Transformers Linearly Represent Constraint-based World Models in Sudoku

Roman Kniazev
LaBRI, CNRS
University of Bordeaux, France
roman@knzv.me

Nathanaël Fijalkow
LaBRI, CNRS
University of Bordeaux, France
nathanael.fijalkow@gmail.com

Abstract

Do transformers, when trained on sequential reasoning traces, build internal models of the underlying task? And if so, does the structure of those internal representations mirror the structure of the domain?

We train an 8-layer transformer on Sudoku solving traces and perform a mechanistic analysis of its internal computation. We find a linearly represented *substructure world model*: it does not represent the board state cell by cell, as a human analyst would expect, but organizes information around the rows, columns, and boxes that Sudoku's constraints act on. We see that mid-layer attention heads organize around Sudoku's substructures, suppressing placement logits for digits already present and promoting those for absent digits. We identify a *naked-single circuit*: a small set of dedicated neurons in the final MLP layer, each individually detecting when exactly one digit remains possible for a specific cell, and reliably promoting that digit.

These findings suggest that transformers trained on solving traces may encode constraint-relevant information more cleanly than individual surface facts. For interpretability, probing only the obvious surface decomposition of a task may therefore underestimate what the model has learned.

1 Introduction

Context. Do transformers, when trained on sequential reasoning traces, build internal *world models* of the underlying environment? Li et al. [2023] and Nanda et al. [2023] showed that the answer is yes for Othello: a GPT-2-style model trained only on game transcripts spontaneously develops a representation of board occupancy (black / white / empty per cell), despite receiving no direct supervision on board state. This finding sits within the programme of mechanistic interpretability Elhage et al. [2021, 2022], which aims to reverse-engineer the algorithms learned by neural networks. A central working hypothesis in this programme is the linear representation hypothesis Park et al. [2024], Mikolov et al. [2013]: that high-level concepts are encoded as linear directions in activation space, and can therefore be read out by simple linear probes Alain and Bengio [2016], Belinkov [2022]. Understanding when and how world models emerge under this hypothesis, and what geometry they take, is important for building reliable, inspectable AI systems.

Background: linear world models and how to find them. Nanda et al. [2023] built on Li et al. [2023]'s result by establishing that the Othello world model is *linear*: the occupancy of each cell is recoverable by a simple logistic regression *probe* on the residual stream, without any nonlinear transformation. They further showed, using *activation patching*, that these linear directions are causally used by the model to predict the next move, not merely a side effect of training. The same paradigm has since revealed world models in chess-playing transformers [Karvonen, 2024] and maze-solving models [Ivanitskiy et al., 2024]. In all of these cases, the unit of representation matches the most natural human decomposition of the state space: one feature per cell.

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

Our question: does the geometry reflect the task's constraint algebra? We ask whether this phenomenon extends to a qualitatively different class of task: combinatorial *constraint-satisfaction*. Our domain is **Sudoku**: a one-player puzzle on a 9×9 grid whose fundamental difficulty is constraint propagation across 27 overlapping *substructures* (9 rows, 9 columns, 9 boxes), each requiring every digit 1–9 to appear exactly once. Unlike Othello, where the state is naturally decomposed by cell, Sudoku has an explicit algebraic structure: the validity of any placement is determined by the digit's presence in three substructures, not in any single cell. This raises a sharp question: does the model represent the board state *per cell* (as in Othello) or *per substructure* (as the constraints demand)?

Following Giannoulis et al. [2026] and Shah et al. [2024], we train a causal transformer on Norvig-style solving traces with backtracking [Norvig, 2006], achieving 98.4% per-cell and 97.5% per-grid accuracy on a held-out test set of 150,000 puzzles. We then apply Nanda et al.'s probe-and-patch methodology to the residual stream.

Findings and contributions. Our contribution is new findings about the geometry of emergent world models and the circuits that read them, obtained by applying the probe-and-patch methodology of Nanda et al. [2023] to a transformer trained on Sudoku following Giannoulis et al. [2026].

1. **Substructure world model** (Section 3). Prior world-model work on games, such as Othello and chess, has consistently found linear representations organized around the surface structure of the task: one feature per cell. In Sudoku, row, column, and box occupancy defines a constraint-sufficient state: it determines which digits are legal in an empty cell without uniquely determining the contents of every cell. We show that the model linearly represents this constraint-relevant information more cleanly than individual cell contents. Binary linear probes for "is digit d present in substructure S?" achieve perfect exact-match accuracy over each substructure at layers 4–6, while per-cell content probes peak at 0.572 exact match on filled cells. The representation forms by mid-layers and tracks the current board state throughout generation, remaining geometrically stable across the entire solving trace. Causal patching confirms these directions are actively used.
2. **Substructure-aligned attention** (Section 4). We identify that mid-layer attention heads organize along Sudoku's substructures: heads concentrate their attention on individual rows, columns, or boxes (or coherent groups of three), and their OV circuits apply a symmetric mechanism: digits already present in the attended substructure have their logits suppressed, while absent digits are promoted. This shows that the transformer represents Sudoku's rules through localized constraint checkers operating directly on the substructure-organized residual stream.
3. **Naked-single circuit** (Section 5). We identify a small set of dedicated neurons in the final MLP layer that detect when exactly one digit remains possible for a specific cell (a *naked single*). Each neuron is specific and monosemantic. Moreover, the residual stream before the final MLP already strongly promotes the correct digit, so mid-layers encode the answer and the final MLP amplifies it. The circuit is sparse and provides a fully mechanistic account of how the model handles the most common forced placement.

Code is available on GitHub, and the dataset with a model checkpoint on HuggingFace.

2 Preliminaries

Sudoku. A Sudoku puzzle is defined on a 9×9 grid of 81 cells. Each cell (r, c), for $r, c \in \{1, \ldots, 9\}$, is to be filled with a digit $d \in \{1, \ldots, 9\}$. The grid is partitioned into 27 *substructures*: 9 rows, 9 columns, and 9 *boxes*. Row r is $\mathsf{Row}_r = \{(r, c) : c \in \{1, \ldots, 9\}\}$; column c is $\mathsf{Col}_c = \{(r, c) : r \in \{1, \ldots, 9\}\}$; box Box_k for $k = 3(i-1) + j$ ($i, j \in \{1, 2, 3\}$) is the 3×3 block $\{(3i-2, 3j-2), \ldots, (3i, 3j)\}$. Every cell (r, c) belongs to exactly three substructures: Row_r, Col_c, and one box Box_k. Three horizontally adjacent boxes sharing the same rows form a *band*; three vertically adjacent boxes sharing the same columns form a *stack*. There are 3 bands and 3 stacks, each covering 27 cells.

A placement of digit d in cell (r, c) is *valid* if d does not already appear in Row_r, Col_c, or Box_k. The *candidate set* $\mathsf{cand}(r, c) = \{1, \ldots, 9\} \setminus (\text{digits in } \mathsf{Row}_r \cup \mathsf{Col}_c \cup \mathsf{Box}_k)$ collects all valid digits for an

empty cell. A puzzle is given as an initial partial assignment (the *clues*), and the task is to complete it uniquely so that every digit appears exactly once per substructure.

Solving techniques. Two basic deductive rules eliminate candidates before backtracking search is needed. Cell (r, c) is a *naked single* if $|\texttt{cand}(r, c)| = 1$; the unique candidate must be placed there. Digit d is a *hidden single* in substructure S if exactly one empty cell $(r, c) \in S$ has $d \in \texttt{cand}(r, c)$; that cell must receive d. When neither rule applies, a *backtracking search* is used: the solver selects the cell with the fewest candidates, guesses a digit, and rolls back if a contradiction is reached.

Solving traces and model vocabulary. Following Giannoulis et al. [2026] and Shah et al. [2024], we generate solving traces using a Norvig-style solver [Norvig, 2006] with several randomizations (order of constrained placements, backtracking cell selection, digit order), producing diverse traces per puzzle. The vocabulary consists of **729 placement tokens** $[\texttt{R}r\texttt{C}c\texttt{=}d]$ for each cell and digit, plus special tokens `[clues_end]` (end of clues), `[push]` (backtracking branch start), `[pop]` (failed branch), `[success]` (puzzle solved), and `[pad]`. A typical trace takes the form:

$$\underbrace{[\texttt{R1C1} = \texttt{5}] \cdots [\texttt{R2C4} = \texttt{3}]}_{\text{clues}}\ \texttt{[clues_end]}\ \underbrace{[\texttt{R3C5} = \texttt{9}]\,[\texttt{R6C8} = \texttt{4}]}_{\text{deductions}}\ \texttt{[push]}\ \cdots$$
$$\cdots [\texttt{R8C8} = \texttt{1}]\ \texttt{[pop]}\ \texttt{[push]}\ [\texttt{R8C8} = \texttt{2}]\ \cdots\ \texttt{[success]}.$$

The `[clues_end]` token serves as the primary anchor for our probing experiments: it follows all clue tokens and precedes any model-generated placement, providing a clean position at which to measure the model's representation of the initial board state.

