WHAT COUNTS AS EVIDENCE FOR A WORLD MODEL? REASSESSING OTHELLO-GPT

Anonymous authors
Paper under double-blind review

ABSTRACT

Othello-GPT, a transformer trained to predict legal moves in Othello, has become the standard example of a neural network that uses a world model. The evidence falls into three categories: near-perfect prediction of legal moves, near-perfect linear decoding of the board state, and highly successful causal interventions. We argue that this evidence is ambiguous once we adopt the right baseline and the right metrics. Our baseline, which we call Othello-MLP, uses heuristics rather than a world model, and scores nearly as well on all three criteria. We then introduce an additional piece of evidence—transfer learning speeds on geometrically coherent versus incoherent extensions of Othello—which further suggests that Othello-GPT is using a collection of heuristics. We conclude that the total evidence is at best ambiguous, and if anything favors a heuristic interpretation. We draw four general lessons. Two concern how to test for world models: (1) when attributing a world model, the right baseline is a network that uses heuristics rather than a network with randomized weights; and (2) the right metrics take into account patterns of error and the full output distribution. The other two lessons are about how to weigh the different kinds of evidence: (3) causal interventions can provide illusory evidence; and (4) transfer learning can be more revealing than task performance, decoding accuracy, and causal interventions.

1 INTRODUCTION

It is possible to solve a task without understanding why the solution works: to navigate by following a memorized series of turns; to play chess by recognizing familiar patterns; to diagnose an engine by recalling which sounds indicate which malfunctions; and to bet in poker by remembering which hands usually win. But it is also possible to solve a task by deriving the solution from a model of the task: to navigate by consulting an internalized map; to play chess by calculating moves; to diagnose an engine by reconstructing how its parts interact; and to bet in poker by computing the odds. These approaches can look the same from the outside. How can we determine when a neural network derives its solution from a model of the task—a 'world model'—rather than using a memorized lookup table or a collection of specialized heuristics (Fig. 1)?

Whether networks use world models bears on several larger questions. First, do networks understand the world at all, or are they "stochastic parrots" that reproduce statistical regularities in their training data (Bender & Koller, 2020; Agüera y Arcas, 2022; Mitchell & Krakauer, 2023)? If a network uses a world model, it must have at least a minimal understanding of its task. Second, what kind of training produces world models? Is it enough to train them to predict passive streams of unimodal data (Butkus & Kriegeskorte, 2025), or does it require something more, such as embodiment (Bisk et al., 2020), multi-modal input (Paik et al., 2021), or the ability to act and observe the results of interventions (Pearl & Mackenzie, 2018; Zečević et al., 2023)? Third, do AI systems share our world models? Because neural networks are often trained on data that reflects our own understanding, they might reverse-engineer the same models. But they might also develop an "alien" understanding of the world, organized on foreign principles. Fourth, can we trust networks in unfamiliar situations? A network with a world model will respond correctly as long as its model still applies. A network using heuristics or lookup tables relies on a patchwork of situation-specific tricks that might not generalize to new situations.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

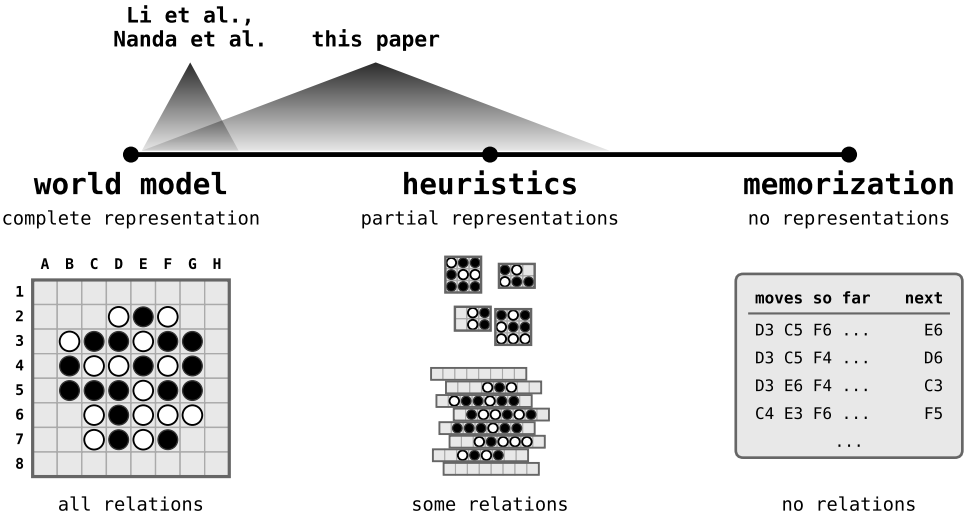


Figure 1: World models, heuristics, and memorization fall along a continuum. They differ in completeness: a representation can include more or fewer squares. They also differ in structural relations: a representation can include more or less of the 8×8 grid structure. We argue that evidence leaves it ambiguous whether Othello-GPT falls near the world-model end.

