2D Spatial Reasoning with Adaptive Neural Cellular Automata

Martin Spitznagel[1], **Janis Keuper**[1,2]
[1] Institute for Machine Learning and Analytics (IMLA), Offenburg University, Germany
[2] University of Mannheim, Germany
firstname.lastname@hs-offenburg.de

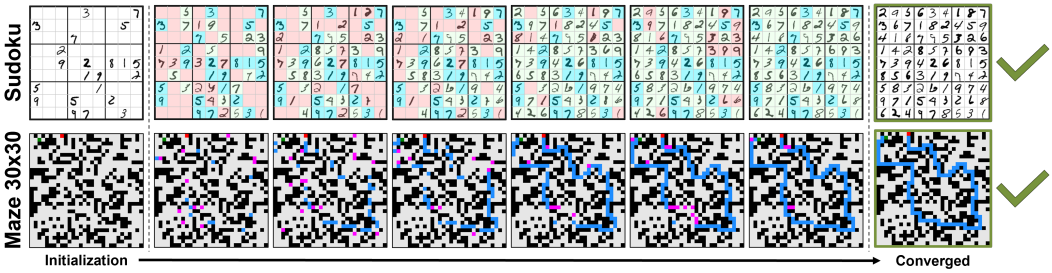


Figure 1: Iterative 2D spatial reasoning at inference time with our proposed *Adaptive Neural Cellular Automata* (aNCA) architecture. Top: solving a hard Sudoku on images containing handwritten (MNIST) digits - achieving a benchmark leading 82.1% average solution rate. Bottom: finding the shortest path through a given maze - achieving benchmark leading 98.56% valid optimal solution rate.

Abstract

Many modern learning approaches are still struggling with spatial reasoning tasks, i.e. they lack the ability to utilize geometric information of perceived entities and their spatial relation to each other to solve problems. We introduce a novel *Adaptive Neural Cellular Automata* (aNCA) architecture which replaces the static and spatially invariant perception of standard NCAs by learnable, spatially variant and data-adaptive perceptive fields. We show that this conceptual change enables NCAs to iteratively reason over 2D spatial relations on grid-like data structures (e.g. images). Empirical results on public benchmarks show state of the art comprehensible results with high generalization abilities for solving image based puzzles like Sudoku or finding the shortest path in a maze.

1 Introduction

Spatial reasoning is the ability to mentally visualize, manipulate, and functionally relate 2D and 3D objects, as well as to understand the relationships between parts, positions, and movements of one or more objects in physical space. It is a key cognitive skill used for navigation, problem-solving, engineering, and designing [19, 13]. Despite its high practical impact, spatial reasoning skills are generally still underdeveloped in most current learning models [2, 19].

In contrast to the recently emerging work on *"World Models"* [6], which try to learn to model complex relations in a top-down fashion, we are following a bottom-up approach which tries to isolate spatial reasoning problems in controlled environments. The focus of our proposed method lies on the also open sub-problem of spatial reasoning on 2D grid structures with a fixed x, y-coordinate system, for which we introduce a novel architecture based on *Neural Cellular Automata* [25]. Following the DISE [13] schematic of spatial reasoning taxonomies, our proposed aNCA architecture is specifically

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

targeting 2D *Extrinsic-Static* problems, that assess the relationships between multiple objects in a fixed scene (like finding the shortest path in 2D mazes [38]) and *Extrinsic-Dynamic* reasoning tasks, that involve reasoning about the changing spatial relationships between multiple objects (like solving Sudokus [40]).

The key contributions of this paper can be summarized as follows:

- We are the first to apply the concept of learnable CA to spatial reasoning tasks, introducing *Adaptive Neural Cellular Automata* (aNCA). For this purpose we move from the space- and time-invariant update rules in traditional NCA to a more powerful and flexible input-dependent, space- and time-variant regime, and show that this significantly boosts spatial reasoning performance
- The inherent recursive nature of aNCA enables us to learn very small, yet competitive and highly generalizable models that are able to infer diverse solutions to ambiguous problems. Visualizing the learned perceptive fields allows us to make the reasoning of spatial relations interpretable.
- aNCA results surpass current state-of-the-art 2D spatial reasoning methods on various public benchmarks, including Sudoku, image-based Sudoku and maze shortest-path finding.

1.1 Related Work

The advancement of spatial reasoning in neural networks has evolved from basic object detection to complex relational logic and artificial dynamic mental simulation. Despite the linguistic fluency of modern multimodal large language models (MLLMs), current research identifies a persistent *"semantic-to-geometric gap"* where models fail to ground spatial prepositions in coherent geometric world models [2, 19].

Cognitive Frameworks and Taxonomies. To diagnose these systematic failures, recent literature has adopted cognitively grounded taxonomies. The DISE framework [13] categorizes spatial tasks into a 2×2 matrix based on Intrinsic vs. Extrinsic properties and Static vs. Dynamic environments. While models are proficient in static settings, they exhibit a high failure rate in tasks requiring dynamic mental simulation, primarily due to deficits in "spatial working memory" and an inability to apply basic geometric axioms.

Architectural Inductive Biases. Traditional architectures often lack the structure for relational reasoning. To address this, Relation Networks (RNs) were introduced to explicitly compute pairwise relations between all objects in a set [32]. For more complex scenes, Scene Graph Neural Networks (SGNNs) represent environments as structured graphs where nodes denote objects and edges represent their relationships, utilizing message-passing to propagate context [46, 29]. More recently, mechanisms for Vision Transformers (ViTs), such as Context-Aware Gating (CAG) in the Spatial Decay Transformer (SDT), inject data-dependent spatial decay into self-attention to selectively focus on semantically relevant regions while maintaining a locality bias [23, 7].

General Spatial Benchmarking. No generally accepted closed-form definition reduces spatial reasoning ability to a single numerical score. Evaluation therefore relies on proxy problems that isolate particular spatial capabilities; designing such benchmarks is itself an active field of research, and their results provide task-specific rather than universal measures [13]. Benchmarking spatial reasoning abilities has transitioned from early synthetic 2D/3D datasets like CLEVR [16] and GQA [14] to complex 3D and multi-hop scenarios. Benchmarks such as StepGame [34] and SpartQA [24] test multi-step logical inference in text and vision. For 3D awareness, 3DSRBench [22] and Ego3D-Bench evaluate robustness across varying camera viewpoints, while SITE [39] and OmniSpatial [15] provide comprehensive evaluations across diverse spatial factors including figural and environmental scales. SpatiaLQA emphasizes "spatial logical reasoning" within open-vocabulary indoor environments [43].

Spatial Reasoning on 2D (Grid) Data. While most of the previous benchmarks focus on spatial reasoning in 3D scenes (or 3D→2D projections of these), there is a growing body of recent works that focus on pure 2D spatial reasoning: e.g. [40] introduced spatial reasoning with SRN Diffusion Models which operate on images. Another line of work proposed to solve 2D reasoning with *Hierarchical Reasoning Models* (HRM) [38], an approach which has recently been extended by ItrSA++ [20] and TRM [17].

Benchmarks for 2D Spatial Reasoning. 2D tasks not only provide simpler spatial reasoning problems which allows the study of novel approaches more easily, but also have several practically relevant benchmark applications like solving Sudoku on images of handwritten digits [40], finding

the shortest path in a maze [38] or finding spatially local solutions with global constraints [40]. We compare aNCA with state-of-the-art SRM [40], HRM [38], and TRM [17] on the tasks introduced in section 3.

2 Neural Cellular Automata

Originating from the seminal work of *John von Neumann* [37], *Cellular Automata* (CA) are discrete computational systems studied since the 1940s. Typically, a CA is defined on a regular, finite grid $\mathcal{G}$ composed of discrete cells. At each time step t, every cell i maintains a discrete state $s_{i,t}$. The system's evolution is governed by the synchronous application of a rule set $\mathcal{R}$ across all cells, where the subsequent state $s_{i,t+1}$ is derived from the local neighborhood $\mathcal{N}(s_{i,t})$ such that $s_{i,t+1} := \mathcal{R}[\mathcal{N}(s_{i,t})]$ [33]. This process is illustrated in Figure 2.

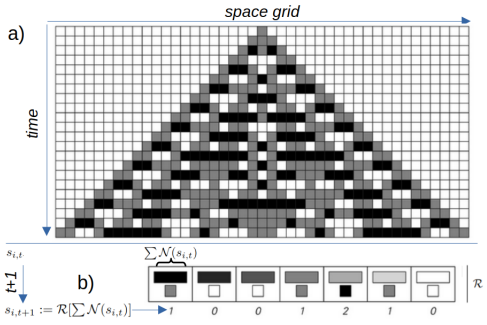


Figure 2: Sketch of a basic 1D Cellular Automaton with three discrete states $s_i \in \{0, 1, 2\}$ and seven update rules $\mathcal{R}$, which operate on the sum over a 3-cell neighborhood $\mathcal{N}$. a) shows the inference of complex patterns from an initial state by the rules visualized in b) over time.

CAs gained significant prominence in the 1970s via *Conway's "Game of Life"* [8], a 2D CA utilizing binary states $s_{(x,y),t} \in \{0, 1\}$ and 3×3 neighborhood rules. CAs have proven effective in modeling diverse phenomena in biology [4, 1, 12], chemistry [9], and physics [41, 45]. Furthermore, their status as a universal model of computation was confirmed by the proof of *Turing Completeness* [3] for specific configurations. These developments underpinned *Stephen Wolfram's "A New Kind Of Science"* [42], which posited that complex scientific problems might be better characterized by simple recursive programs like CAs than with differential equations.