Model architecture and training. We train a standard causal decoder-only transformer following Giannoulis et al. [2026]: $L = 8$ layers, $H = 8$ attention heads, pre-layer normalization, $d_{\text{model}} = 576$, $d_{\text{mlp}} = 3{,}456$ (expansion ratio $6\times$). The model is trained on 2.7M puzzles from the sudoku-3m dataset [Radcliffe, 2020], with traces trimmed to 250 tokens (only 0.9% are longer; see Section A.2), for 6 epochs on a Google TPU v6e, achieving **98.4% per-cell** and **97.5% per-grid** accuracy on a held-out test set of 150K puzzles.

Linear probing. A *linear probe* [Alain and Bengio, 2016] for a binary concept f is a logistic regression classifier trained on residual stream activations $x \in \mathbb{R}^{d_{\text{model}}}$:

$$P(f = 1 \mid x) = \sigma(w_f \cdot x + b_f).$$

The probe direction $w_f \in \mathbb{R}^{d_{\text{model}}}$ recovers a linearly accessible encoding of f in the model's internal representation. A near-perfect probe (AUC ≈ 1) certifies that the concept is linearly encoded; an imperfect one does not rule out encoding at a different level of abstraction. Unless otherwise stated, probes are trained on 5,120 puzzles from the test set and evaluated on 1,280, probing activations at the `[clues_end]` position after each layer.

Causal intervention. Linear probes identify *where* information is encoded but not whether the model *uses* it. Activation patching [Geiger et al., 2021] answers this causal question: we subtract a scaled probe direction from the residual stream at a given layer and measure the change in output logits. If the intervention disrupts the correct prediction, the direction is causally involved in the computation, not merely a side effect of training.

Direct logit attribution and the logit lens. To track how predictions build up across layers, we use *direct logit attribution* [Elhage et al., 2021] and the *logit lens* [nostalgebraist, 2020]. The logit lens projects the residual stream after each layer through the final unembedding matrix, reading off the model's "current best guess" at every intermediate step. Direct logit attribution decomposes the final logit difference into additive contributions from each attention head and MLP layer, identifying which components are responsible for promoting the correct token.

Attention head decomposition. Each attention head can be factored into two independent circuits [Elhage et al., 2021]. The **QK circuit** (query–key matrices) determines *which* positions the head attends to. The **OV circuit** (output–value matrices) determines *what* information is read from those positions and written into the residual stream. Analyzing them separately lets us ask: *what does the head look at?* (QK) and *what does it do with what it sees?* (OV).

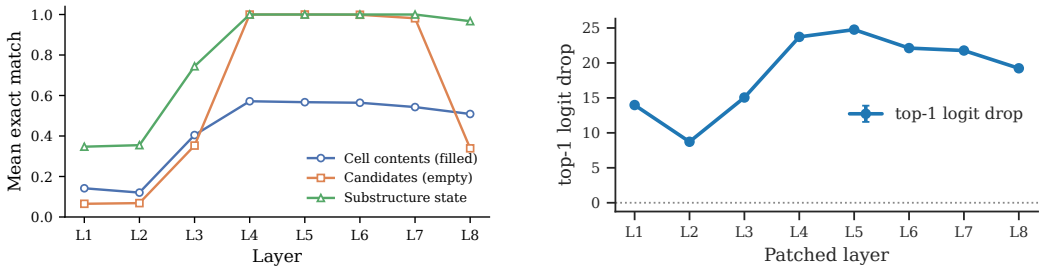


Figure 1: **(Left)** Mean exact match accuracy of three families of binary linear probes trained at the `[clues_end]` token over different layers. *Blue*: 729 probes predicting if a digit occupies a cell, trained and evaluated only on filled cells (exact match across all nine predictions for a cell); *Orange*: 729 probes predicting if a digit is a valid candidate in a cell, trained and evaluated only on empty cells (exact match across the cell's candidate set); *Green*: 243 probes predicting if a digit is present in a substructure (exact match across all nine predictions for a substructure). Scores are averaged across cells or substructures, respectively. **(Right)** Logit drop of the clean model's top-ranked placement token after activation patching of substructure probe directions across layers.

3 Understanding the Latent Space

3.1 Cell-Level Representation

Prior studies into world representation in sequence models trained on grid-based environments, such as OthelloGPT (Nanda et al. [2023], Li et al. [2023]), have based the analysis on cell-based features, as it is the smallest feature scale of the environment. Given that Sudoku is presented visually and structurally as a 9×9 grid, the natural baseline hypothesis is that the network learns a similar spatial decomposition, representing the state of the board via independent, cell-local features.

Are the contents of cells linearly represented? To test this spatial hypothesis, we apply linear probing at the cell level. We train 729 independent binary linear probes, one per $(\text{cell}, \text{digit})$ pair, on the residual stream activations at the `[clues_end]` token. Each probe predicts whether its digit occupies its cell and is trained and evaluated only on puzzles in which that cell is filled. The probes are trained independently after each of eight layers.

As the main performance measure, we use exact-match accuracy: all nine predictions for a cell must be correct. As shown in Fig. 1 (left), accuracy averaged across cells peaks at 0.572 at layer 4 and remains between 0.54 and 0.57 through layers 5–7. We additionally measure mean ROC-AUC across the 729 probes, which peaks at 0.972 at layer 4 (Fig. 6, right). For comparison, we also train 81 nine-class linear probes, one per cell and restricted to filled cells, which reach 0.809 top-1 accuracy at layer 5; their MSE and ROC-AUC are shown in Fig. 6. These results indicate that substantial information about cell contents is linearly recoverable, but the probes do not recover them perfectly.

Are the candidates linearly represented in each cell? In OthelloGPT, the linear representation of the board only emerged once the probe target was reframed from absolute (black/white) to relative (mine/theirs). Following this, we hypothesize that the network primarily works with the candidates instead of filled digits. In this experiment, we maintain the cell as our spatial unit but shift the predicted concept to candidate sets.

We train 729 binary linear probes to predict the valid candidates of each cell, that is, each probe answers "If cell (r, c) is empty, is digit d a valid candidate in it?". In stark contrast to the state probes, these candidate-set probes achieve 1.0 exact match accuracy through layers 4 to 6 when predicting the full 9-digit candidate set for any given cell, as can be seen in Fig. 1 (left).

To check if the 729 cell-candidate probes encode 729 independent features, we extract the normal vectors of the 81 probes for a fixed digit and compute their pairwise cosine similarity at layer 6. We find that the vectors are not independent. Two cells that share a substructure have probes aligned at ≈ 0.33 on average; two cells that share two substructures (e.g. a row and a box, or a column and a box) align at ≈ 0.65. Vectors for cells with no shared substructure are near-orthogonal. Other mid-layers

show the same pattern with greater spread (Section B.2). This structural alignment strongly implies that the model does not maintain 81 isolated candidate sets, but instead the cell-level candidate vectors are linear combinations of substructure-level features.

3.2 Substructure-Level Representation

Are the candidates linearly represented in each substructure? The structural colinearity of the cell-candidate vectors suggests that the network factorizes the board state according to the puzzle rules: whether a digit is a valid candidate in a cell is dictated not by the cell itself, but entirely by absence of that digit in the row, column, or box the cell belongs to.

We test whether the representation uses this algebraic structure by shifting the unit of analysis from the cell to the logical constraint group. We train 243 binary linear probes that predict a global constraint: "Is digit d present in substructure S?" (covering all 9 rows, 9 columns, and 9 boxes for each of the 9 digits). As shown in Fig. 1 (left), these substructure probes achieve 1.0 exact-match accuracy at layers 4–6. Exact match requires all nine digit-presence predictions within a substructure to be correct; we then average over the 27 rows, columns, and boxes. Accuracy remains near-perfect at layer 7 and decreases to 0.967 at layer 8, while cell-candidate exact-match accuracy falls to 0.339.

