12.0	$\approx$0

Table 14. Per-token accuracy (%) on the interpolation and extrapolation splits for medium-sized models ($\sim$1M parameters). Neural rows are archived size-sweep diagnostics; symbolic rows are size-invariant current full-split baselines copied from the main diagnostic ledger.

Model	Regime	Interpolation	Extrapolation	Δ	Params (M)
Transformer-ACT	In Context (ICL)	50.9	29.5	21.4	0.13
Tiny Recursive	In Context (ICL)	50.6	28.7	22.0	0.13
Transformer	In Context (ICL)	51.0	28.8	22.2	0.11
LSTM	In Context (ICL)	49.9	25.4	24.4	0.12
1D CNN	In Context (ICL)	42.5	10.7	31.8	0.11
HRM	In Context (ICL)	49.7	26.7	22.9	0.17
NCA 1D	In Context (ICL)	46.1	16.8	29.3	0.11
Transformer	Task Embedding	50.8	29.6	21.2	0.11
1D CNN	Task Embedding	51.1	29.9	21.2	0.11
HRM	Task Embedding	50.7	27.9	22.8	0.17
LSTM	Task Embedding	50.7	27.6	23.1	0.12
Transformer-ACT	Task Embedding	50.4	27.9	22.5	0.13
Tiny Recursive	Task Embedding	39.7	21.7	17.9	0.13
NCA 1D	Task Embedding	50.3	28.9	21.4	0.11
Proposal+Verifier	Symbolic	**87.9**	**88.2**	-0.3	$\approx$0
Posterior Selector	Symbolic	**88.0**	**88.2**	-0.2	$\approx$0
de Bruijn	Symbolic	80.4	60.6	19.8	$\approx$0
Most Frequent	Symbolic	50.4	28.2	22.2	$\approx$0
Copycat	Symbolic	32.7	18.0	14.7	$\approx$0
Random	Symbolic	29.7	17.6	12.0	$\approx$0

Table 15. Per-token accuracy (%) on the interpolation and extrapolation splits for small models ($\sim$100k parameters). Neural rows are archived size-sweep diagnostics; symbolic rows are size-invariant current full-split baselines copied from the main diagnostic ledger.