Traditional CA models, such as those discussed in [42], [8], and [37], utilize manually defined rule sets. While these rules can be enumerated combinatorially for a given state space $\mathcal{S}$ and neighborhood size $|\mathcal{N}|$, the search space expands exponentially; for instance, a binary 2D CA with a 3×3 neighborhood already permits $2^{2^{3\times3}} = 2^{512}$ possible rules. To address this complexity, [25] proposed learning rules from data, introducing ***Neural Cellular Automata*** **(NCA)**. In this framework, handcrafted rules are replaced by an artificial neural network f_ϕ with learnable parameters ϕ. The update step is expressed as:

$$s_{i,t+1} := s_{i,t} + f_\phi[\mathcal{N}(s_{i,t})] \tag{1}$$

Since f_ϕ is applied identically to every position i (spatial invariance), [25] utilized "nested" *Convolutional Neural Networks* (CNNs) [21] for efficient implementation. Here, convolutional filters function as learnable local rules over kernel-defined neighborhoods. This architecture is depicted in Figure 3. While [10] established the theoretical equivalence between CA and such CNNs, most NCA models depart from tradition by employing continuous vector-valued states $s_{i,t} \in \mathbb{R}^n$ rather than discrete ones.

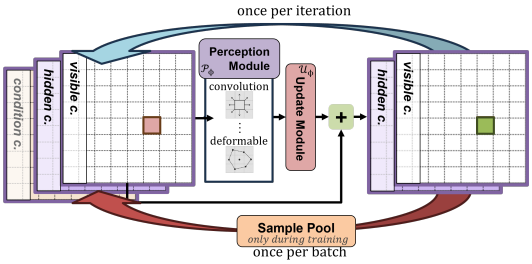


Figure 3: Sketch of a basic CNN implementation [35] of a 2D NCA with a 3D state space. The perception module processes the cell state using one of several configurable perception options. The update module processes the perceived features to compute state updates. Stochastic updates are applied before updating the visible and hidden channels to form the new cell state for the next iteration. The sample pool is used after one batch of images is processed to retain and reuse evolved states for subsequent training iterations.

Initial work by [25] demonstrated robust self-organization and classification using single-layer convolutional NCAs. This has since expanded into more expressive architectures: [31] applied larger CNNs for segmentation, [36] introduced attention-based variants and [27] real-time dynamic texture synthesis using NCAs. Furthermore, NCAs have been integrated with GANs [26], VARs [28], and diffusion models [18] for generative tasks, while [11] extended

the paradigm to graph-structured data. Due to their inherent spatial structure, NCAs intuitively appear to be well suited for (2D) spatial reasoning tasks. However, to the best of our knowledge, this has never been attempted before.

2.1 NCA Perception Modules

Building on eq. (1), we follow the notation from [35] and factor the learnable update rule f_ϕ into two distinct sub-modules: a *perception module* $\mathcal{P}_\phi$ and an *update module* $\mathcal{U}_\phi$, and re-formulate the NCA iteration as:

$$s_{i,t+1} = s_{i,t} + \mathcal{U}_\phi\big[\mathcal{P}_\phi(s_{i,t})\big]. \tag{2}$$

The perception module $\mathcal{P}_\phi$ determines how each cell reads information from its neighborhood and therefore controls the effective receptive field and communication structure of the automaton, while the update module $\mathcal{U}_\phi$ transforms the perceived features into the state increment applied to the cell. In our implementation, the cell state is first concatenated channel-wise with the task condition and, when enabled, fixed positional embeddings. The resulting tensor is processed by one or more perception branches operating in parallel; their outputs are concatenated and passed to a shared 1×1 convolutional update network that predicts the state increment. Because we keep $\mathcal{U}_\phi$, the cell-state structure, and the training protocol fixed across experiments, any difference in task performance can be attributed to the inductive bias of $\mathcal{P}_\phi$. Figure 3 illustrates how these components interact within a single NCA step. In the remainder of this section we describe three perception variants: standard $k \times k$ convolution (e.g. 3×3 or 5×5), dilated convolution and a Sudoku-specific constraint-aligned variant that combines row, column, and 3×3-box convolutions.

Standard $k \times k$ convolution. Our baseline perception is a single learnable $k \times k$ convolution layer with 128 kernels and leaky-ReLU activation and circular padding, following the original formulation of [25]. Each cell's receptive field is therefore limited to its immediate neighbors per step, and longer-range dependencies must be built up recurrently.

Dilated Convolutions. We propose dilated convolutions [44] as an NCA perception module for spatial reasoning tasks requiring long-range communication. A dilated 3×3 kernel with rate r samples a $(2r+1) \times (2r+1)$ grid region at no extra parameter cost, directly enlarging the per-step receptive field. In the NCA setting, fewer iterations are needed to propagate information across the grid, which is a particularly effective inductive bias for tasks where constraints span the entire grid.

Manually Constraint-Aligned Convolutions. For tasks whose constraint structure is known a priori, we propose to integrate that structure manually into the perception module itself rather than asking the NCA to discover it over many iterations. For Sudoku in particular, every valid solution must satisfy three constraint families, one per row, column, and 3×3 box, each of which spans an entire "unit" of the 9×9 grid. We therefore combine three parallel perception branches whose kernels match the shapes of these units: a 1×9 *row convolution*, a 9×1 *column convolution*, and a structured 3×3 *box perception* that aggregates the other cells in each 3×3 box.

2.2 Training NCA

Following [35], we train the NCA by unrolling the update rule for a randomly sampled number of steps $T \sim \mathcal{U}[T_{\min}, T_{\max}]$ and comparing the resulting state to the target output under a task-specific objective. For conditional tasks such as *Sudoku* and maze solving, each cell is initialized from the puzzle input and conditioned on the same input throughout the rollout, so that the observable evidence remains available at every step. Gradients are backpropagated through the full unrolled trajectory. To further encourage temporal stability, we adopt the *sample pool* strategy of [25]: a fraction of each mini-batch is initialized from previously evolved states drawn from a fixed-size buffer rather than from fresh seeds, and the newly evolved states are written back into the pool after each training step. This persistent-state regime exposes the update rule to intermediate configurations far from the initial seed and discourages it from collapsing or diverging on longer rollouts, which is essential for tasks that

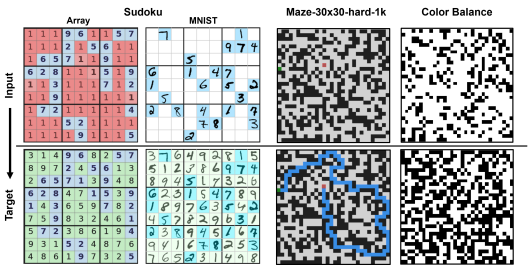


Figure 4: The four 2D spatial reasoning tasks evaluated in this work. Each column shows the input (top) and target (bottom).

require many iterations to propagate information across the grid. Full hyperparameter details for each task are listed in sections A.3.1, A.5.1 and A.6.1 in the Appendix.

3 Spatial Reasoning with NCA

To the best of our knowledge, we are the first to propose the use of NCAs for (2D) spatial reasoning tasks. We cast each spatial reasoning task as a recurrent computation on a 2D grid $\mathcal{G}$, solved by the NCA defined in eq. (1). All tasks share the same architecture, optimizer, and sample-pool training regime: we unroll for $T \sim \mathcal{U}[T_{\text{min}}, T_{\text{max}}]$ steps, optimize with AdamW under gradient clipping, and concatenate the condition channel-wise into the cell state at *every* step. Because high-difficulty Sudoku puzzles admit many valid completions, path-finding admits shortest-path ties, and the color-balance task has no per-pixel supervision, we use *constraint-based* objectives that reward any valid output. The four tasks, illustrated in fig. 4, use established benchmarks from prior work.

Sudoku. We use the one-million-puzzle dataset of [40], generating puzzles on the fly by masking cells at a configurable difficulty; a held-out set of solutions is never seen during training. Each cell carries $D = 9$ visible digit channels and the condition $c_i \in \{0, 1\}$ marks clue cells. The objective exploits the puzzle's constraint structure: letting $p_{i,d}$ be the softmax probability of digit d at cell i, we penalize the squared deviation of per-digit unit sums from one across all rows, columns, and 3×3 boxes, rewarding any structurally valid completion independently of which specific digits the NCA commits to. Training hyperparameters, loss details, and computational cost are provided in section A.3.

MNIST Sudoku. We reuse the same solutions and split, rendering each cell as a handwritten glyph drawn from MNIST to give a $9s \times 9s$ image where each $s \times s$ tile is blank or carries a digit. The condition is a binary pixel mask over given cells. End-to-end NCA training exposed a trade-off: rollouts either produced crisp MNIST glyphs without satisfying Sudoku constraints, or satisfied constraints while letting glyphs drift. We therefore train a separate *Sudoku codec* (fig. 5): a frozen encoder E_ψ maps a pixel Sudoku to a $9 \times 9 \times 9$ digit-logit grid and a frozen decoder D_ω renders a board back to pixels via a conditional VAE with a style latent. The NCA operates entirely in board latent space under the same constraint objective as the array variant, with per-digit probabilities supplied by a frozen MNIST-CNN applied to the decoded tiles rather than read from raw digit channels. Codec architecture, training details, and an isolation evaluation of encoder and decoder accuracy are provided in section A.4.