There is a useful asymmetry here. If a digit is present in any of a cell's row, column, or box, it cannot be a candidate there. Candidacy can therefore be determined by a linear combination of the three digit-presence indicators followed by a threshold. These indicators do not, however, uniquely determine cell contents: a digit being present in all three substructures does not mean that it occupies their shared cell, which may simply be empty. Substructure state is thus sufficient to determine candidate sets without specifying individual cell contents. This asymmetry is consistent with the cleaner decoding of candidate sets than cell contents.

3.3 Causal Ablation

Is the substructure-level representation actually used for prediction? We have shown that substructure directions can be decoded from the residual stream. Are they actually used by the model when it predicts the next placement? We test this by overwriting those directions to encode that the digit is already present in the row, column, and box of a target cell and check whether the model's prediction for that placement collapses.

Concretely, we collect 300 grid states (G1) at `[clues_end]` whose first generated token is a legal placement, together with the following state (G2) in which that placement fills cell (r, c) with digit d. We then patch the residual stream of (G1) at a given layer by transplanting the (Row_r, d), (Col_c, d), and (Box_k, d) probe direction components from (G2) into (G1):

$$x'_{G1} = x_{G1} + \sum_{S \in \{\mathsf{Row}_r,\, \mathsf{Col}_c,\, \mathsf{Box}_k\}} \left(\hat{w}_{S,d}^{\top} x_{G2} - \hat{w}_{S,d}^{\top} x_{G1}\right) \hat{w}_{S,d} \tag{1}$$

where $\hat{w}_{S,d}$ is the unit-normalized probe direction for substructure S and digit d. We then measure the drop in the logit of the clean model's highest-scoring placement token (mean clean logit 16.82). All top-1 statistics in this experiment are computed over the 729 placement tokens.

As shown in Fig. 1 (right), the logit of the clean model's top-ranked placement drops across all layers, peaking at layer 5 with a mean drop of approximately 25 logits and remaining high through layer 8. To assess whether this drop changes the prediction, we measure how often the patched top-1 placement differs from the clean top-1. This occurs in 95.3–99.7% of cases across layers, supporting the causal role of these directions in the model's predictions.

We separately check whether the patched prediction is still a legal placement on the unchanged board. On the same selected sample, the clean top-1 placement is valid in all 300 states. After patching at layers 4 and 5, validity falls to 229/300 (76.3%) and 218/300 (72.7%), respectively. Thus, although most patched predictions remain legal at these layers, the intervention reduces validity by 23.7 and 27.3 percentage points. Per-layer results are reported in Table 4.

3.4 Transfer Across Solving Steps and Layers

Is the substructure-level representation consistent across layers? To identify at which layer the substructure representation stabilizes, we perform a cross-layer probe transfer experiment: for each

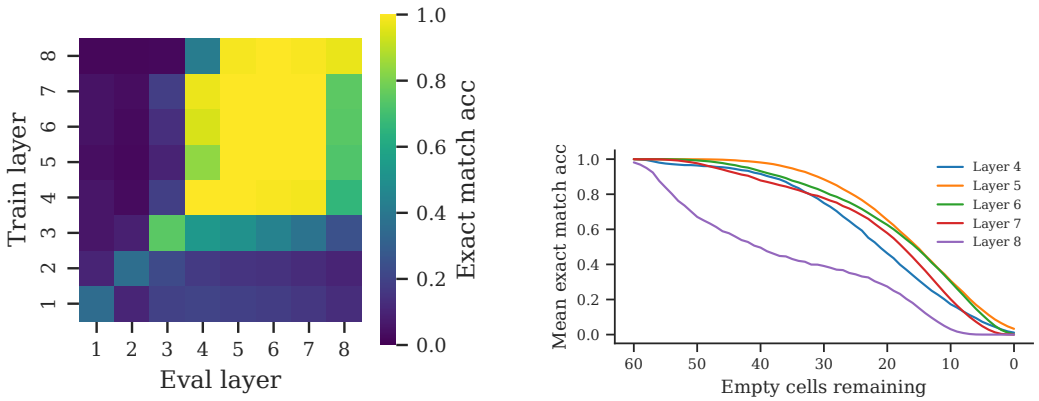


Figure 2: **(Left)** Cross-layer transfer of substructure-state probes: entry (i, j) is the mean exact match accuracy of probes trained on layer i activations and evaluated on layer j activations without retraining; **(Right)** Cross-position transfer of substructure-state probes trained on different layers: mean exact match accuracy of probes trained at `[clues_end]` and evaluated on later positions without retraining.

of the 243 (substructure, digit) probes, we train on activations at a fixed source layer and evaluate on every other layer without retraining. As shown in Fig. 2 (left), the transfer matrix reveals a sharp phase transition at layer 4: probes trained on layers 1–3 fail to transfer to any later layer, while probes trained on layers 4–7 achieve near-perfect exact match accuracy across the entire mid-layer span. It means that the (substructure, digit) world model is absent in early layers, appears at layer 4, and remains stable through layer 7.

Is the substructure-level representation persistent across solving steps? To test if the substructure representation persists dynamically, we apply the frozen probes (trained at the `[clues_end]` token) to subsequent positions in the generating trace. As shown in Fig. 2 (right), the exact match accuracy of these probes steadily degrades toward zero as the puzzle is solved. However, this drop is an artefact of the probes' decision threshold, not the underlying representation. As the grid fills, almost every digit becomes present in almost every substructure, so the optimal threshold shifts, but the probes' biases were fixed at training time on a much sparser distribution. To isolate the underlying feature directions from this shifting decision threshold, we evaluate the ROC-AUC of the exact same probes. As detailed in Section B.3, the AUC stays at 1.0 across the entire trace for the mid-layers, confirming that the model maintains a structurally invariant world model throughout generation.

4 Mid-Layer Routing and Constraint Elimination

In Section 3 we showed that the model's residual stream is organized by Sudoku's substructures. In this section, we ask which components write that organisation into the stream. We analyze the behavior of mid-layer attention heads across a dataset of 6,400 puzzles by extracting the activations at the `[clues_end]` token and evaluating both the attention distributions (QK circuit) and the direct output to the residual stream (OV circuit).

4.1 Attention Patterns

What information are attention heads focusing on? We project the average attention weights from the `[clues_end]` token onto the 9×9 grid. Numerous mid-layer heads heavily concentrate their attention mass on specific substructures: individual boxes, horizontal bands, or vertical stacks (an example can be seen in Fig. 3; full patterns, including ones that cannot be clearly attributed to a substructure, in Section C.1). The mapping between heads and substructures is not strictly one-to-one: some heads attend broadly across a band with stronger concentration on one of its boxes, and some substructures are jointly covered by several heads. Because these heads sit in the middle of the

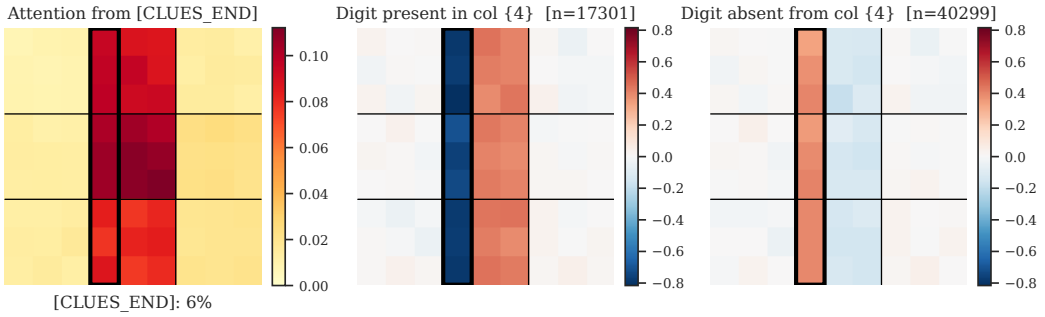


Figure 3: Substructure constraint elimination in mid-layer attention heads. Attention map (left) shows that the head is specialized for routing information from tokens representing placements in middle columns. Direct Logit Attribution (middle and right) shows that the head suppresses candidate logits for digits that are present in its target substructure and promotes digits that are absent.