Othello provides a controlled setting for thinking these questions through. It is a two-person game in which black and white take turns capturing pieces by flanking them. Othello-GPT is an 8-layer GPT-style transformer trained to predict legal moves: it takes a sequence of moves as input and outputs a probability distribution over the next move. Li et al. (2023) and Nanda et al. (2023) argue that it uses a world model. If they are right, it is plausible that large language models use world models too, since they are also trained on next-token prediction. Their argument appeals to three kinds of evidence: (i) high performance on the prediction task, (ii) high decoding accuracy of the board state from internal layers of the network, and (iii) high efficacy of causal interventions on those internal layers. Othello-GPT has become the standard example of a neural network using a world model (e.g., Richens & Everitt, 2024; Guan et al., 2023; Millière & Buckner, 2024; Gurnee & Tegmark, 2024), and many take Li et al. (2023) and Nanda et al. (2023) to provide a blueprint for establishing world models in other networks and tasks (Ivanitskiy et al., 2023; Karvonen, 2024; Butkus & Kriegeskorte, 2025; Kniazev & Fijalkow, 2026).

Against this consensus, we argue that the evidence that Othello-GPT uses a world model is at best ambiguous, and if anything better supports heuristics. We begin by proposing two necessary conditions on using a world model, which we call *completeness* and *structural relations*. We then make four contributions, each of which bears on the broader question of when to attribute world models to neural networks.

1. **Baselines.** We build Othello-MLP, a network that satisfies neither of the two conditions and solves the task through a collection of heuristics, and show that Othello-GPT's advantage is too small to establish a world model. More generally, the right baseline for attributing a world model is a network that uses heuristics, not one with randomized weights.
2. **Metrics.** We show that the metrics reported in previous work are too coarse, because they aggregate away details that distinguish a world model from heuristics. Accordingly, we use finer-grained measures that account for the full output distribution as well as patterns in Othello-GPT's errors. Our metrics reveal signatures of heuristics.
3. **Causal interventions.** Although causal interventions on Othello-GPT's residual stream move its predictions in the right direction, we show that the edited network still does not treat newly legal moves as legal; that their success depends on the type and location of the edit; and that they succeed about as well on our heuristic baseline, Othello-MLP. More generally, causal interventions can provide illusory evidence for a world model.

4. **Transfer.** We introduce a fourth kind of evidence, based on *transfer* to extensions of the original game. Othello-GPT learns a geometrically coherent extension of the board no faster than an incoherent one. More generally, task transfer can be more revealing than task performance, decoding accuracy, or causal interventions.

2 What does it mean to use a world model?

There is no agreed-upon definition of what it means to use a world model, but for Othello there is a minimal notion that should be uncontroversial. A network uses a world model only if it derives its predictions from a single, unified representation of the board that satisfies two conditions:

- **Completeness.** The representation includes the occupancy of every square.
- **Structural relations.** The representation includes the arrangement of those squares into an 8×8 grid, and thus which squares lie along the same rows, columns, and diagonals.

By *heuristics* we mean rules defined over the move history rather than over a representation of the board. A network using heuristics fails both conditions. Each heuristic looks for its own pattern, and while that might involve representing some squares and some of their relations, there is no single, unified representation that includes all squares and their arrangement into a grid. A lookup table fails these conditions entirely, mapping each move sequence to a prediction without representing any squares or relations. We take no stand on whether these conditions generalize beyond Othello (see Appendix A for further discussion).

Two features of this notion matter for what follows. First, world models, heuristics, and lookup tables lie on a continuum, since completeness and structural relations each admit of degree. Second, they are not exclusive: a network might use heuristics for some moves and a world model for others—perhaps heuristics in the early and end game and a world model in the midgame. We argue that the evidence leaves it ambiguous where Othello-GPT falls on this continuum, and if anything favors heuristics. We also argue that the ambiguity is systematic: there is no range of moves on which the evidence unambiguously favors the world-model hypothesis.

3 Related Work

Othello has been a machine learning testbed since the 1980s (Rosenbloom, 1982; Lee & Mahajan, 1990; Moriarty & Miikkulainen, 1995). Li et al. (2023) were the first to argue that a transformer trained to predict legal moves uses a world model. Nanda et al. (2023) improved on their work by showing that the board state could be linearly decoded. Follow-up work has extended this framework across models (Yuan & Søgaard, 2025) and modalities (Chen et al., 2025), and has provided similar evidence for world models in chess (Karvonen, 2024), mazes (Ivanitskiy et al., 2023), Sudoku (Kniazev & Fijalkow, 2026), street maps (Beckmann et al., 2026), and structural causal models (Butkus & Kriegeskorte, 2025).

Closest to our paper, Lin et al. (2024) and Singh et al. (2025) recover decision rules from Othello-GPT's neurons. However, they define these rules over the board state. On their account, the network first represents the board and then applies rules to it; on ours, the network might just use rules defined over the move history.