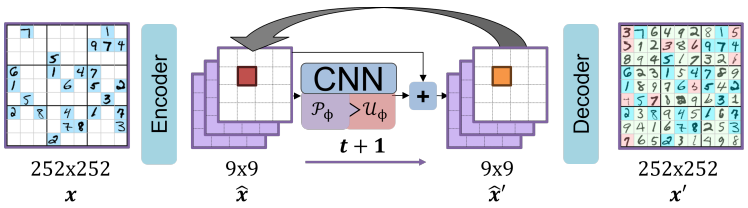


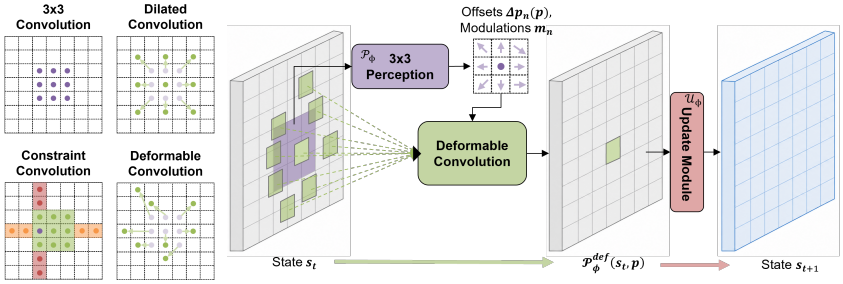
Figure 5: Latent pixel Sudoku pipeline. The encoder E_ψ maps the initial handwritten Sudoku image into a $9 \times 9 \times 9$ grid of digit logits, the NCA iterates in this latent board space under the Sudoku constraint loss until a valid completion emerges, and the decoder D_ω renders the final board back to pixels. Freezing E_ψ and D_ω decouples perception from reasoning and allows different NCA variants to be evaluated within the same image-space pipeline.

Maze solving. We use the 30×30 maze dataset of [38] (1K train, 1K test), with shortest-path targets. The condition encodes walls, start, and goal. The target is a binary shortest-path mask. Since walls are trivially predictable from the condition, the loss is restricted to non-wall cells and combines a Dice term (handling path/non-path imbalance), a BCE term (encouraging binary commitment), and a normalized length penalty $|\text{len}_{\text{pred}} - \text{len}_{\text{true}}|$. Training hyperparameters and loss details are provided in section A.5.

Global color balance. We adapt the global-constraint task of [40] to black-and-white pixels. Starting from a randomly biased binary image, the NCA must fill an unconditioned grid to realize a prescribed black/white ratio (e.g. $50/50$), with no per-pixel supervision. The objective combines a count term pushing the global color mean toward the target ratio and a saturation term that zeroes only at fully committed pixel values. Training hyperparameters and loss details are provided in section A.6.

4 Adaptive Neural Cellular Automata (aNCA)

We propose *Adaptive Neural Cellular Automata* (aNCA), which extend the reasoning abilities from learned space- and time-invariant update rules of traditional NCA to a more powerful and flexible input-dependent, space- and time-variant regime. We achieve this significant change in NCA capabilities through a multi-head perception module based on DCNv2 *deformable convolutions* [5, 47], which instantiates the NCA framework of eq. (2) as perception module $\mathcal{P}_\phi$ and allows the effective neighborhood $\mathcal{N}$ to adapt across grid positions and NCA steps. Figure 6 illustrates the architecture: fig. 6a contrasts the fixed sampling patterns of the perception variants above against the adaptive sampling of the proposed deformable perception, fig. 6b shows the full single-step aNCA computation graph. The deformable perception module is formalized below.



(a) Fixed vs. adaptive perception patterns.

(b) aNCA single-step computation graph.

Figure 6: (a) Comparison of perception sampling patterns: standard 3×3, dilated, constraint-aligned, and deformable convolution. (b) One aNCA update step: the perception module $\mathcal{P}_\phi$ predicts offsets $\Delta\mathbf{p}_n(\mathbf{p})$ and modulations m_n from s_t, which guide the deformable convolution to aggregate features from adaptive grid locations. The update module $\mathcal{U}_\phi$ then produces the next state s_{t+1}.

Adaptive Perception via Deformable Convolutions. Unlike dilated convolutions, which fix the neighborhood geometry at initialization, *deformable convolutions* [5] augment each kernel tap with a learned, position-dependent spatial offset. Rather than committing to a fixed sampling pattern, the model learns *which* positions each cell should attend to, depending on the current input. This allows it to discover long-range constraint structures from data without explicit apriori modeling of these structures. Concretely, a standard $k \times k$ convolution at grid position $\mathbf{p}$ reads

$$\mathcal{P}(s, \mathbf{p}) = \sum_{n=1}^{k^2} w_n \cdot s(\mathbf{p} + \mathbf{p}_n)\,, \tag{3}$$

where $\mathbf{p}_n$ enumerates the k^2 fixed kernel positions and w_n are learnable weights. We use the DCNv2 variant [47], which augments each kernel position with both a spatial offset $\Delta\mathbf{p}_n(\mathbf{p}) \in \mathbb{R}^2$ and a learned modulation scalar $m_n(\mathbf{p}) \in [0, 2]$, each predicted by a separate auxiliary convolution applied to $s_{i,t}$. Its difference from the offset-only DCNv1 variant is detailed in section A.2.1. Fractional positions are resolved by bilinear interpolation:

$$\mathcal{P}_\phi^{\text{def}}(s, \mathbf{p}) = \sum_{n=1}^{k^2} w_n \cdot m_n(\mathbf{p}) \cdot s(\mathbf{p} + \mathbf{p}_n + \Delta\mathbf{p}_n(\mathbf{p}))\,. \tag{4}$$

The modulation scalar allows each kernel position to be suppressed or amplified independently, giving the model additional flexibility beyond the spatial offset alone. We stack $H = 3$ deformable branches in parallel as $\mathcal{P}_\phi$, producing $3 \times (2 + 1) \times 9 = 81$ offset and modulation values per cell per step. Different heads can thereby specialize to complementary constraint directions. Because the learned offsets are explicit and inspectable, we can directly audit which spatial relations the NCA has discovered. In Section 5.1 we show that the three heads learn sampling patterns that align with Sudoku's row, column an box constraints, without encoding this explicitly in the architecture.
Our approach significantly differs from the "deformable convolution" implemented in in the NCA-Torch framework of [35] - there a single deformable convolution based on [5] has been (unsuccessfully) used for generative tasks and is not identical with our formulation (which is based on [47]. In

Contrast to [35], our per-tap modulation lets a cell suppress or amplify individual sampling positions rather than only displace them. This allows different heads to specialize to complementary constraint directions, which is critical for spatial reasoning.

5 Empirical Evaluation

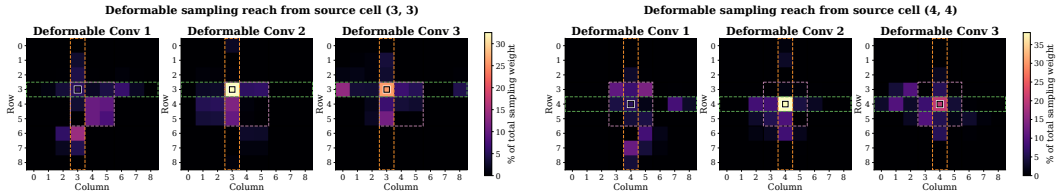
5.1 Solving *Sudoku* on Array Representations

We evaluate on the held-out test split of the SRM Sudoku dataset [40], grouping puzzles by difficulty into Easy (1–27 removed cells), Medium (28–54), and Hard (55–81) buckets. Two metrics are reported per bucket: the *Average Solve Rate* (ASR), the fraction of test puzzles whose final NCA state is a valid Sudoku completion of the given clues, and the *Average Cell Precision* (ACP), the fraction of non-clue cells whose predicted digit is consistent with a valid completion. Because high-difficulty puzzles often admit several valid completions, ASR is the primary metric.
Table 1 compares the perception variants at matched cell-state size and network capacity, reporting mean and standard deviation over four training runs each. All variants solve Easy and Medium puzzles near-perfectly, so the evaluation performance is decided on the Hard buckets, where aNCA leads on both metrics while remaining comparable in parameter count. The aNCA performance gain can not be explained by receptive-field size or capacity: i.e. enlarging the fixed kernel beyond 5×5 does not increase the performance, 7×7 and 9×9 variants even perform worse on Hard despite using more parameters then aNCAs. The 9×9 variant is also markedly unstable across seeds (± 37.15 on Hard), whereas the smaller fixed kernels are reproducible.

Table 1: Sudoku on the array representation, reported as mean $\pm$ standard deviation over four training runs per perception. Best Hard-bucket ASR/ACP in bold.

Perception	Easy [1, 27]		Medium [28, 54]		Hard [55, 81]		Param.
	ASR	ACP	ASR	ACP	ASR	ACP	
NCA w. 3x3 Conv.	100 ± 0	100 ± 0	86.35 ± 1.00	98.85 ± 0.13	78.23 ± 1.84	98.63 ± 0.10	50k
NCA w. 5x5 Conv.	100 ± 0	100 ± 0	87.47 ± 1.65	98.93 ± 0.12	75.03 ± 8.38	96.27 ± 4.05	105k
NCA w. 7x7 Conv.	100 ± 0	100 ± 0	82.70 ± 1.62	98.43 ± 0.21	76.95 ± 2.07	98.43 ± 0.15	189k
NCA w. 9x9 Conv.	100 ± 0	100 ± 0	71.58 ± 18.82	95.88 ± 4.41	39.80 ± 37.15	84.95 ± 19.71	300k
NCA w. R/C/B Conv.	100 ± 0	100 ± 0	87.38 ± 2.64	98.90 ± 0.22	78.18 ± 2.81	98.58 ± 0.19	145k
NCA w. Dilated Conv.	100 ± 0	100 ± 0	86.98 ± 1.64	98.88 ± 0.17	77.23 ± 10.43	98.48 ± 0.67	150k
NCA w. DCNv1	99.9 ± 0.14	100 ± 0	75.78 ± 6.93	97.75 ± 0.7	64.55 ± 10.43	97.5 ± 0.58	159k
aNCA	100 ± 0	100 ± 0	$\mathbf{89.50 \pm 4.66}$	$\mathbf{99.08 \pm 0.40}$	$\mathbf{87.78 \pm 6.74}$	$\mathbf{99.18 \pm 0.46}$	165k

Visualizing Learned Spatial Relations. A central appeal of the adaptive perception module is that the predicted per-cell offsets are themselves interpretable: for every Sudoku cell position at every timestep, they allow to reconstruct to which other cells the model learned to attend. We exploit this to ask whether the adaptive perception learns Sudoku's row, column, and 3×3-box constraint graph from data, despite this is never explicitly modeled. For each test puzzle we run the trained NCA for



(a) Source cell $(3, 3)$.