Head	Region	Target Δlogit	Control Δlogit
L4H6	cols 4–6	1.426 ± 0.004	-0.170 ± 0.002
L5H8	rows 7–9	1.690 ± 0.004	-0.216 ± 0.002
L6H3	box 5	0.937 ± 0.005	-0.108 ± 0.003

Table 1: Mean-ablation suppression gaps for specialized attention heads. Values are mean changes in illegal-placement logits, $\mathbb{E}[\ell_{\text{ablated}} - \ell_{\text{clean}}]$, reported as mean $\pm$ standard error. Positive Δtarget indicate that ablating the head removes suppression in the substructure.

network, "attending to position (r, c)" means reading the residual stream that has accumulated at that position over the previous layers, not the raw token embedding for the cell.

4.2 Heads Direct Logit Attribution

What role does each attention head play in the decision? We hypothesize that this specialized attention allows specific heads to act as dedicated constraint managers for their respective substructures. To verify this, we evaluate the isolated output of these heads using Direct Logit Attribution (DLA): we project each head's contribution to the residual stream through the unembedding matrix to read off its effect on the logits of placement tokens and to see how they update the output probabilities.

We illustrate it with a stack-attending head (Fig. 3). For each digit d and each column within the stack, we split examples by whether d already appears in that column, and average the head's logit contribution across digits and examples. We find that the head implements a symmetric mechanism. When d is present in the column, the head suppresses the logit of d across all cells of that column and slightly promotes d in the other columns of the stack. When d is absent, the head promotes d across the column. The same pattern holds for heads specialized on rows, boxes, bands, and stacks (Section C.2).

4.3 Causal Ablation

If a head suppresses digits already present in its target substructure, then removing it should make the invalid digits more plausible to the model. We test this with mean ablation on three heads.

For each of the three heads we replace its output at the `[clues_end]` position with its mean over a held-out set of 1,280 puzzles. We then measure the change in logit of illegal placements $[\texttt{R}r\texttt{C}c{=}d]$ where (r, c) is in the head's target substructure and digit d already appears there. We additionally measure the effect on a substructure of the same type that the head does not specialize in.

As shown in Table 1, ablating each head raises logits of illegal placements in its target substructure by 0.94–1.69, and lowers them slightly in the control. Thus, the suppression mechanism is implemented specifically and locally, with each head enforcing constraints in its own substructure, and removing it does not disrupt constraints of other substructures.

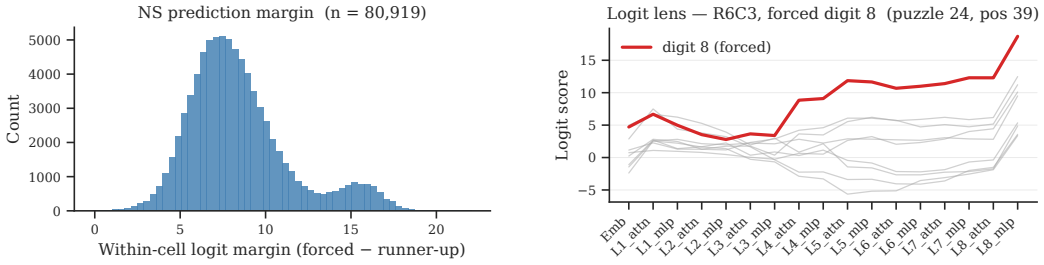


Figure 4: **(Left)** The distribution of the within-cell logit margin (correct digit score - max incorrect digit score) across 80,919 states with unique NS placement. **(Right)** A representative logit lens trace of tokens of a NS cell. The correct digit candidate (red) separates from other digits of the cell (gray) across the layers, with a general promotion of the logits for all digits of the cell in the last MLP block.

5 Understanding the Decision Layer

In Section 4, we demonstrated that the mid-layers accumulate constraint representations in the residual stream: valid candidate placements are promoted, while invalid ones are suppressed.

In this section, we show that the final decision is implemented in the MLP block of the last layer through a two-stage mechanism: mid-layers accumulate the constraints, and a sparse set of neurons in the final MLP gates the commitment. We examine this mechanism on the most common placement type: naked singles (NS).

5.1 Logit Scores Before the Last MLP Block

To establish the state of the network before MLP's intervention, we apply a logit lens to the residual stream immediately following the L8 attention block for all states containing a unique NS. From all 487,725 states, we extract 1,073,757 target NS placements (since one grid may contain several NS simultaneously) and then rank the placement tokens for the target cell based on their raw logit scores. We observe that the correct digit is ranked first in 98.77% of cases. The distribution of margin scores (the logit score of the correct digit for the cell minus the logit score of the runner-up) for the 80,919 states where an NS is unique can be seen in Fig. 4 (left). It shows that before the residual stream reaches the final MLP block, it already contains the correct guess by a wide margin.

5.2 Final Layer Promotion

To determine how the network decides NS placements, we first manually inspect the logit trajectories. As shown in Figure 4 (right), we observe that the final MLP block applies a massive, simultaneous logit promotion to all digit tokens associated with a cell once it reaches a unique candidate state.

In order to isolate the mechanism driving this boost, we scan the final MLP for cell-specific neurons. For each (cell, neuron) pair, we compute the neuron's mean post-activation conditioned on the number of valid candidates remaining in that cell, and define the *activation gap* as the difference between the neuron's mean activation when the cell has exactly one candidate (i.e. an NS state) and its maximum mean activation across other candidate counts.

The distribution of these gaps shows that the vast majority of cell-neuron pairs cluster near zero. However, there's a small high-magnitude population separated from this noise floor. Setting a cutoff at 3.0 isolates 91 neurons. These neurons act as dedicated naked single detectors: they fire exclusively when their associated cell is an NS, and remain unactivated otherwise (Fig. 5). The coverage is complete: 71 cell have a single associated neuron and 10 cells have two.

We quantify the effect of these 91 neurons on the final prediction by applying DLA to their output weights directly: rather than measuring per-example contributions as in Section 4, we project each neuron's output weight vector through the unembedding matrix (folding in the final LayerNorm scale) to read off the logit contribution it makes whenever it fires. We find that the NS-detector neurons output a massive vector that boosts logits of all tokens associated with the cell. On average, a firing neuron applies a +3.00 logit boost strictly to the placement tokens of its associated cell. The variance

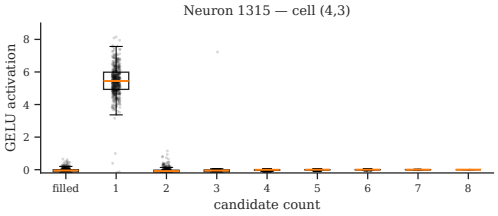


Figure 5: The distribution of activations of an NS neuron.

Placement	Logit drop	Probability drop
Target NS placement	11.408 ± 0.084	0.585 ± 0.003
Other NS placements	-0.655 ± 0.008	–

Table 2: Final-MLP naked-single neuron ablation. Values are clean minus ablated scores, reported as mean $\pm$ standard error over 1000 evaluation states.

of this boost across the 9 target digits is small ($\sigma = 0.10$), which shows that the amplification is essentially digit-uniform. Preliminary analysis of the ten cells associated with two neurons suggests that the paired neurons make similar overall logit contributions, with small digit-dependent differences that largely compensate for one another. Furthermore, the projection onto all other empty cells on the board is negligible ($\mu = -0.02, \sigma = 0.03$). Additional statistics are listed in Section D.

Together, these results describe the full circuit for NS placements: first, the mid-layers rank the unique valid candidate first among the 9 placements for the cell; second, the cell's NS neuron in the final MLP detects the unique-candidate state and uniformly boosts all 9 placement logits for that cell. Since the correct digit was already top-ranked, the uniform boost preserves its lead and the softmax concentrates probability on it.

5.3 Causal Ablation

If an NS neuron is what commits the model to a placement, then ablating it should drop that placement's probability, but leave other naked single placements on the same board untouched. This sets up a within-puzzle control: on a board with two simultaneous NS cells, we ablate the neuron(s) for one and read off the effect on both in a single forward pass.

We collect 1,000 such held-out states and mean-ablate the neuron(s) associated with the cell the model is about to predict. As shown in Table 2, ablating these neurons drops the correct digit's logit by 11.4 and its softmax probability by 0.59. In the clean run the model splits its probability mass roughly evenly between the two NS placements, so a 0.59 drop on one effectively shifts the probability mass to the other. The same ablation barely affects the other NS cell (logit increase 0.66). Thus, the mid-layer signal ranks the correct digit first; the final-layer neuron is what makes the model commit, and it does so for its own cell only.