Skepticism about world models has also been raised elsewhere (Vafa et al., 2024; 2025; Nikankin et al., 2025). Our points about baselines and interventions build on work showing that decodability does not imply use (Hewitt & Liang, 2019; Belinkov, 2022) and that interventions on probe-derived subspaces, though better evidence of use (Elazar et al., 2021), can themselves produce illusory effects (Makelov et al., 2024). Our reliance on transfer follows a long tradition of behavioral tests for world models (Tolman, 1948; Warrier et al., 2026; Park, 2026), and contrasts with Chawla et al. (2026), who train one model on several versions of Othello and report a shared board representation.

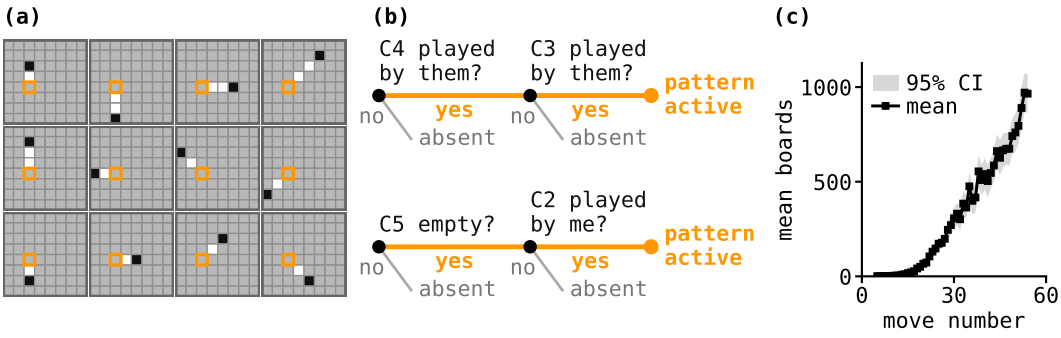


Figure 2: Othello-MLP, a heuristic baseline for legal-move prediction on Othello. **(a)** Example flanking patterns around C4 (orange). **(b)** We train decision trees to predict whether a flanking pattern is active. **(c)** Mean number of boards consistent with the played (unordered) encoding, by move number.

4 METHODS

4.1 OTHELLO-GPT

Othello-GPT is an 8-layer transformer with the GPT-2 architecture. The input is a sequence of numbers, one per move, indicating which square was played. The output is a probability distribution over all possible moves. Li et al. (2023) trained the network on two datasets: actual championship games and synthetic games, which were generated by randomly sampling a legal move at each step. We focus on the synthetic version because it provided stronger evidence of a world model.

4.2 OTHELLO-MLP

Othello-MLP is organized around *flanking patterns*. In Othello, a move is legal when it flanks a line of the opponent's pieces (Fig. 2a). The line can extend in eight possible directions: up, down, left, right, and the four diagonals. Taking into account all possible squares, directions, and lengths, there are 960 possible flanking patterns in total. Each flanking pattern is *local*: it depends only on the pieces along a single line.

Othello-MLP is a multilayer perceptron with three layers: an input layer encoding the move history, a fixed hidden layer that detects flanking patterns (derived from decision trees), and an output layer of 960 pattern scores. We describe each in turn.

Input. The move history was encoded as $x \in \{0,1\}^{60\times 2}$, where for each playable square s_i, $i = 1, \ldots, 60$,

$$x_{i,1} = \mathbb{1}[s_i \text{ has been played}], \qquad x_{i,2} = \mathbb{1}[s_i \text{ was played by the current mover}].$$

Here *the mover* is the player to move, determined by the parity of the move count under strict alternation; this ignores forfeits, in which a player with no legal move is skipped. This follows Nanda et al. (2023), who found the board linearly decodable from Othello-GPT only in mover-relative terms. We call this the *played (unordered)* encoding, since it records whether and by whom each square was played, but not the order in which the moves were made. This encoding contains strictly less information than Othello-GPT's input, since it discards move order, and Othello-GPT could compute it in a single layer. We found that discarding move order made it easier to identify leaves that generalized across positions, and, as we'll see, it strengthens our argument that Othello-MLP does not use a world model.

Hidden layer. For each of the 960 flanking patterns—each specified by a target square, a direction, and a flank length—we trained a decision tree to predict from a played (unordered) encoding of the move history whether that pattern is present. We then instantiated one hidden unit per leaf, which

fires exactly when the move history satisfies the conjunction of feature tests along that leaf's root-to-leaf path: for example, that B5 has not been played and that C5 and D5 were both played by the opponent. Pooling the leaves of all 960 trees gives a fixed hidden layer of 47,061 units, constructed rather than learned. Approximately 2% of them are active for any given move, since exactly one leaf per tree fires. See Appendix B for more details.

Output. A dense linear readout, trained by gradient descent, maps the hidden layer to one score per flanking pattern, passed through a sigmoid. The trees fix which conjunctions the hidden layer computes, but the readout is free to combine them however best predicts each pattern; only this readout is trained. The predicted move is the target square of the highest-scoring pattern.