(b) Source cell $(4, 4)$.

Figure 7: Learned sampling reach of each deformable perception head from two interior source cells (black-bordered squares at grid positions $(3, 3)$ and $(4, 4)$). Each panel is one head: the heatmap shows the fraction of the head's total sampling weight placed on every cell of the 9×9 board, aggregated over held-out test puzzles at the final NCA step. Dashed overlays mark the source cell's row (green), column (orange), and 3×3 box (purple). The learned heat concentrates inside these overlays and the three heads specialize to complementary constraint directions.
the standard rollout horizon and record every head's tap offsets at the final step. Fixing one source cell on the 9×9 board, we then aggregate where the nine taps of each head end up sampling from that cell across held-out testset puzzles. Figure 7 shows the result as one heatmap per head, for two

interior source cells ($(3, 3)$ at the intersection of the top-left 3×3 box with the center box, and $(4, 4)$ at the center of the board).

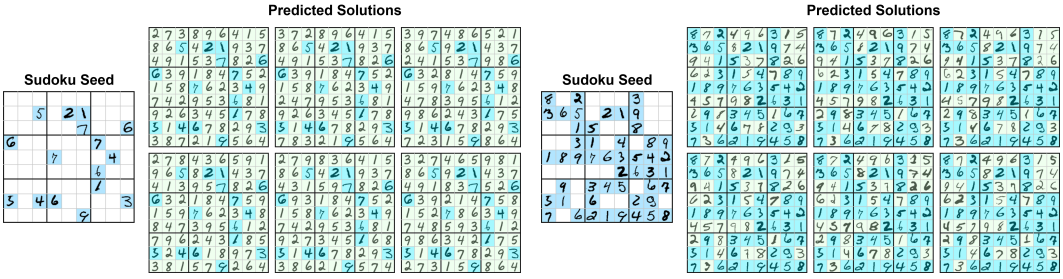
Dashed rectangles overlay the source cell's row, column, and 3×3 box, so the reader can directly compare the learned hot cells against Sudoku constraints. Across both positions and all heads, the heat concentrates inside those three rectangles. The heads also specialize, with different deformable convolutions preferring the row, column, or box direction, so that together they cover the full constraint graph of the source. Quantitative alignment scores are reported in table 7.

5.2 Solving *Sudoku* on Images

We extend the perception comparison to the image-level Sudoku task, in which each cell is rendered as an MNIST-style digit glyph [40]. The NCA operates in the latent board space defined by the frozen encoder–decoder pipeline of fig. 5. Held-out splits, difficulty buckets, and metrics are reused, with ASR evaluated with a frozen MNIST classification CNN on the decoded pixel image. We further compare against the two image-space Sudoku solvers released with the dataset [40]: a 118M-parameter diffusion model and SRM at the same capacity. Unlike these end-to-end baselines, our pipeline decouples digit recognition from constraint reasoning via the frozen codec. This modularity potentially allows the same reasoner to operate on any object encoding for which a codec maps inputs to the board representation. Codec isolation accuracy is reported in section A.4.3. Table 2 reports the results. aNCA surpass both reference models on every difficulty bucket while using fewer parameters. Among the NCA variants themselves, the ordering matches the array task, with aNCA ahead on Medium and Hard. The parameter column lists the shared codec and the NCA separately. The reported NCAs are the *exact* checkpoints evaluated in table 1 on the array task.

Table 2: MNIST Sudoku. Results in ASR (Average Solve Rate), reported as mean ± standard deviation over four training runs. Best Hard-bucket ASR in bold. Per-bucket ACP is reported in table 9.

Method	Easy [1, 27]	Medium [28, 54]	Hard [55, 81]	Param.
Diffusion Model	99.4	53.6	0.8	118M
SRM	99.8	75.4	51.6	118M
				Codec + NCA
NCA 3x3 Conv.	99.40 ± 0.14	80.90 ± 0.90	71.98 ± 1.61	1.1M + 50k
NCA 5x5 Conv.	99.13 ± 0.50	79.97 ± 1.70	69.00 ± 6.63	1.1M + 105k
NCA 7x7 Conv.	98.73 ± 0.10	76.55 ± 1.21	72.30 ± 2.60	1.1M + 189k
NCA 9x9 Conv.	99.30 ± 0.23	66.73 ± 16.99	35.83 ± 33.56	1.1M + 300k
NCA R/C/B Conv.	99.20 ± 0.08	80.38 ± 1.41	71.25 ± 2.61	1.1M + 145k
NCA Dilated Conv.	99.25 ± 0.17	80.78 ± 0.95	70.95 ± 8.39	1.1M + 150k
NCA DCNv1	98.43 ± 0.54	70.15 ± 6.79	60.13 ± 9.75	1.1M + 159k
aNCA	99.00 ± 0.37	**81.90 ± 5.50**	**82.10 ± 4.89**	1.1M + 165k

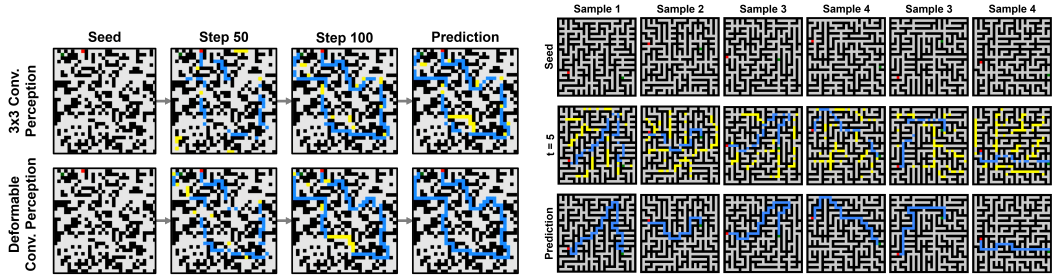


(a) (b)

Figure 8: Output diversity of the Sudoku codec. (a) For an underconstrained clue pattern with at least 10^6 valid completions, repeated stochastic NCA rollouts produce multiple distinct valid solutions. (b) For a puzzle with a unique solution, resampling the decoder style latent preserves the solved board while varying the digit style.

Output Diversity. Highly underconstrained Sudoku puzzles admit many valid completions, so matching the dataset target is neither necessary nor desirable. Sampling the same NCA 1000 times on a puzzle with at least 10^6 valid completions yields valid completions in 997/1000 rollouts and 303 distinct boards (fig. 8a). For a puzzle with a unique solution, resampling produces identical boards but varied digit stroke styles (fig. 8b).

5.3 Solving Maze Puzzles



(a) (b)

Figure 9: (a) Maze solving: the model receives walls (black), start (green), and goal (red). Blue cells indicate correct path predictions matching the target; yellow cells are predicted path pixels that deviate from the target. (b) Generalization to unseen 30×30 Prim-style maze structures.

We evaluate on the 30×30 mazes described in section 3 and report two graph-level metrics: *Valid Path*, the fraction of predictions that connect start and goal without crossing walls, and *Valid Optimal*, the fraction whose path length matches the shortest-path target.

Table 3 shows that purely local perception is insufficient: the 3×3 NCA almost never forms a complete path. Enlarging the communication radius closes much of this gap, with dilated perception reaching 64.30% optimality. Adaptive NCAs perform best, producing valid paths for 98.56% of mazes and optimal paths for 95.2%, surpassing HRM, ItrSA++ and TRM-Att while using fewer parameters. The R/C/B variant also transfers well despite being designed for Sudoku, but trails deformable perception, suggesting that content-adaptive routing is more useful than a fixed constraint geometry on mazes. Same-resolution transfer to a procedurally distinct maze distribution is reported in section A.5.3.

Table 3: Maze solving on 30×30 grids, reported as mean $\pm$ standard deviation over five training runs. Valid Path checks connectivity and wall avoidance. Valid Optimal additionally requires shortest-path length.

Method	Source	Valid Path	Valid Optimal	Param.
HRM	[38]	–*	74.5%	27M
ItrSA++	[20]	–*	78.6%	3M
TRM-Att	[17]	–*	85.3%	7M
NCA w. 3x3 Conv.	Ours	2.54 ± 0.84	0.08 ± 0.08	50k
NCA w. 5x5 Conv.	Ours	24.78 ± 1.68	10.78 ± 1.21	105k
NCA w. Dilated Conv.	Ours	74.26 ± 1.07	64.30 ± 1.32	150k
NCA w. R/C/B Conv.	Ours	93.78 ± 1.99	81.26 ± 2.01	280k
NCA w. DCNv1	Ours	86.73 ± 4.58	77.48 ± 6.42	236k
aNCA	Ours	$\mathbf{98.56 \pm 0.72}$	$\mathbf{94.18 \pm 1.14}$	256k

* indicates metrics not reported in prior work.