6 Conclusion and Future Work

We trained an 8-layer transformer on Sudoku solving traces and performed a mechanistic analysis of its internal computation, establishing three results.

The model builds a *substructure world model*: linear binary probes for "is digit d present in substructure S?" achieve perfect exact-match accuracy over each substructure at layers 4–6, while binary cell-content probes peak at 0.572 exact match on filled cells. The geometry of the representation thus mirrors the puzzle's constraint algebra, not its surface decomposition into cells. A sharp phase transition at layer 4 separates the layers that build this representation from those that use it.

Mid-layer attention heads are *substructure-aligned*: they concentrate their attention on individual rows, columns, or boxes, and their OV circuits suppress digits already present in the attended substructure while promoting absent ones, creating a localized constraint-checking mechanism operating directly on the substructure-organized residual stream.

The final MLP contains a *naked-single circuit*: 91 monosemantic neurons that detect when exactly one digit remains possible for a given cell. The residual stream before the final MLP already promotes the correct digit in 98.77% of unique naked-single cases, confirming that mid-layers encode the answer and the final MLP amplifies and commits to it.

Preliminary analyses of two additional constraint-satisfaction tasks provide complementary evidence beyond Sudoku. In work allocation, assigned-only logistic probes achieve 0.546 exact match across each job's nine assignment bits at layer 3, while logistic coverage probes recover the complete 45-bit worker and machine states with exact-match accuracies of 0.966 and 0.976, respectively. In graph coloring, several attention heads specialize on vertices within particular cliques, mirroring Sudoku's substructure-aligned attention. Appendix E describes both experiments and their supporting plots.

These findings suggest that transformers trained on solving traces may encode the current constraint state more cleanly than individual surface facts. In Sudoku, this state keeps track of digit occupancy in rows, columns, and boxes, which determines which digits are legal in an empty cell. An implication for interpretability research is that probing only individual cell contents may underestimate what the model has learned. Interpretability analyses should also consider representations organized around task constraints.

Limitations. Our Sudoku results concern a single model architecture (8 layers, 8 heads, $d_{\text{model}} = 576$) trained on Sudoku traces; analyses of the additional domains are preliminary. We do not test whether the substructure world model or the naked-single circuit appear in models of different sizes or in models trained on non-backtracking traces. Although we have preliminary indications that the representational finding is not sensitive to transformer hyperparameters, separating the respective contributions of task structure, architecture, and training protocol remains future work. The attention-head analysis in Section 4 characterizes the OV circuits but does not provide a full account of how attention patterns are selected (QK circuits).

Future work. The most immediate directions are tracking when the substructure geometry and the circuits emerge during training, either gradually or through a grokking-like transition [Power et al., 2022], and testing how broadly these findings extend to constraint-satisfaction tasks whose compact constraint state differs from their surface state.

Acknowledgments and Disclosure of Funding

This work was supported by the PEPR IA SAIF project, funded by the "France 2030" government investment plan managed by the French National Research Agency, under the reference ANR-23-PEIA-0006.

References

G. Alain and Y. Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Y. Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, N. DasSarma, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah. A mathematical framework for transformer circuits. 2021. Transformer Circuits Thread, `https://transformer-circuits.pub/2021/framework/index.html`.

N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec, Z. Hatfield-Dodds, R. Lasenby, D. Drain, C. Chen, R. Grosse, S. McCandlish, J. Kaplan, D. Amodei, M. Wattenberg, and C. Olah. Toy models of superposition. 2022. Transformer Circuits Thread, `https://transformer-circuits.pub/2022/toy_model/index.html`.

A. Geiger, H. Lu, T. Icard, and C. Potts. Causal abstractions of neural networks. *Advances in neural information processing systems*, 34:9574–9586, 2021.

P. Giannoulis, Y. Pantis, and C. Tzamos. Teaching transformers to solve combinatorial problems through efficient trial & error. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL `https://openreview.net/forum?id=MLprqOvAAK`.

M. Ivanitskiy, A. F. Spies, T. Räuker, G. Corlouer, C. Mathwin, L. Quirke, C. Rager, R. Shah, D. Valentine, C. D. Behn, K. Inoue, and S. W. Fung. Linearly structured world representations in maze-solving transformers. In M. Fumero, E. Rodolá, C. Domine, F. Locatello, K. Dziugaite, and C. Mathilde, editors, *Proceedings of UniReps: the First Workshop on Unifying Representations in Neural Models*, volume 243 of *Proceedings of Machine Learning Research*, pages 133–143. PMLR, 15 Dec 2024. URL https://proceedings.mlr.press/v243/ivanitskiy24a.html.

A. Karvonen. Emergent world models and latent variable estimation in chess-playing language models. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=PPTrmvEnpW.

K. Li, A. K. Hopkins, D. Bau, F. Viégas, H. Pfister, and M. Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*, 2023. URL https://openreview.net/forum?id=DeG07_TcZvT.

T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In L. Vanderwende, H. Daumé III, and K. Kirchhoff, editors, *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL https://aclanthology.org/N13-1090/.

N. Nanda, A. Lee, and M. Wattenberg. Emergent linear representations in world models of self-supervised sequence models. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 16–30, 2023.

P. Norvig. Solving every Sudoku puzzle. https://norvig.com/sudoku.html, 2006.

nostalgebraist. Interpreting GPT: the logit lens. https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/%interpreting-gpt-the-logit-lens, 2020.

K. Park, Y. J. Choe, and V. Veitch. The linear representation hypothesis and the geometry of large language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=UGpGkLzwpP.

A. Power, Y. Burda, H. Edwards, I. Babuschkin, and V. Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022. arXiv:2201.02177.

D. Radcliffe. 3 million Sudoku puzzles with ratings. Kaggle dataset, https://www.kaggle.com/datasets/radcliffe/3-million-sudoku-puzzles-with-ratings, 2020.

K. Shah, N. Dikkala, X. Wang, and R. Panigrahy. Causal language modeling can elicit search and reasoning capabilities on logic puzzles. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=i5PoejmWoC.

A Details on the Model and Training

A.1 Sudoku solving traces

Training traces are generated by a Norvig-style solver [Norvig, 2006] modified to introduce randomization at every choice point. At each step, the solver first looks for naked singles (cells with exactly one remaining candidate) and emits all of them before considering any other rule; when several naked singles are available simultaneously, their order is randomised. Once no naked single remains, the solver looks for hidden singles (digits with exactly one remaining position in a row, column, or box) and emits them in random order. When neither rule applies, the solver picks a random cell among those with the smallest candidate set and opens a backtracking branch over that cell's candidates, again in random order; failed branches are rolled back via `[pop]` and the next candidate is tried. This randomization produces multiple valid traces per puzzle and prevents the model from latching onto a fixed cell ordering or digit ordering at training time. We generate exactly one trace per puzzle and the randomization in the solver ensures the model cannot rely on a fixed cell or digit order.

A.2 Model architecture and training regime

We train a standard causal decoder-only transformer following the architecture of Giannoulis et al. [2026]: $L = 8$ layers, $H = 8$ attention heads, pre-layer normalisation, model dimension $d_{\text{model}} = 576$, MLP width $d_{\text{mlp}} = 3{,}456$ (expansion ratio $6\times$), and GELU activations. We do not use dropout.

The training objective is the standard next-token cross-entropy loss, with the loss mask applied only to tokens after `[clues_end]` so that the model is supervised on its own placements and bookkeeping tokens but not on memorising the input clues. This differs from Giannoulis et al. [2026] who applies special loss based on the validity of the next possible tokens; we found the standard masked loss sufficient to reach comparable accuracy on this task.

We train on 2.7M puzzles from the sudoku-3m dataset [Radcliffe, 2020], with traces trimmed to 250 tokens (only 0.9% of puzzles have longer traces), for 6 epochs. Optimisation uses AdamW with learning rate 10^{-3}, weight decay 0.1, batch size 512, and a cosine schedule with 5M tokens of linear warmup. Training runs on a single Google TPU v6e and reaches 98.4% per-cell and 97.5% per-grid accuracy on a held-out test set of 150K puzzles. Truncation occurs at token boundaries, so retained prefixes need not end with `[success]` and may leave a backtracking branch open. These prefixes remain valid next-token training examples. Early experimentation found no substantial performance effect from retaining them, so we did not discard them.