Training objective. Othello-MLP is trained on a different objective than Othello-GPT; it is trained to detect flanking patterns rather than to predict the next move.[1] What matters for our purposes is that Othello-MLP does not use a world model (by design) yet scores well on Li et al. metrics.

4.3 Othello-MLP uses heuristics

Othello-MLP is best understood as using a collection of heuristics rather than a world model: each hidden unit detects an interpretable conjunction of move features, and the network's output is simply a weighted vote, over those features, for or against each flanking pattern. It fails completeness, since it derives its predictions from roughly 47,000 independently calculated features rather than a single board representation. By construction, no hidden unit reads the output of another, and each is a fixed conjunction of input features, so the hidden layer contains no mechanism by which information about different squares could be combined before the readout. And the mere fact that the board could be reconstructed from this information makes it no more a world model than a set of memorized routes makes a map. It fails structural relations, since nothing represents how the squares or patterns are arranged relative to one another. Its input contains no move-order information and therefore does not even entail a unique board state, meaning it cannot read predictions off the actual board. Finally, if even this network counted as having a world model, so would almost any feature-combining network, rendering the notion trivial and undermining the reason Othello-GPT is considered a significant example in the first place. See Appendix C for further discussion.

5 Results

5.1 Othello-MLP predicts legal moves nearly as well as Othello-GPT

In general, task performance does not reliably indicate use of a world model: a network with one can perform poorly if its model oversimplifies the task, and a network without one can perform well given the right heuristics. Some tasks might nonetheless be too complicated for heuristics, especially when the network must generalize. Is legal move prediction in Othello such a task? Li et al. (2023) point out that Othello-GPT's predictions are legal far more often than those of an untrained version of the same model: in our evaluation, 99.9% versus 16.3%. To establish a world model, however, one should compare with a strong model that does *not* use a world model, like Othello-MLP. Notably, Othello-GPT barely exceeds this baseline: 99.9% versus 98.2%.

The remaining gap in performance may be explained by Othello-GPT's far greater depth (8 to 1 if we just count the MLP layers) and its use of attention (64 heads in total), which can express a much richer set of heuristics. Its greater depth lets it learn nonlinear interactions among patterns: exclusive-or (the output depends on two patterns differing), exact counts (the output depends on a specific number of patterns being present), and gating (one pattern contributes to the output only when another is present or absent). Its attention lets it dynamically weight earlier moves by context, including the move number and its parity. The result might be a more effective collection of heuristics rather than a world model.

[1] A variant of Othello-MLP with the same hidden layer but trained on next-move prediction did not learn the task as well, perhaps because next-move prediction only indicates that a move was legal, not which flanking pattern made it so, and thus doesn't provide a sufficiently informative training signal.

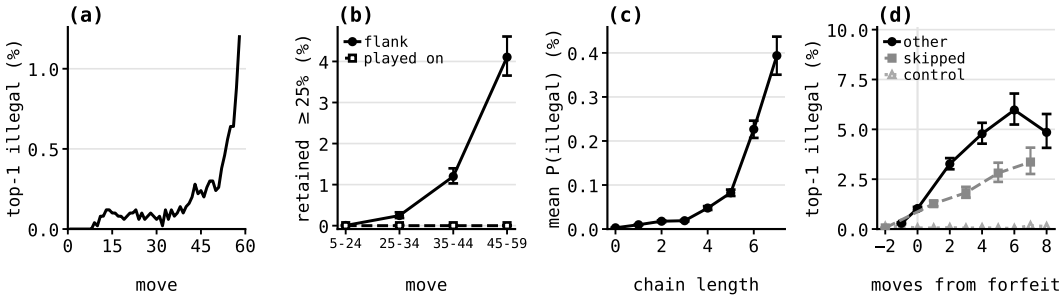


Figure 3: Behavioral analysis of Othello-GPT. **(a)** Error rate (percentage of top-1 predictions that are illegal) by move number. **(b)** Percentage of moves that were legal on the previous move but are now illegal and still retain $\geq$25% of their previous probability, split by whether a piece was played on the square (played on) or a move elsewhere disrupted the flanking pattern (flank). **(c)** Mean probability of an illegal move by the length of an adjacent opponent chain whose far end is uncapped. **(d)** Error rate by moves since a forfeit, for the player who skipped (skipped), the other player (other), and matched games without a forfeit (control).

The full output distribution tells a similarly mixed story. Othello-GPT places more than 5% probability on an illegal move in a sixth of games, which is consistent with its heuristics covering some positions better than others. Othello-GPT holds up better than Othello-MLP on top-3 and top-5 predictions (Appendix D), but some gap in performance is expected given that Othello-GPT has information about the move order and can implement a richer set of heuristics.

5.2 Systematic errors in Othello-GPT's predictions suggest heuristics

We just considered the frequency of Othello-GPT's errors, and found the evidence ambiguous. We now consider patterns in its errors, and find evidence of heuristics.