5.4 Learning to Adhere Global Constraints

The color-balance task tests whether the NCA can satisfy a constraint defined over the entire grid, without local target labels or a conditioning image. We evaluate the output by its absolute color-count error and exact-ratio accuracy, reporting mean and standard deviation across independent training runs in table 4. Table 4 shows that aNCA achieves the highest mean exact-ratio accuracy among the compared models while using substantially fewer parameters than the diffusion and SRM references. The 5×5 NCA remains competitive and achieves the lowest mean count error. These metrics capture different aspects of performance: exact accuracy measures how often the global constraint is satisfied, whereas mean count error measures the average deviation from the $50/50$ balance. These results show that adaptive perception also helps for a spatially uniform objective. In table 4, aNCA with modulated DCNv2 clearly improves exact accuracy over offset-only DCNv1, while gains over the 5×5 baseline are smaller and within run-to-run variability (also see fig. 11 for qualitative results).

Table 4: Global color balance. NCA results are mean $\pm$ standard deviation across independent training runs. Imbalance measures absolute color-count error. Accuracy measures exact satisfaction of the $50/50$ ratio.

Perception	Source	Imbalance ↓	Accuracy ↑	Param.
Diffusion	[40]	1.27	25.0%	19.7M
SRM	[40]	0.53	51.8%	19.7M
NCA w. 3x3 Conv.	Ours	1.13±0.17	26.0%±5%	48k
NCA w. 5x5 Conv.	Ours	**0.07±0.06**	95.3%±2%	100k
NCA w. DCNv1	Ours	1.61±0.59	18.7%±11%	147k
aNCA	Ours	0.1±0.08	**96.8%±3%**	152k

6 Conclusion

We introduced *Adaptive Neural Cellular Automata* (aNCA), which implement a conceptually novel perception approach to enable NCAs to learn reasoning capabilities. On spatially structured tasks, Sudoku, image-based Sudoku, and maze solving, aNCA sets new state-of-the-art results while using a fraction of the parameters of prior methods. The learned perceptions are directly inspectable, revealing that the model learns task-relevant spatial relations (e.g. Sudoku rows, columns, and boxes) without explicit encoding in the architecture.

Limitations. While aNCA achieves strong results on spatially structured tasks. However, its adaptive perception comes at higher per-step computational cost than fixed-convolution baselines. Detailed resource measurements are provided in section A.3.3. The current architecture targets regular 2D grids. Adapting its learned perception mechanism to irregular graphs, 3D scenes, and variable-resolution inputs is a promising direction for future work.

Funding Acknowledgement

The authors acknowledge the financial support by the German Federal Ministry of Research, Technology and Space (BMFTR) in the program “Forschung an Fachhochschulen in Kooperation mit Unternehmen (FH-Kooperativ)” within the joint project "KI-Bohrer" under grant 13FH525KX1.

AI use statement

In this work, we used generative AI tools as a coding assistant and for language editing, to revise and shorten passages of the manuscript that we had drafted ourselves. Generative AI tools were not used to generate research ideas, to design the proposed architecture, to produce or interpret experimental results, or to retrieve or summarize related work. All code was reviewed and tested by the authors, and every reported number was produced by running that code on the checkpoints described in section A.2. We have reviewed all AI-assisted text and take full responsibility for the final content of this work, including all text, claims and artifacts.

Ethics statement

This work studies architectures for 2D spatial reasoning on synthetic and publicly available benchmarks. It involves no human subjects, no personally identifiable information, and no user data. All datasets used are public and were employed within their intended scope: the Sudoku and MNIST-Sudoku data of [40] and the maze dataset of [38]. We release no new dataset. The proposed models are small and were trained on a single consumer GPU, so the computational footprint of this work is modest. We see no direct pathway from puzzle solving to harmful application, although improved spatial reasoning is a general capability whose downstream uses we cannot fully anticipate. The authors declare no conflicts of interest.

Reproducibility statement

All datasets and benchmarks used in this work are publicly available and are cited in section 3. Model and training hyperparameters for each task are given in sections A.3.1, A.5.1 and A.6.1, the exact training objectives in sections A.3.2, A.5.2 and A.6.2, and the encoder-decoder codec used for MNIST Sudoku in sections A.4.1 and A.4.2, together with an isolation evaluation of its accuracy in section A.4.3. Computational cost and hardware are reported in section A.3.3 and section A.2. All reported results are means and standard deviations over multiple independent training runs rather than single runs, with per-seed behavior discussed in section A.3.4. The full source code, including the configuration files reproducing every reported run, will be released upon publication as a plugin for the *NCATorch* framework of [35].

References

[1] Y Bouligand. Fibroblasts, morphogenesis and cellular automata. In *Disordered Systems and Biological Organization*, pages 367–379. Springer, 1986.

[2] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14455–14465, 2024.

[3] Matthew Cook et al. Universality in elementary cellular automata. *Complex systems*, 15(1):1–40, 2004.

[4] Stephen Coombes. The geometry and pigmentation of seashells. *Nottingham: Department of Mathematical Sciences, University of Nottingham*, 2009.

[5] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks, 2017.

[6] Jingtao Ding, Yunke Zhang, Yu Shang, Yuheng Zhang, Zefang Zong, Jie Feng, Yuan Yuan, Hongyuan Su, Nian Li, Nicholas Sukiennik, et al. Understanding world or predicting future? a comprehensive survey of world models. *ACM Computing Surveys*, 58(3):1–38, 2025.

[7] Qihang Fan, Huaibo Huang, Mingrui Chen, Hongmin Liu, and Ran He. Rmt: Retentive networks meet vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5641–5651, 2024.

[8] Martin Gardner. Mathematical games - the fantastic combinations of john conway's new solitaire game "life". *Scientific American*, 223(4):4, 1970.

[9] Martin Gerhardt and Heike Schuster. A cellular automaton describing the formation of spatially ordered structures in chemical systems. *Physica D: Nonlinear Phenomena*, 36(3):209–221, 1989.

[10] William Gilpin. Cellular automata as convolutional neural networks. *Physical Review E*, 100(3):032402, 2019.

[11] Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Learning graph cellular automata. *Advances in Neural Information Processing Systems*, 34:20983–20994, 2021.

[12] Haralampos Hatzikirou, David Basanta, Matthias Simon, Karl Schaller, and Andreas Deutsch. 'go or grow': the key to the emergence of invasion in tumour progression? *Mathematical medicine and biology: a journal of the IMA*, 29(1):49–65, 2012.

[13] Xinmiao Huang, Qisong He, Zhenglin Huang, Boxuan Wang, Zhuoyun Li, Guangliang Cheng, Yi Dong, and Xiaowei Huang. Spatial-dise: A unified benchmark for evaluating spatial reasoning in vision-language models. *arXiv preprint arXiv:2510.13394*, 2025.

[14] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019.

[15] Mengdi Jia, Zekun Qi, Shaochen Zhang, Wenyao Zhang, Xinqiang Yu, Jiawei He, He Wang, and Li Yi. Omnispatial: Towards comprehensive spatial reasoning benchmark for vision language models. *arXiv preprint arXiv:2506.03135*, 2025.

[16] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017.

[17] Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks, 2025.

[18] John Kalkhof, Arlene Kühn, Yannik Frisch, and Anirban Mukhopadhyay. Frequency-time diffusion with neural cellular automata. *arXiv preprint arXiv:2401.06291*, 2024.

[19] Amita Kamath, Jack Hessel, and Kai-Wei Chang. What's "up" with vision-language models? investigating their struggle with spatial reasoning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9161–9175, 2023.

[20] Kenji Kubo, Shunsuke Kamiya, Masanori Koyama, Kohei Hayashi, Yusuke Iwasawa, and Yutaka Matsuo. C-voting: Confidence-based test-time voting without explicit energy functions. In *The Fourteenth International Conference on Learning Representations*, 2026.

[21] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 1998.

[22] Wufei Ma, Haoyu Chen, Guofeng Zhang, Yu-Cheng Chou, Jieneng Chen, Celso de Melo, and Alan Yuille. 3dsrbench: A comprehensive 3d spatial reasoning benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6924–6934, 2025.

[23] Yuxin Mao, Zhen Qin, Jinxing Zhou, Bin Fan, Jing Zhang, Yiran Zhong, and Yuchao Dai. Learning spatial decay for vision transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 7945–7953, 2026.

[24] Roshanak Mirzaee, Hossein Rajaby Faghihi, Qiang Ning, and Parisa Kordjamshidi. Spartqa: A textual question answering benchmark for spatial reasoning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4582–4598, 2021.

[25] Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson, and Michael Levin. Growing neural cellular automata. *Distill*, 2020. https://distill.pub/2020/growing-ca.

[26] Maximilian Otte, Quentin Delfosse, Johannes Czech, and Kristian Kersting. Generative adversarial neural cellular automata. *arXiv preprint arXiv:2108.04328*, 2021.

[27] Ehsan Pajouheshgar, Yitao Xu, Tong Zhang, and Sabine Süsstrunk. Dynca: Real-time dynamic texture synthesis using neural cellular automata. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20742–20751, 2023.

[28] Rasmus Berg Palm, Miguel González-Duque, Shyam Sudhakaran, and Sebastian Risi. Variational neural cellular automata. *arXiv preprint arXiv:2201.12360*, 2022.

[29] Quang PM Pham, Khoi TN Nguyen, Lan C Ngo, Truong Do, Dezhen Song, and Truong-Son Hy. Tesgnn: Temporal equivariant scene graph neural networks for efficient and robust multi-view 3d scene understanding. *arXiv preprint arXiv:2411.10509*, 2024.

[30] R. C. Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957.

[31] Mark Sandler, Andrey Zhmoginov, Liangcheng Luo, Alexander Mordvintsev, Ettore Randazzo, et al. Image segmentation via cellular automata. *arXiv preprint arXiv:2008.04965*, 2020.