B Representation Probing Materials

B.1 MSE and AUC Scores of Probes

All probe families are trained on residual-stream activations at the `[clues_end]` token. For the binary cell-content experiment, we use 6,400 held-out puzzles and fit an independent logistic regression for each $(\text{cell}, \text{digit})$ pair at each layer. For each cell, we retain only puzzles in which it is filled, then split those examples into 80% training and 20% test data. We reuse these per-cell splits across layers. The probes use $C = 1$, a maximum of 1,000 iterations. Predicted probabilities above 0.5 are classified as positive. Exact match requires all nine predictions for a cell to be correct and is averaged equally across the 81 cells; ROC-AUC is computed separately for each of the 729 binary probes and then averaged.

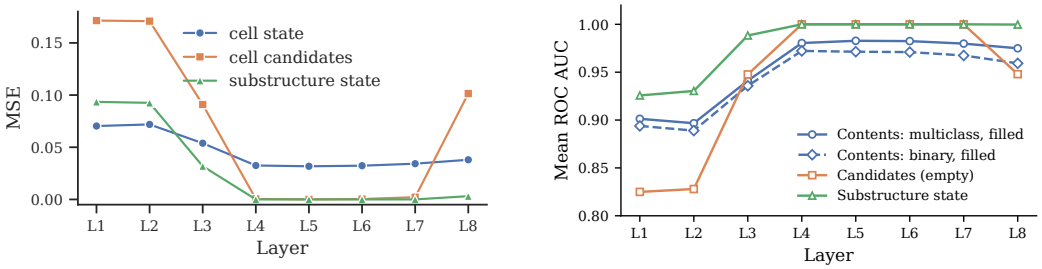


Figure 6: **(Left)** Mean squared error between the probes' predicted probability vectors and the target vectors, per layer, for the three probe families: 81 multiclass cell-content probes (blue), 729 cell-candidate probes (orange), and 243 substructure-state probes (green). Multiclass cell-content probes settle around $\text{MSE} \approx 0.03$ in mid-layers and never reach zero, consistent with their imperfect top-1 accuracy; substructure-state probes drive MSE close to zero from layer 4 onward and stay there through the final layer; cell-candidate probes match this in mid-layers but deteriorate sharply at L8. **(Right)** Mean ROC-AUC of the three probe families shown on the left, together with the filled-only binary cell-content probes (dashed blue). The latter peak at 0.972 at layer 4, while substructure-state probes reach 1.0 at layers 4–7 and remain near-perfect at layer 8. The solid blue curve shows the multiclass cell-content probes; cell-candidate AUC drops at layer 8.

B.2 Substructure Alignment of Cell-Candidate Probes

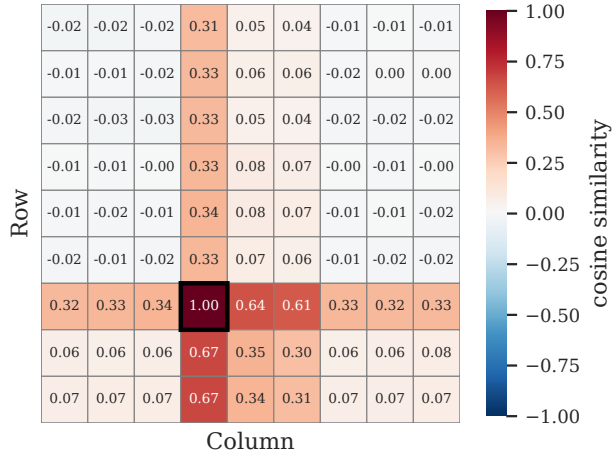


Figure 7: Cosine similarities between the probe trained at layer 6 activations to predict if 3 is a valid candidate in cell $(7, 4)$ and probes trained on other cells for digit 3.

Category	L3	L4	L5	L6	L7	L8
r∩box	0.888 ± 0.017	0.737 ± 0.098	0.648 ± 0.069	0.648 ± 0.021	0.623 ± 0.017	0.559 ± 0.054
c∩box	0.650 ± 0.064	0.582 ± 0.093	0.707 ± 0.058	0.657 ± 0.017	0.636 ± 0.017	0.587 ± 0.053
r,¬box	0.390 ± 0.048	0.443 ± 0.076	0.325 ± 0.035	0.339 ± 0.017	0.329 ± 0.016	0.351 ± 0.066
c,¬box	0.306 ± 0.062	0.342 ± 0.063	0.383 ± 0.068	0.326 ± 0.016	0.319 ± 0.018	0.333 ± 0.062
box	0.560 ± 0.070	0.332 ± 0.093	0.379 ± 0.047	0.340 ± 0.022	0.298 ± 0.022	0.334 ± 0.074
stack	0.238 ± 0.066	0.105 ± 0.035	0.069 ± 0.014	0.053 ± 0.014	0.036 ± 0.014	0.224 ± 0.061
band	0.097 ± 0.069	0.049 ± 0.030	0.071 ± 0.013	0.057 ± 0.013	0.040 ± 0.016	0.223 ± 0.075
none	-0.062 ± 0.055	0.014 ± 0.014	0.009 ± 0.007	-0.016 ± 0.009	-0.022 ± 0.010	0.117 ± 0.067

Table 3: Mean cosine similarity with std across cell probe vectors trained to predict valid candidates in a cell. Scores are grouped by probes sharing a digit and a certain type of substructure, e.g. r∩box means that two probes share a digit, a row and a box.

B.3 MSE and AUC Scores for Transfer Across Timesteps

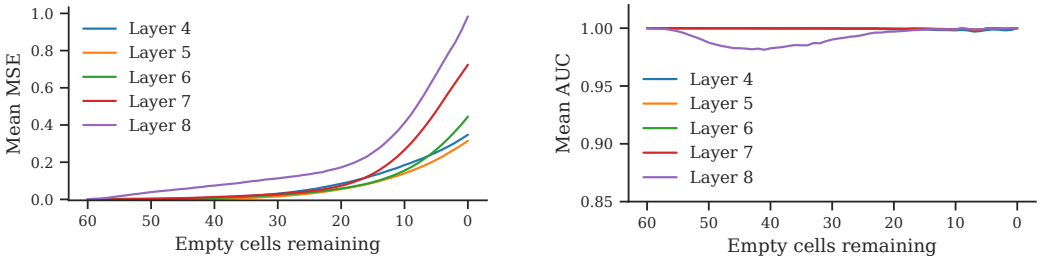


Figure 8: Cross-position transfer diagnostics complementing Fig. 2 (right). Frozen substructure-candidate probes trained at the `[clues_end]` token are evaluated at every subsequent position of the solving trace, plotted against the number of empty cells remaining at that position. Curves are means over 1,280 held-out puzzles, with one curve per probe layer (L4–L8). **(Left)** Mean squared error rises sharply as the puzzle is solved, particularly for L8 which approaches ≈ 1.0 at the end of the trace. **(Right)** Mean ROC-AUC of the same probes stays at ≈ 1.0 for L4–L7 across the entire trace, with only L8 dipping marginally (to ≈ 0.98) in the mid-trace region. The contrast between the two panels confirms that the apparent degradation in exact-match accuracy across the trace is a calibration artefact of the probes' fixed biases, not a loss of the underlying substructure directions, which remain linearly separable throughout generation.

B.4 Causal Ablation Using Substructure Probe Vectors

Layer	Logit drop	Patched logit	Valid top-1	Changed top-1
L1	13.978 ± 0.311	2.845	0.723	0.987
L2	8.724 ± 0.293	8.099	0.690	0.953
L3	15.054 ± 0.295	1.769	0.723	0.993
L4	23.727 ± 0.332	-6.904	0.763	0.993
L5	24.766 ± 0.313	-7.943	0.727	0.997
L6	22.124 ± 0.285	-5.301	0.650	0.997
L7	21.775 ± 0.264	-4.952	0.597	0.997
L8	19.235 ± 0.243	-2.413	0.607	0.993

Table 4: Per-layer results of the counterfactual patching intervention using substructure probes on 300 initial states selected for a legal first generated placement. Layers are numbered 1–8, as in Fig. 1. All top-1 statistics rank only the 729 placement tokens. *Logit drop*: mean decrease in the logit of the clean model's highest-scoring placement token (mean clean logit 16.82), reported as mean $\pm$ standard error. *Patched logit*: mean logit of that same token after patching. *Valid top-1*: fraction of patched runs whose top-ranked placement is legal on the initial board; the matched clean baseline is 300/300 (100%). *Changed top-1*: fraction of patched runs whose top-ranked placement differs from the clean top-ranked placement.