Newly illegal moves. There are two ways that a legal move becomes illegal: a piece is played on the square itself, or a move elsewhere disrupts the flanking pattern. Othello-GPT reliably drives the probability back toward zero when a piece is played on the square, but not always when the flanking pattern is disrupted elsewhere (see Fig. 3b). The heuristic explanation is straightforward: if a square is occupied, this can be inferred from a single move, and is therefore easily handled. But when a move becomes illegal through a different mechanism, this must be inferred from multiple moves, requiring many more heuristics, and there are sometimes gaps that allow the probability to remain high. This is hard to explain if the network derives its predictions from a unified model of the board: it could recompute the move's legality, and push its probability toward zero.

Long opponent chains. Othello-GPT assigns more probability to an illegal move when there is an adjacent chain of opponent pieces, even though no piece of its own caps the far end, and the amount increases with the number of opponent pieces in the chain (see Fig. 3c). The heuristic explanation is again straightforward: a row of the opponent's pieces running toward a square is partial evidence that a move there is legal, so the network has heuristics that raise its probability. Sometimes such partial evidence is strong enough to push the illegal move to the top. This is hard to explain if the network derives its predictions from a unified model of the board: it could check whether the far end is capped, and so would not treat the row's length as evidence of legality.

Forfeits. A forfeit occurs when one player cannot move and the other player moves again. As expected, Othello-GPT's error rate increases immediately after a forfeit. Surprisingly, the error rate keeps increasing on subsequent moves (see Fig. 3d). The heuristic explanation is once again straightforward: the network has heuristics to handle the forfeit and its immediate aftermath, but a forfeit breaks the regular turn-and-parity structure that its other heuristics depend on, so it interacts incorrectly with all of them. Forfeits are too rare, and their interactions too complex, for the network to have learned to compensate over the rest of the game. This is hard to explain if the network derives its predictions from a unified model of the board. A network accurately reconstructing the board immediately after a forfeit should not make errors on subsequent moves.

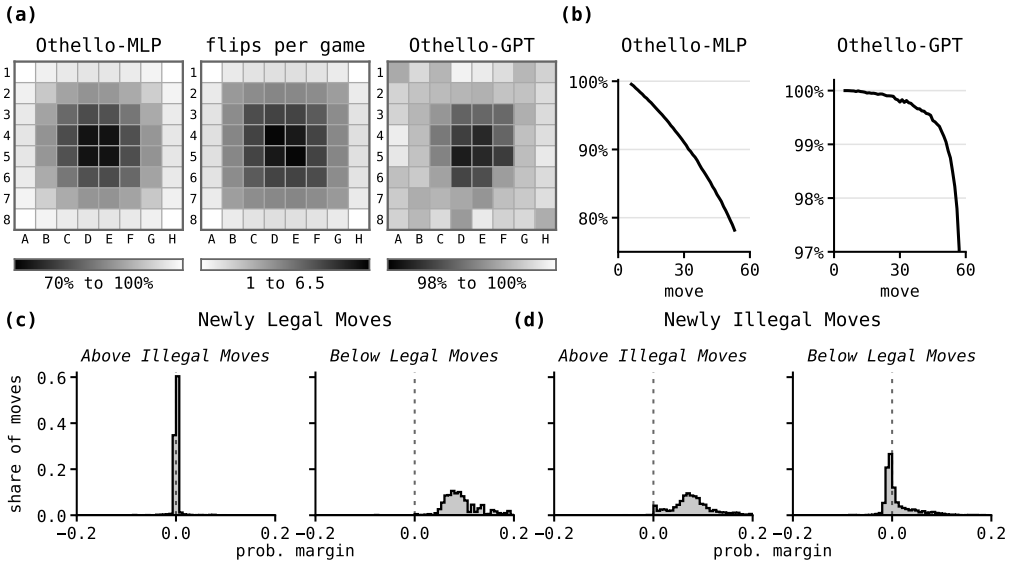


Figure 4: **(a)** Decoding accuracy by square for Othello-MLP and Othello-GPT (left and right), with the mean number of flips per game for each square (center). Othello-GPT is more accurate (note the different color scales), but both are worst on central squares, which flip most often. **(b)** Decoding accuracy by move number. Othello-GPT is again more accurate, but both decline over the game. **(c)** Interventions that make a move legal barely raise it above the most probable illegal move and leave it well below the least probable legal move. **(d)** Interventions that make a move illegal barely lower it below the least probable legal move and leave it well above the most probable illegal move.

We find the same patterns in Othello-MLP, though more severe (see Appendix E), confirming that these are signatures of heuristics.

5.3 The board can be linearly decoded from Othello-MLP

We now consider linear probes, where the evidence is again ambiguous. Nanda et al. (2023) showed that the board can be decoded from Othello-GPT's activations by a linear probe. For each of the 64 squares, the probe is a linear map from the residual stream (of width 512) to three classes: empty, occupied by the current player, or occupied by the opponent. This player-relative coding is central to the result. Li et al. (2023) originally probed for the absolute color of each square and found the board only weakly linearly decodable; Nanda et al. (2023) showed that classifying each square as belonging to the current player, the opponent, or neither makes it linearly decodable. At layer 6, the probe reaches 99.6% accuracy on a balanced test set, far above the 34.4% (near chance) of a probe trained on a randomized network. But the randomized network is again the wrong baseline. The relevant comparison is with a network that uses only heuristics, such as Othello-MLP, and its board is linearly decodable at 90.5% (Table 2). This decoding accuracy is near the theoretical limit, given that Othello-MLP's played (unordered) inputs are consistent with multiple board states (see Fig. 2c, Appendix F).