[32] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30, 2017.

[33] Joel L Schiff. *Cellular automata: a discrete view of the world*. John Wiley & Sons, 2011.

[34] Zhengxiang Shi, Qiang Zhang, and Aldo Lipani. Stepgame: A new benchmark for robust multi-hop spatial reasoning in texts. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 11321–11329, 2022.

[35] Martin Spitznagel and Janis Keuper. A new kind of network? review and reference implementation of neural cellular automata, 2026.

[36] Mattie Tesfaldet, Derek Nowrouzezahrai, and Chris Pal. Attention-based neural cellular automata. *Advances in Neural Information Processing Systems*, 35:8174–8186, 2022.

[37] John Von Neumann, Arthur Walter Burks, et al. Theory of self-reproducing automata, 1966.

[38] Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model, 2025.

[39] Wenqi Wang, Reuben Tan, Pengyue Zhu, Jianwei Yang, Zhengyuan Yang, Lijuan Wang, Andrey Kolobov, Jianfeng Gao, and Boqing Gong. Site: towards spatial intelligence thorough evaluation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9058–9069, 2025.

[40] Christopher Wewer, Bartlomiej Pogodzinski, Bernt Schiele, and Jan Eric Lenssen. Spatial reasoning with denoising models. In *International Conference on Machine Learning (ICML)*, 2025.

[41] Stephen Wolfram. Statistical mechanics of cellular automata. *Reviews of modern physics*, 55(3):601, 1983.

[42] Stephen Wolfram and M Gad-el Hak. A new kind of science. *Appl. Mech. Rev.*, 56(2):B18–B19, 2003.

[43] Yuechen Xie, Xiaoyan Zhang, Yicheng Shan, Hao Zhu, Rui Tang, Rong Wei, Mingli Song, Yuanyu Wan, and Jie Song. Spatialqa: A benchmark for evaluating spatial logical reasoning in vision-language models. *arXiv preprint arXiv:2602.20901*, 2026.

[44] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions, 2016.

[45] Magdalena Załuska-Kotur, Hristina Popova, and Vesselin Tonchev. Step bunches, nanowires and other vicinal "creatures"—ehrlich–schwoebel effect by cellular automata. *Crystals*, 11(9):1135, 2021.

[46] Lizong Zhang, Haojun Yin, Bei Hui, Sijuan Liu, and Wei Zhang. Knowledge-based scene graph generation with visual contextual dependency. *Mathematics*, 10(14):2525, 2022.

[47] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. *CoRR*, abs/1811.11168, 2018.

A Technical appendices and supplementary material

Contents

A.1 Video Material 14
A.2 Implementation Details 15
A.2.1 Difference between DCNv1 and DCNv2 15
A.3 Sudoku 16
A.3.1 Model and Training Hyperparameters 16
A.3.2 Training Loss 16
A.3.3 Computational Cost 17
A.3.4 Additional Results: Sudoku 17
A.4 MNIST Sudoku 19
A.4.1 Codec Hyperparameters 19
A.4.2 Codec Training Loss 20
A.4.3 Codec Accuracy 20
A.4.4 Additional Results: MNIST Sudoku 21
A.5 Maze 22
A.5.1 Model and Training Hyperparameters 22
A.5.2 Training Loss 22
A.5.3 Additional Results: Mazes 23
A.6 Global Color Balance 24
A.6.1 Model and Training Hyperparameters 24
A.6.2 Training Loss 24
A.6.3 Additional Results: Global Color Balance 25

A.1 Video Material

`https://youtu.be/WF0Bm6Ie7WA`

- 0:00 Sudoku: aNCA Solving (Batch)
- 0:16 Sudoku: aNCA Solving (Hard)
- 0:47 Sudoku: aNCA Output Diversity
- 1:10 Sudoku MNIST: aNCA Solving (Batch)
- 1:47 Sudoku MNIST: aNCA Solving (Hard)
- 2:10 Sudoku MNIST: aNCA Output Diversity (Solutions)
- 2:25 Sudoku MNIST: aNCA Output Diversity (Digit Style)
- 2:42 Sudoku MNIST: aNCA learned spatial relations
- 4:54 Maze 30x30: NCA w. 3x3 Conv.
- 5:07 Maze 30x30: aNCA
- 5:20 Maze DFS 20x20 (OOD): aNCA
- 5:37 Maze Prim 40x40 (OOD): aNCA
- 5:45 Color Balance Task Perception Comparison

A.2 Implementation Details

All models are implemented in PyTorch. Deformable convolutions use `torchvision.ops.deform_conv2d`, the standard torchvision DCNv2 implementation. All experiments were run on a single NVIDIA RTX 4090. To provide reproducibility, the code will be released upon publication, as a plugin for the *NCATorch* framework of [35].

A.2.1 Difference between DCNv1 and DCNv2

Both variants predict $2k^2$ offsets from the current cell state, yielding two displacement coordinates for each of the k^2 kernel taps. DCNv1 then samples only at the displaced positions:

$$\mathcal{P}_\phi^{\mathrm{DCNv1}}(s, \mathbf{p}) = \sum_{n=1}^{k^2} w_n \cdot s(\mathbf{p} + \mathbf{p}_n + \Delta\mathbf{p}_n(\mathbf{p}))\,. \tag{5}$$

DCNv2 additionally predicts one content-dependent modulation value per tap, requiring a further k^2 outputs, and weights every sampled feature before aggregation:

$$\mathcal{P}_\phi^{\mathrm{DCNv2}}(s, \mathbf{p}) = \sum_{n=1}^{k^2} w_n \cdot m_n(\mathbf{p}) \cdot s(\mathbf{p} + \mathbf{p}_n + \Delta\mathbf{p}_n(\mathbf{p}))\,. \tag{6}$$

Thus, DCNv1 adapts only *where* each tap samples, whereas DCNv2 also adapts *how strongly* each sampled location contributes. In our implementation, $m_n(\mathbf{p}) = 2\,\sigma(a_n(\mathbf{p})) \in [0, 2]$, where a_n is produced by a separate modulation convolution. The offset and modulation predictors are zero-initialized, so the module initially behaves like a regular convolution: offsets are zero and modulation values are one.

A.3 Sudoku

A.3.1 Model and Training Hyperparameters

Table 5: Hyperparameters shared across all NCA variants for the Sudoku task. Perception-specific settings (kernel size, dilation, number of deformable heads) vary per variant and are described in section 2.

Category	Hyperparameter	Value
NCA state	Cell channels	24
	Positional embeddings	yes
Update network	Architecture	MLP (1×1 Conv), 1 hidden layer, 128 channels
Optimization	Optimizer	AdamW
	Learning rate	4×10^{-4}
	LR schedule	WSD (warmup–stable–decay)
	Warmup / stable / decay	10% / 80% / 10%
	Final LR	0
	Gradient clip norm	1.0
Training	Total steps	100 000
	Batch size	128
	Rollout steps T	$\mathcal{U}[18,\ 24]$
Evaluation	Inference rollout steps	1 500
	Digit prediction	argmax over 9 channels (no post-processing)
Sample pool	Pool size	512
	Pool fraction	0.5

A.3.2 Training Loss

Let $p_{i,d} = \mathrm{softmax}(\hat{s}_i)_d$ be the predicted probability of digit $d \in \{1, \ldots, 9\}$ at cell i. A valid Sudoku completion requires each digit to appear exactly once in every row, column, and 3×3 box. Denoting the set of all 27 such units as $\mathcal{U}$, we penalize squared deviations from the per-unit digit count of one:

$$\mathcal{L}_{\text{constraint}} = \frac{1}{|\mathcal{U}|} \sum_{U \in \mathcal{U}} \sum_{d=1}^{9} \left(\sum_{i \in U} p_{i,d} - 1 \right)^2 . \tag{7}$$

An additional cross-entropy term enforces consistency with the given clue cells $\mathcal{C}$ whose correct digit is y_i:

$$\mathcal{L}_{\text{given}} = -\frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} \log p_{i,\, y_i}. \tag{8}$$

The total loss is $\mathcal{L} = \mathcal{L}_{\text{constraint}} + \mathcal{L}_{\text{given}}$.

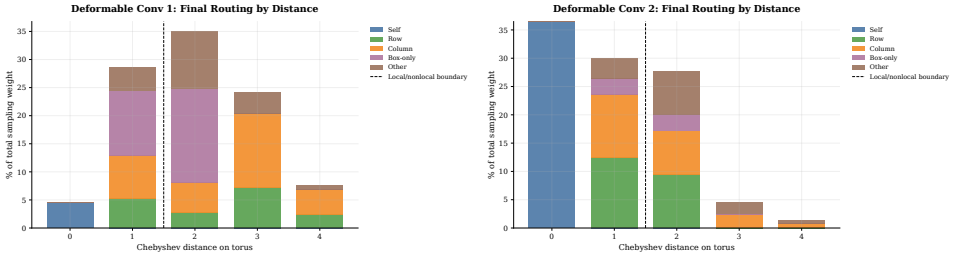
A.3.3 Computational Cost

Table 6: Resource usage of the Sudoku NCA variants. Training VRAM is the peak CUDA memory allocated during a representative training step (batch size 128, rollout length 24). Inference VRAM and runtime are measured at batch size 1 with a rollout of 1500 steps on an NVIDIA RTX 4090. Inference time is averaged per puzzle over repeated runs after warm-up.