B.5 Substructure Organization of the Unembedding Vectors

We note that the placement-token unembeddings are substructure-organized. Pairwise cosine similarities among the 729 unembedding vectors cluster by which features the tokens share — row, column, box, or digit — with near-identical similarity values within each cluster. For instance, all token pairs sharing a row and a digit have similarity 0.2, and those sharing a row, a box and a digit 0.37, while logically unrelated pairs are near-orthogonal.

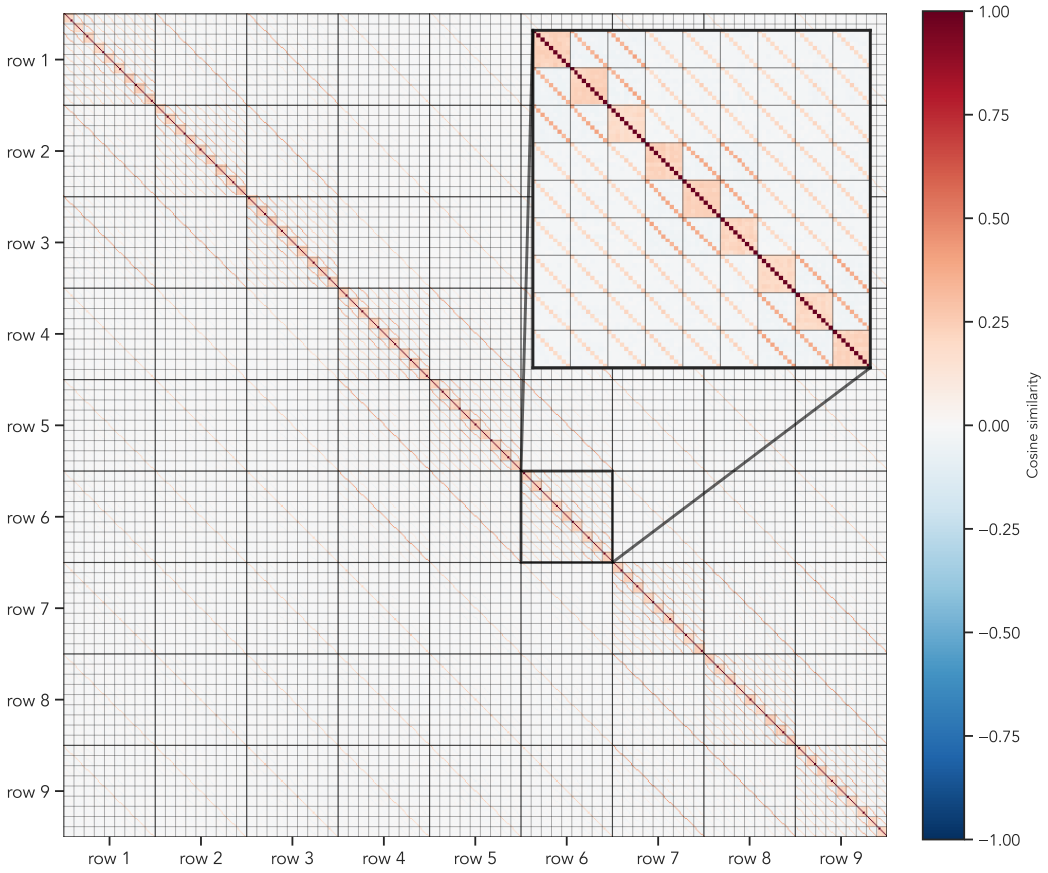


Figure 9: Pairwise cosine similarity of the unembedding vectors. Each of the nine big cells is a row, each small square is a column, and the smallest squares are digits. Zoomed-in part shows cosine similarities of the unembedding vectors of row 6.

C Mid-Layer Attention

C.1 Mean Attention From `[clues_end]` Across All Heads

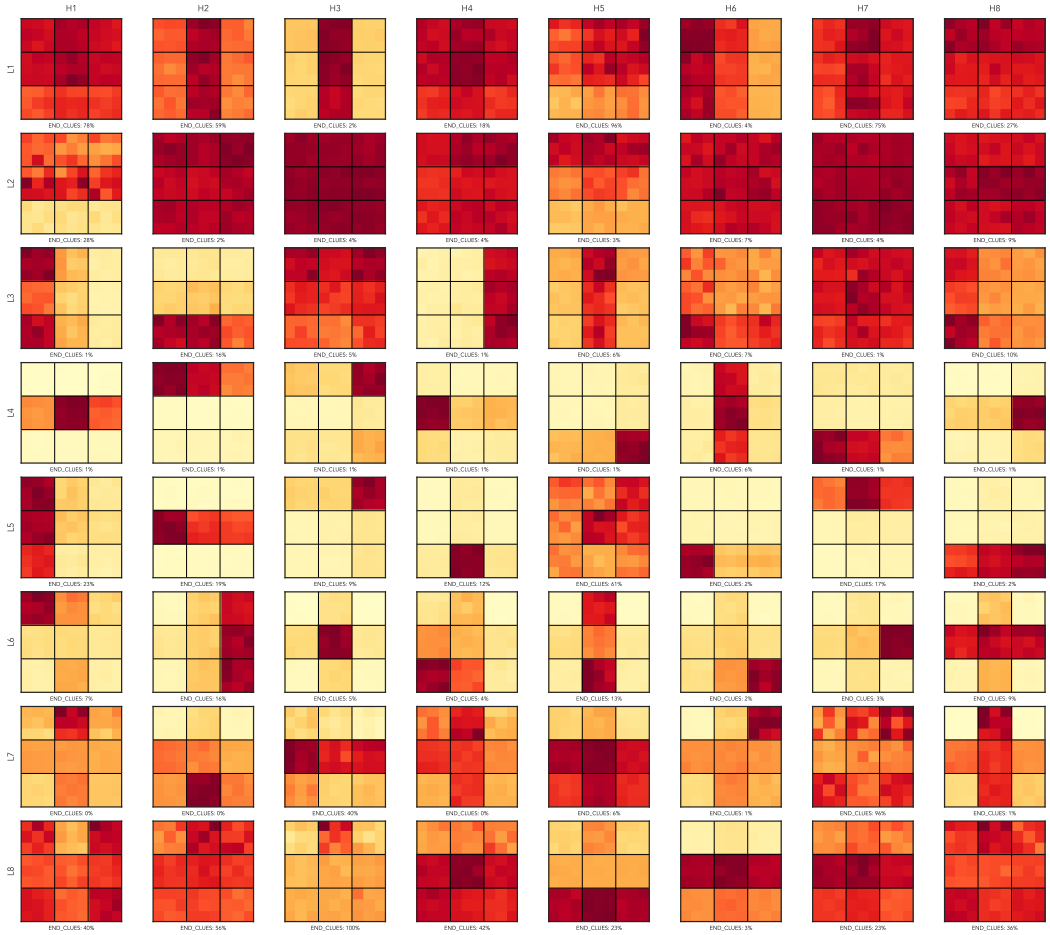


Figure 10: Mean attention scores from `[clues_end]` token to other tokens in the grid, averaged over all digits, computed over 6400 puzzles.