Let's further compare where probes fail. First, Othello-MLP's probe is least accurate on central squares and most accurate toward the edges, near-perfect in the corners. The reason is that central squares flip far more often, and a single-layer network has too few heuristics to track all the interactions that flip them (Fig. 4a). Second, its accuracy falls as the game goes on, as the number of interactions increases (Fig. 4b). Both patterns are signatures of heuristics.

Strikingly, Othello-GPT shows both patterns (Fig. 4a,b). Its probe is much more accurate overall, but it too is worst on central squares and degrades over the course of the game. Once we take the pattern of errors into account, Othello-GPT and Othello-MLP appear to differ in degree, not in kind.

5.4 CAUSAL INTERVENTIONS

So far, we have considered task performance and linear decoding, and found the evidence ambiguous. We now turn to the third and, according to many, most convincing kind of evidence: causal interventions. Nanda et al. (2023) intervened on the network's internal representation in a way that they argue corresponds to edits of its world model. Specifically, for a given square, the kind of linear probe described in the previous section has a readout direction corresponding to each state (empty, occupied by the player, occupied by the opponent). They steer the activation along that direction until the probe outputs the desired state. They then continue the forward pass and measure whether the move predictions changed as the edit predicted.

Interventions on Othello-GPT succeed by the standard metrics. We begin by replicating the approach and analysis in Nanda et al. (2023). We improve on their method by calibrating the intervention magnitude for each square, rather than using a constant one. To evaluate the efficacy of their intervention, they measure "average number of errors", i.e., if there are N legal moves in the counterfactual board state, how many of the network's N most likely moves are illegal before and after the intervention? The intervention increases probabilities of legal moves 98.5% of the time, and moves the number of errors from 1.37 before the intervention to 0.31 after.

But heuristics predict this too. On the heuristic interpretation, many separate heuristics rely on features that carry information about a square's state, each with its own direction in activation space. The probe succeeds by finding a single direction that partly overlaps with many of them. Steering along that direction moves each feature only to the extent of the overlap, so the heuristics shift the move probabilities in the right direction, enough to succeed on standard metrics, but not enough to change which moves the network treats as legal.

The magnitude of the shift is a stronger test. If an intervention really edits the network's representation of the board, a move that is newly legal according to that edit should receive as much probability as the other legal moves, and a move that is newly illegal should lose it. This is not what we find. Across all 60 squares, interventions shift the probability of a newly legal move by only 0.25%. This leaves its probability far below the other legal moves (see Fig. 4c). In positions reached by ordinary play where the square is already occupied by that piece, the moves it makes legal receive 12.4%. Interventions supposed to create newly illegal moves also fall short, removing only 2.15% of the 10.83% that the same kind of move typically carries (see Fig. 4d). The interventions shift the predictions in the right direction but do not change whether the network treats the moves as legal or illegal.

Othello-MLP shows the same pattern. Interventions on Othello-MLP move the probabilities in the right direction 95.9% of the time, and the shifts are *larger* than the shifts induced in Othello-GPT: 1.11% for newly legal moves and 3.76% for newly illegal moves. The top-N metric improves less, from 4.03 errors to 3.64, against 1.37 to 0.31 for Othello-GPT. This is a downstream consequence of the fact that Othello-MLP performs worse on top-N accuracy even without interventions (Appendix D).

Similar effects across other kinds of intervention. Intervention success in Othello-GPT and Othello-MLP varies sharply with the kind of edit, the square targeted, and the number of squares edited at once. This is again what we would expect if they are using heuristics (see Appendix H).

Overall, Othello-GPT does better than Othello-MLP on the standard metrics, but the difference appears to be in degree, not kind: in both networks the interventions improve the ranking of legal moves, barely change the probabilities, corrupt the rest of the board when scaled up, and vary in success with the kind and location of the edit. This is consistent with Othello-GPT being a collection of better heuristics.

5.5 TRAINED VARIATIONS OF OTHELLO-MLP CLOSE THE GAP TO OTHELLO-GPT

Othello-MLP's hidden units are constructed rather than learned. We trained four further MLPs by gradient descent. These networks are also plausibly collections of heuristics, and the best reach 98.9% legal-move prediction and 93.3% decoding accuracy, against Othello-GPT's 99.9% and 99.6% (Appendix G).