Variant	Params	Train VRAM (MB)	Infer VRAM (MB)	Infer ms / puzzle
NCA w. 3x3 Conv.	50k	303.1	0.5	159.64
NCA w. 5x5 Conv.	105k	311.9	0.9	166.38
NCA w. R/C/B Conv.	145k	610.3	1.0	251.04
NCA w. Dilated Perc.	150k	480.8	1.2	238.28
aNCA	165k	1139.6	17.7	832.75

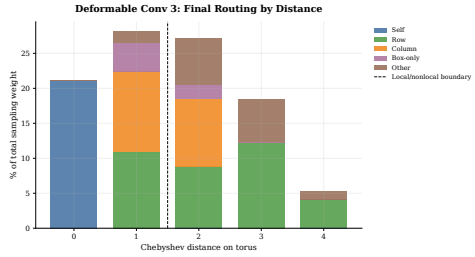
A.3.4 Additional Results: Sudoku

This section supplements the main array-Sudoku evaluation with further analysis of aNCA's adaptive perception and reproducibility. We visualize the sampling-distance distributions of the deformable heads, quantify their alignment with Sudoku's row, column, and box constraints, and report the variation across independent training seeds.



(a) A

(b) B



(c) C

Figure 10: Learned sampling distance distributions of the three deformable perception heads (A, B, C) at the final NCA step, aggregated over held-out test puzzles. Each plot shows how far each kernel tap reaches from the source cell, revealing that the heads develop complementary routing strategies: some taps remain local while others extend to capture row, column, or box constraints.

Quantitative alignment scores. Table 7 reports, for each deformable head, the mean fraction of total sampling weight landing inside the source cell's row, column, 3×3 box, and their union (Sudoku-relevant), averaged over all 81 source cells and 10 000 held-out test puzzles at the final rollout step. The random baseline is the expected fraction under uniform sampling. All three heads concentrate 81–87% of their weight on Sudoku-relevant cells, 3.1–$3.4\times$ above the 25.9% random baseline.

Table 7: Per-head constraint alignment at the final NCA step, averaged over all 81 source cells and 10 000 test puzzles. Union% is the fraction of weight on the source cell's row $\cup$ column $\cup$ box. Random baseline is the expected Union% under uniform sampling (25.9%).

Head	Row%	Col%	Box%	Union%	Enrichment
Head 1	20.7	32.6	47.5	81.0	3.13×
Head 2	61.1	59.6	69.6	86.9	3.35×
Head 3	53.0	45.8	54.3	83.5	3.22×
Random baseline	–	–	–	25.9	1.00×

A.4 MNIST Sudoku

The NCA variants evaluated on MNIST Sudoku are the *exact* checkpoints trained for the array task (section A.3); no additional NCA training is performed. The pixel-space pipeline wraps these frozen NCA checkpoints with a separately trained encoder–decoder codec, described below.

A.4.1 Codec Hyperparameters

Table 8: Architecture and training hyperparameters for the Sudoku codec. Encoder and decoder are trained independently with AdamW.

Category	Hyperparameter	Value
Encoder E_ψ	Base channels	32
	Hidden dim	128
Decoder D_ω	Base channels	64
	Hidden dim	128
	Style latent dim	8
Optimization	Optimizer	AdamW
	Learning rate	1×10^{-3}
	LR schedule	Warmup–cosine (1000 steps)
	Total steps	100 000
Training	Batch size	32

A.4.2 Codec Training Loss

The decoder D_ω is trained in isolation with a composite loss. Let $\hat{x} = D_\omega(b, z)$ be the rendered image for board b with style latent $z \sim q_\omega(z \mid b)$, x^* the target pixel image, and M_{pix} a pixel-space binary mask over occupied (non-blank) cells. The total loss is

$$\mathcal{L}_{\text{dec}} = w_{\text{KL}}\,\mathcal{L}_{\text{KL}} + w_{\text{pix}}\,\mathcal{L}_{\text{pix}} + w_{\text{feat}}\,\mathcal{L}_{\text{feat}} + w_{\text{cls}}\,\mathcal{L}_{\text{cls}} + w_{\text{TV}}\,\mathcal{L}_{\text{TV}} + w_{\text{ink}}\,\mathcal{L}_{\text{ink}} + w_{\text{bin}}\,\mathcal{L}_{\text{bin}} \tag{9}$$

with weights $(w_{\text{KL}}, w_{\text{pix}}, w_{\text{feat}}, w_{\text{cls}}, w_{\text{TV}}, w_{\text{ink}}, w_{\text{bin}}) = (0.001, 1.0, 0.5, 1.0, 0.01, 0.0025, 0.01)$.

KL divergence. $\mathcal{L}_{\text{KL}} = D_{\text{KL}}(q_\omega(z \mid b) \,\|\, \mathcal{N}(0, I))$, the standard VAE term regularising the style posterior.

Pixel L1. $\mathcal{L}_{\text{pix}} = \sum_i |\hat{x}_i - x_i^*| \cdot M_{\text{pix},i} \,/\, \sum_i M_{\text{pix},i}$, a masked per-pixel reconstruction loss.

Feature L1. Let $\phi(\cdot)$ denote the pooled intermediate features of the frozen digit classifier E_ψ, and let $\hat{c}, c^*$ be the rendered and target cell crops. $\mathcal{L}_{\text{feat}} = \|\phi(\hat{c}) - \phi(c^*)\|_1$ is a perceptual term encouraging feature-level similarity to the target digit appearance.

Classifier CE. $\mathcal{L}_{\text{cls}} = \text{CE}(E_\psi(\hat{c}),\, d^*)$, cross-entropy under the frozen classifier, directly rewarding legible digit rendering.

Total variation. Let $\text{ink} = (1 - \hat{x}) \cdot M_{\text{pix}}$. Then $\mathcal{L}_{\text{TV}} = \text{mean}(|\Delta_h\,\text{ink}| + |\Delta_w\,\text{ink}|)$ encourages smooth, connected strokes.

Ink sparsity. $\mathcal{L}_{\text{ink}} = \sum_i (1 - \hat{x}_i)\, M_{\text{pix},i} \,/\, \sum_i M_{\text{pix},i}$ penalises dark pixels outside digit regions.

Binary commitment. $\mathcal{L}_{\text{bin}} = \sum_i \hat{x}_i(1 - \hat{x}_i)\, M_{\text{pix},i} \,/\, \sum_i M_{\text{pix},i}$ penalises intermediate grey values, pushing pixels toward binary black/white.

The encoder E_ψ is trained separately with a masked cross-entropy loss on the given (clue) cells $\mathcal{C}$:

$$\mathcal{L}_{\text{enc}} = -\frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} \log p_{i,\, y_i}, \tag{10}$$

where $p_{i,d}$ is the predicted probability of digit d at cell i and y_i is the ground-truth digit.

A.4.3 Codec Accuracy

We evaluate the encoder E_ψ and decoder D_ω in isolation on 10 000 test puzzles across the full difficulty range to quantify how much of the pipeline error originates from the codec versus the NCA.

Encoder. Running E_ψ on the input pixel Sudoku and taking the argmax over digit logits yields 99.38% per-cell accuracy on the given (non-blank) cells. The perfect-board rate (the fraction of boards where every given cell is correctly recognized) is 74.58%, meaning roughly one in four boards contains at least one misrecognised clue cell. Despite these imperfect inputs, aNCA maintains strong downstream solving performance (table 9) without additional training, highlighting its ability to operate within noisy encoder conditioning.

Decoder round-trip. We pass the ground-truth one-hot board through D_ω, decode the resulting pixel image with E_ψ, and compare the re-encoded digits to the ground truth. Per-cell accuracy is 100% and the perfect round-trip rate is 100%: the decoder renders digits reliably enough that the encoder always recovers the correct symbol. Decoder errors therefore contribute zero to the pipeline failure rate. ASR is evaluated against ground-truth clue digits, so encoder misrecognitions affect NCA conditioning but not the validity metric.

A.4.4 Additional Results: MNIST Sudoku

This section supplements the main MNIST Sudoku evaluation with complete per-bucket results. In addition to ASR, we report ACP for every NCA perception variant on the Easy, Medium, and Hard splits, using the same frozen codec and array-trained NCA checkpoints as in the main evaluation.

Table 9: Sudoku Images

Method	Source	Easy [1, 27]		Medium [28, 54]		Hard [55, 81]		Param.
		ASR	ACP	ASR	ACP	ASR	ACP	
Related Work								
Diffusion Model	[40]	99.4%	$-^{\star}$	53.6%	$-^{\star}$	0.8%	$-^{\star}$	118M
SRM	[40]	99.8%	$-^{\star}$	75.4%	$-^{\star}$	51.6%	$-^{\star}$	118M
NCAs								
NCA w. 3x3 Conv.	Ours	99.40 ± 0.14	99.90 ± 0.00	80.90 ± 0.90	98.33 ± 0.10	71.98 ± 1.61	98.20 ± 0.08	1.1M + 50k
NCA w. 5x5 Conv.	Ours	99.13 ± 0.50	99.87 ± 0.06	79.97 ± 1.70	98.17 ± 0.15	69.00 ± 6.63	95.77 ± 4.13	1.1M + 105k
NCA w. 7x7 Conv.	Ours	98.73 ± 0.10	99.83 ± 0.05	76.55 ± 1.21	97.73 ± 0.17	72.30 ± 2.60	98.15 ± 0.19	1.1M + 189k
NCA w. 9x9 Conv.	Ours	99.30 ± 0.23	99.93 ± 0.05	66.73 ± 16.99	95.18 ± 4.42	35.83 ± 33.56	84.25 ± 20.15	1.1M + 300k
NCA w. R/C/B Conv.	Ours	99.20 ± 0.08	99.90 ± 0.00	80.38 ± 1.41	98.13 ± 0.17	71.25 ± 2.61	98.13 ± 0.17	1.1M + 145k
NCA w. Dilated Perc.	Ours	99.25 ± 0.17	99.88 ± 0.05	80.78 ± 0.95	98.13 ± 0.05	70.95 ± 8.39	98.05 ± 0.57	1.1M + 150k
aNCA	Ours	99.00 ± 0.37	99.88 ± 0.05	$\mathbf{81.90 \pm 5.50}$	$\mathbf{98.40 \pm 0.50}$	$\mathbf{82.10 \pm 4.89}$	$\mathbf{98.85 \pm 0.40}$	1.1M + 165k

* indicates metrics not reported in prior work.