C.2 Additional Examples of DLA for Substructure Aligned Heads

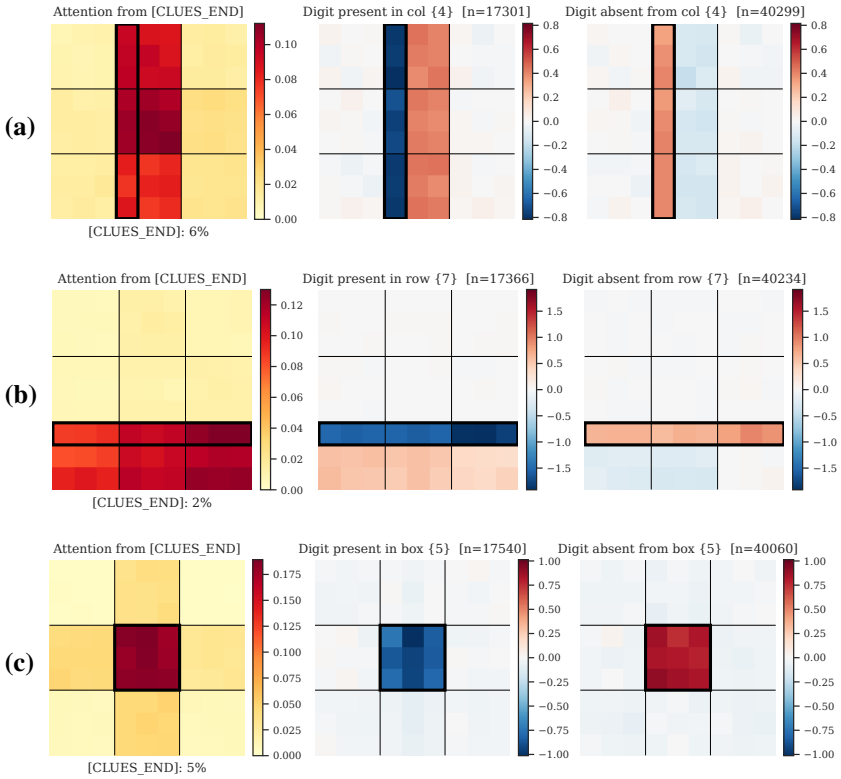


Figure 11: Substructure constraint elimination in mid-layer attention heads. Attention maps (left) shows that heads are specialized for routing information from specific substructures: (a) columns, (b) rows, and (c) boxes. Direct Logit Attribution (middle and right) shows that each head suppresses candidate logits for digits already present in its target substructure, while promoting absent digits.

D NS Circuit Materials

Statistic	Target Mean	Target Std	Other Mean	Other Std
Mean	3.00	0.11	−0.03	0.03
Std	0.72	0.04	0.01	0.01
Min	0.87	0.03	−0.05	0.02
25%	2.53	0.08	−0.03	0.03
50%	3.12	0.10	−0.03	0.03
75%	3.48	0.13	−0.02	0.03
Max	4.48	0.23	0.00	0.05

Table 5: DLA metrics for the $N = 91$ identified naked single neurons. *Target* metrics represent logit contributions to the 9 placement tokens of the neuron's associated cell. *Other* metrics represent the global projection onto all 720 placement tokens of the remaining 80 empty cells.

E Additional Constraint-Satisfaction Domains

Work allocation. The task assigns 45 jobs to 45 workers and 45 machines, using each exactly once. Each job admits three workers and three machines, with respective offsets $\{0, 1, 3\}$ and $\{0, 3, 7\}$ from the job index, modulo 45. Each compatible (job, worker, machine) triple is an atomic token, giving 405 assignment tokens. Traces begin with preassignments, averaging 17, and use `[push]`, `[pop]`, and `[success]` to encode backtracking and completion.

We probe post-MLP residual activations at `[SEP]` in the six-layer model, using a shared instance split of 4,800 probe-training and 1,600 test examples. All readouts are independent logistic regressions with $C = 1$ and a fixed 0.5 probability threshold. Nine probes per job predict its assignment; each job's probes are trained and evaluated only on examples where that job is assigned, so their target is a one-hot nine-bit vector. Coverage probes use all examples and predict whether each worker (B) or machine (C) is assigned. Table 6 reports scores at layer 3, where the assigned-only job probes perform best; Fig. 12 shows the layerwise comparison. The coverage states are recovered much more reliably despite requiring 45 simultaneous correct predictions rather than nine, reproducing the qualitative separation observed in Sudoku (Section 3).

Target	Mean binary accuracy	Exact match
Assigned jobs (9 bits)	0.9408	0.5457
Workers (45 bits)	0.9992	0.9656
Machines (45 bits)	0.9994	0.9756

Table 6: Work-allocation logistic probes at layer 3. Job scores use assigned examples only. Exact match requires all nine bits of one job's assignment vector, or all 45 bits of a worker- or machine-coverage vector, to be correct.

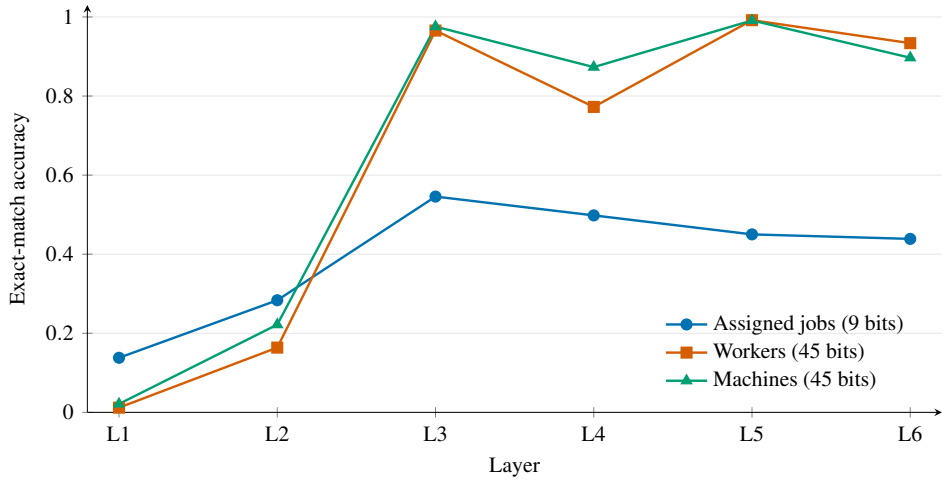


Figure 12: Layerwise exact-match accuracy for work-allocation logistic probes. Job probes are trained and evaluated only on assigned examples; worker- and machine-coverage probes use all examples from the same instance split. The target vectors contain nine bits per job and 45 bits per resource set. All classifiers use the same fixed 0.5 probability threshold.

Graph coloring. We use a fixed graph with 48 vertices arranged into twelve four-vertex cliques, with additional inter-clique edges giving every vertex degree six. The task assigns four colors so that adjacent vertices differ. Each (vertex, color) assignment is an atomic token, giving 192 assignment tokens, and solving traces include backtracking. We inspect attention from the end-of-clues token to the clue vertices. Several heads specialize on particular cliques (Fig. 13), extending the constraint-aligned attention observation from Sudoku (Section 4).



Figure 13: Graph-coloring attention patterns over 6,400 instances. Rows are layers and columns are heads; the vertex layout groups the twelve four-vertex cliques. Attention is normalized over clue tokens within each instance and averaged conditional on a vertex being a clue. Colors are scaled independently for each head. Labels beneath the panels give mean attention to the end-of-clues token. Some heads concentrate on particular cliques, while others attend more broadly.

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 claim two results—a substructure world model and a naked-single circuit—which are directly supported by the experimental findings in Sections 3 to 5. Claims are scoped to a single model trained on one puzzle type and do not over-generalise.

   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: The paper currently includes a discussion on limitations in the conclusion section.

   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 is entirely empirical and contains no theorems or formal proofs.

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: Model architecture, training dataset, hyperparameters, and probing protocol are described in Section 2 and the appendix. The sudoku-3m dataset is publicly available. Code and model checkpoints are accessible via anonymized links.

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: Code, training data and model checkpoints are accessible via anonymized links. Original puzzle data (sudoku-3m) is publicly available on Kaggle [Radcliffe, 2020].

   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: Architecture, dataset size, context length, training epochs, hardware, and accuracy metrics are reported in Section 2. Optimizer and learning-rate schedule details are in the appendix.

   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 experimental results come with error bars and standard deviation / standard error.

   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: Training hardware (Google TPU v6e) is stated in Section 2. Analysis experiments (probing, logit lens, neuron identification) run on CPU/single GPU and are lightweight relative to training; this is noted in the appendix.

   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 is fundamental mechanistic interpretability research on a Sudoku-solving model. It involves no human subjects, no sensitive data, and no harmful applications.

   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 is foundational interpretability research on a narrow task. There is no direct path to harmful applications. The positive contribution is to scientific understanding of how transformers represent structured knowledge, supporting the broader goal of interpretable AI.

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 released model is a small transformer trained solely to solve Sudoku puzzles. It poses no meaningful risk of misuse.

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: The sudoku-3m dataset is cited [Radcliffe, 2020]; its Kaggle licence (CC0/public domain) permits unrestricted research use. All other assets are publicly available research code cited in the bibliography.

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: The trained model checkpoint and analysis code are available with documentation.

    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: The paper involves no crowdsourcing or human subjects.

    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 involves no human subjects research.

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: The paper analyses a transformer trained on Sudoku traces; no external LLM is used as part of the research methodology.

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.