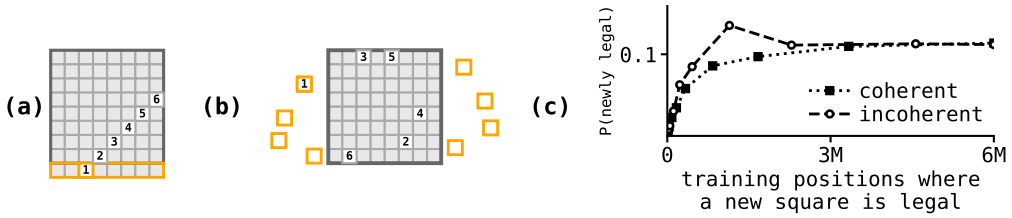


Figure 5: Transfer to extended versions of Othello. **(a)** Coherent extension: eight new squares form a ninth row, and flip rules extend along existing rows, columns, and diagonals. One of the rules is numbered. **(b)** Incoherent extension: the new squares are not located on the board, and flip rules involve arbitrary squares. One of the rules is numbered. **(c)** Probability assigned to newly legal moves over training, measured in positions where a new square is legal (Appendix I). Othello-GPT learns the incoherent extension at least as fast as the coherent one.

5.6 Transfer to coherent and incoherent extensions favors heuristics

We end with a kind of evidence that Li et al. (2023) and Nanda et al. (2023) do not consider: transfer learning. We ask how quickly Othello-GPT learns *extensions* of Othello. We construct two extensions, each adding eight new squares. In the *coherent* extension, the new squares form a ninth row, and the flip rules extend to them along the existing rows, columns, and diagonals (see Fig. 5a). In the *incoherent* extension, the new squares are not located on the board, and the flip rules governing them involve arbitrary squares scattered across it (see Fig. 5b). The coherent extension continues the board's 8×8 grid geometry onto the new squares; the incoherent extension does not.

If Othello-GPT uses a unified representation of the grid geometry, it should transfer that representation to the coherent extension and learn it faster. If it doesn't use a representation of that kind, there should be no such advantage. In fact, it learns the incoherent extension at least as fast as the coherent one (see Fig. 5c; see Appendix I for controls and other details). This is evidence that it does not use a unified representation of the board satisfying the structural relations condition.

6 Discussion

Summary. Whether neural networks, including language models, use world models has important consequences for how we understand them and how much we trust them. Othello-GPT is one of the most influential cases for emergent world models in neural networks. We argued that the evidence is weaker than it appears, since a network built from heuristics, Othello-MLP, scores nearly as well on the same measures. We also introduced an additional kind of evidence, transfer speed, and found that it favors a heuristic interpretation of Othello-GPT.

Limitations. None of this evidence rules out a world model. It is consistent with Othello-GPT using one to varying degrees, and with its using an unsystematic set of rules to generate predictions from that model. Our claim is that the evidence is ambiguous between these possibilities and heuristics defined over the move history.

Other tasks. Othello-GPT is only one of the networks said to have a world model; similar arguments have been offered for networks trained for chess, mazes, and causal inference, among other domains. Our findings suggest that strong baselines and detailed behavioral evaluations are essential for evaluating these other arguments.

Other capacities. Whenever a system completes a task, it can almost always be described as using sophisticated internal machinery—world models, Bayesian inference, hypothesis testing—or as doing something simpler. That choice has been debated longest in the study of animal cognition, and that field has developed a principle to guard against over-attribution, Morgan's Canon: a behavior should be explained by a more sophisticated capacity only when it cannot be explained by a less sophisticated one (Morgan, 1894). The principle has its critics (Sober, 1998), but it places the burden on those who would make the stronger attribution (Buckner, 2023). For Othello-GPT, that burden has not been met.

Reproducibility Statement

All of the code used in this paper is available here: `https://anonymous.4open.science/r/Othello_heuristic-F6FB/`

Ethics Statement

This work is a conceptual and empirical analysis of a small model trained on a board game, and does not raise any ethical concerns.

AI Disclosure

In this work, we used generative AI tools (Claude) to provide feedback on research methodology and experiments and to draft and edit parts of this paper. We did not use generative AI tools to generate synthetic datasets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide ingredients for proving mathematical claims, assist in writing proofs, propose or refine hypotheses, implement methods, assist with translation, clean or reformat datasets, or interpret results. We have reviewed all AI-assisted work: all experiments were designed and run by the authors, all reported results come from our own analysis pipelines, and every claim and passage was checked by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

References

Blaise Agüera y Arcas. Do large language models understand us? *Daedalus*, 151(2):183–197, 2022.

Pierre Beckmann, Matthieu Queloz, and Andre Freitas. World modeling in transformers, 2026. URL `https://arxiv.org/abs/2609.21748`.

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

Emily M Bender and Alexander Koller. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5185–5198, 2020.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, et al. Experience grounds language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8718–8735, 2020.

Cameron Buckner. Black boxes or unflattering mirrors? Comparative bias in the science of machine behaviour. *The British Journal for the Philosophy of Science*, 74(3):681–712, 2023. doi: 10.1086/714960.