A.5 Maze

A.5.1 Model and Training Hyperparameters

Table 10: Hyperparameters shared across all NCA variants for the maze task. Perception-specific settings vary per variant and are described in section 2.

Category	Hyperparameter	Value
NCA state	Cell channels	24
	Positional embeddings	no
	Fire rate	0.5
Update network	Architecture	MLP (1×1 Conv), 1 hidden layer, 128 channels
Optimization	Optimizer	AdamW
	Learning rate	1×10^{-3}
	Weight decay	1×10^{-4}
	LR schedule	WSD (warmup–stable–decay)
	Warmup / stable / decay	10% / 80% / 10%
	Gradient clip norm	1.0
Training	Total steps	100 000
	Batch size	64
	Rollout steps T	$\mathcal{U}[18,\ 24]$
Evaluation	Inference rollout steps	100
	Path prediction	threshold at 0.5 (no post-processing)
Sample pool	Pool size	512
	Pool fraction	0.5

A.5.2 Training Loss

Let $\hat{y}_i \in [0, 1]$ be the predicted path probability at cell i, $y_i \in \{0, 1\}$ the ground-truth binary path mask, and $\mathcal{F}$ the set of non-wall cells. Walls are excluded since they are trivially predictable from the condition input. The loss combines three terms. A *BCE* term encourages binary commitment to path vs. non-path:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{|\mathcal{F}|} \sum_{i \in \mathcal{F}} \big[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \big]. \tag{11}$$

A *Dice* term directly optimizes path-prediction overlap, handling the imbalance between path and non-path cells:

$$\mathcal{L}_{\text{Dice}} = 1 - \frac{2 \sum_{i \in \mathcal{F}} \hat{y}_i \, y_i + \epsilon}{\sum_{i \in \mathcal{F}} \hat{y}_i + \sum_{i \in \mathcal{F}} y_i + \epsilon}. \tag{12}$$

A *length penalty* penalizes paths that are longer or shorter than the shortest-path target, normalized by the number of free cells:

$$\mathcal{L}_{\text{len}} = \left| \frac{\sum_{i \in \mathcal{F}} \hat{y}_i}{|\mathcal{F}|} - \frac{\sum_{i \in \mathcal{F}} y_i}{|\mathcal{F}|} \right|. \tag{13}$$

The total loss is $\mathcal{L} = \mathcal{L}_{\text{BCE}} + \mathcal{L}_{\text{Dice}} + w_{\text{len}} \, \mathcal{L}_{\text{len}}$ with $w_{\text{len}} = 0.05$.

A.5.3 Additional Results: Mazes

This section supplements the main maze evaluation with qualitative rollout trajectories and a quantitative same-resolution distribution-shift experiment. We evaluate the trained 30×30 aNCA checkpoint, without finetuning, on 1000 unseen Prim-generated mazes and compare its Valid Path and Valid Optimal scores with those on the original Sapient test distribution.

Generalization beyond the training maze distribution. To evaluate whether the trained aNCA learns a reusable path-finding procedure, we run the same checkpoint, without finetuning, on 1000 procedurally generated 30×30 mazes using randomized Prim-style maze generation [30]. Although these mazes differ from the training distribution in wall density and corridor geometry, the NCA propagates a connected path from start to goal and adapts its rollout to the new layout. It achieves 100% Valid Path and 99.4% Valid Optimal, showing robust transfer to a new maze distribution at the same grid size. Qualitative examples and the full comparison are shown in fig. 9b.

Table 11: Generalization of the deformable aNCA maze solver to a different maze distribution at the training resolution. The model is trained on 30×30 Sapient mazes and evaluated without finetuning on 1000 mazes per test distribution. Valid Path checks connectivity and wall avoidance. Valid Optimal additionally requires shortest-path length.

Test Maze Size	Generator	Valid Path	Valid Optimal
30×30	Sapient	98.56%	94.18%
20×20	Prim	100%	100%
30×30	Prim	100%	99.4%

A.6 Global Color Balance

A.6.1 Model and Training Hyperparameters

Table 12: Hyperparameters for the global color balance task.

Category	Hyperparameter	Value
NCA state	Cell channels	24
	Positional embeddings	no
	Fire rate	0.5
Update network	Architecture	MLP (1×1 Conv), 1 hidden layer, 128 channels
Optimization	Optimizer	AdamW
	Learning rate	3×10^{-4}
	LR schedule	WSD (warmup–stable–decay)
	Warmup / stable / decay	10% / 80% / 10%
	Gradient clip norm	1.0
Training	Total steps	25 000
	Batch size	16
	Rollout steps T	$\mathcal{U}[12,\, 24]$
Evaluation	Inference rollout steps	200
	Test samples per run	1000
	Independent training runs	3
Sample pool	Pool size	512
	Pool fraction	0.5
Loss weights	w_{count}	1.0
	w_{sat}	0.01

A.6.2 Training Loss

Let $x_i \in [-1, 1]$ be the predicted pixel value of cell i (where -1 = black, $+1$ = white), N the total number of pixels, and $\mu^* = 2r - 1$ the target mean corresponding to a prescribed white fraction r. The loss combines two terms. A *count term* pushes the global pixel mean toward the target ratio:

$$\mathcal{L}_{\text{count}} = \left(\frac{1}{N} \sum_{i=1}^{N} x_i - \mu^* \right)^2 . \tag{14}$$

A *saturation term* pushes each pixel to commit to either -1 or $+1$, zeroing only at fully binary values:

$$\mathcal{L}_{\text{sat}} = \frac{1}{N} \sum_{i=1}^{N} \left(1 - x_i^2\right) . \tag{15}$$

The total loss is $\mathcal{L} = \mathcal{L}_{\text{count}} + \mathcal{L}_{\text{sat}}$.

A.6.3 Additional Results: Global Color Balance

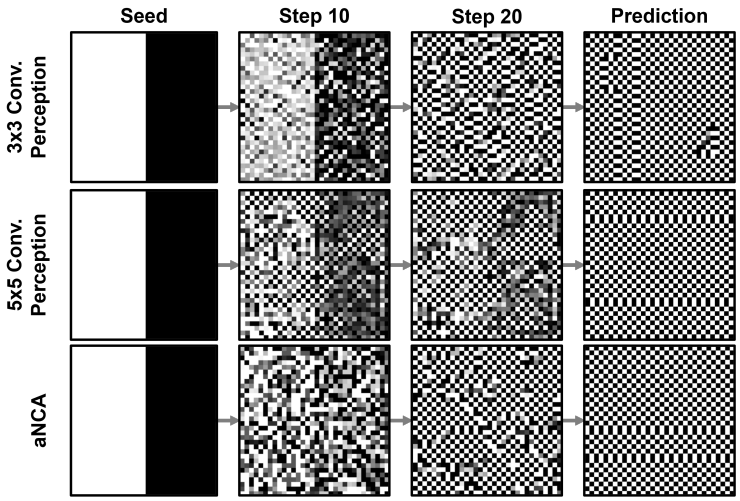


Figure 11: Global color balance: the model must satisfy a prescribed black/white ratio without per-pixel supervision.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction correctly describe the main contributions, which are substantiated by the experimental results in Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A dedicated Limitations paragraph in section 6 discusses computational overhead of deformable convolutions, generalization degradation to larger spatial domains, information bottlenecks in pipeline architectures, and the restriction to 2D grid-structured data.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper makes no theoretical claims requiring formal proofs; all contributions are empirical.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The architecture, training protocol, loss functions, and datasets are described in section 2 and section 3. Exact hyperparameters (learning rate, batch size, rollout range) are provided in sections A.3.1 and A.5.1.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All datasets are from cited prior work. Code is not yet publicly released; we plan to release it upon acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: raining and evaluation protocols are described in section 2 and section 3. Full hyperparameter details are provided in sections A.3.1 and A.5.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All reported performance gaps are statistically significant given the evaluation sample sizes. Each Sudoku difficulty bucket is evaluated on the full 10 000 held-out puzzles (95% Wilson CI $\approx \pm 0.9\%$); the smallest Hard-bucket gap between methods exceeds 20 percentage points. Maze and color-balance models are evaluated on 1 000 held-out samples (95% CI $\approx \pm 2.5\%$); the aNCA–TRM-Att gap on maze is 9.9 percentage points, and color-balance gaps exceed 40 percentage points. For the primary model (aNCA on Sudoku), a 5-seed reproducibility study additionally reports mean Hard ASR 74.7 ± 2.0 (std, n=3 converged seeds) in section A.3.4.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Justification: VRAM and inference time per puzzle are reported in section A.3.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work involves no human subjects, no sensitive data, and poses no foreseeable ethical risks.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: This work is foundational research on spatial reasoning applied to abstract puzzles. There is no direct path to negative societal applications.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The proposed models pose no meaningful misuse risk as they solve abstract spatial puzzles.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets are from cited prior work. MNIST is publicly available; the Sudoku and maze datasets are used under their respective licenses as described in the cited works.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: No new datasets are released. Model checkpoints and code will be made available upon acceptance.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing or research with human subjects was conducted.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not contain any experiments with human subjects

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are not used as a component of the proposed method. Any LLM usage was limited to writing assistance and does not affect the methodology or results.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.