Eivinas Butkus and Nikolaus Kriegeskorte. Causal discovery and inference through next-token prediction. *Advances in Neural Information Processing Systems*, 38:68213–68239, 2025.

Aviral Chawla, Galen Hall, and Juniper L Lovato. MetaOthello: A controlled study of multiple world models in transformers. In *Forty-third International Conference on Machine Learning*, 2026.

Xinyi Chen, Yifei Yuan, Jiaang Li, Serge Belongie, Maarten de Rijke, and Anders Søgaard. What if Othello-playing language models could see? In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 12598–12609, 2025.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175, 2021.

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging pretrained large language models to construct and utilize world models for model-based task planning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 79081–79094, 2023.

Wes Gurnee and Max Tegmark. Language models represent space and time. In *The Twelfth International Conference on Learning Representations*, 2024.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2733–2743, 2019.

Michael Igorevich Ivanitskiy, Alex F Spies, Tilman Räuker, Guillaume Corlouer, Chris Mathwin, Lucia Quirke, Can Rager, Rusheb Shah, Dan Valentine, Cecilia Diniz Behn, et al. Structured world representations in maze-solving transformers. *arXiv:2312.02566*, 2023.

Adam Karvonen. Emergent world models and latent variable estimation in chess-playing language models. In *First Conference on Language Modeling*, 2024.

Roman Kniazev and Nathanaël Fijalkow. Transformers linearly represent highly structured world models. *arXiv:2605.18847*, 2026.

Kai-Fu Lee and Sanjoy Mahajan. The development of a world class othello program. *Artificial Intelligence*, 43(1):21–36, 1990.

Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations*, 2023.

J Lin, J Schonbrun, A Karvonen, and C Rager. OthelloGPT learned a bag of heuristics. *LessWrong*, 2024. URL `https://www.lesswrong.com/posts/gcpNuEZnxAPayaKBY/othellogpt-learned-a-bag-of-heuristics-1`.

Aleksandar Makelov, Georg Lange, Atticus Geiger, and Neel Nanda. Is this the subspace you are looking for? An interpretability illusion for subspace activation patching. In *The Twelfth International Conference on Learning Representations*, 2024.

Raphaël Millière and Cameron Buckner. A philosophical introduction to language models, part II: The way forward, 2024.

Melanie Mitchell and David C Krakauer. The debate over understanding in AI's large language models. *Proceedings of the National Academy of Sciences*, 120(13):e2215907120, 2023.

C. Lloyd Morgan. *An Introduction to Comparative Psychology*. Walter Scott, London, 1894.

David E Moriarty and Risto Miikkulainen. Discovering complex Othello strategies through evolutionary neural networks. *Connection Science*, 7(3):195–210, 1995.

Neel Nanda, Andrew Lee, and Martin 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*, pp. 16–30, 2023.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. Arithmetic without algorithms: Language models solve math with a bag of heuristics. In *The Thirteenth International Conference on Learning Representations*, pp. 55939–55965, 2025.

Cory Paik, Stéphane Aroca-Ouellette, Alessandro Roncone, and Katharina Kann. The world of an octopus: How reporting bias influences a language model's perception of color. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 823–835, 2021.

Core Francisco Park. Convergent world representations and divergent tasks. In *Forty-third International Conference on Machine Learning*, 2026.

Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, 2018.

Jonathan Richens and Tom Everitt. Robust agents learn causal world models. In *The Twelfth International Conference on Learning Representations*, 2024.

Paul S Rosenbloom. A world-championship-level Othello program. *Artificial Intelligence*, 19(3): 279–320, 1982.

Aditya Singh, Zihang Wen, Srujananjali Medicherla, Adam Karvonen, and Can Rager. Automatically finding rule-based neurons in Othellogpt. *arXiv:2511.00059*, 2025.

Elliott Sober. Morgan's canon. In Denise Dellarosa Cummins and Colin Allen (eds.), *The Evolution of Mind*, pp. 224–242. Oxford University Press, New York, 1998.

Edward C Tolman. Cognitive maps in rats and men. *Psychological Review*, 55(4):189–208, 1948.

Keyon Vafa, Justin Y Chen, Ashesh Rambachan, Jon Kleinberg, and Sendhil Mullainathan. Evaluating the world model implicit in a generative model. *Advances in Neural Information Processing Systems*, 37:26941–26975, 2024.

Keyon Vafa, Peter G. Chang, Ashesh Rambachan, and Sendhil Mullainathan. What has a foundation model found? Using inductive bias to probe for world models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, pp. 60727–60747, 2025.

Archana Warrier, Dat Nguyen, Michelangelo Naim, Moksh Jain, Yichao Liang, Karen Schroeder, Cambridge Yang, Joshua B. Tenenbaum, Sebastian Josef Vollmer, Kevin Ellis, and Zenna Tavares. Benchmarking world-model learning with environment-level queries. In *Forty-third International Conference on Machine Learning*, 2026.

Yifei Yuan and Anders Søgaard. Revisiting the Othello world model hypothesis. *arXiv:2503.04421*, 2025.

Matej Zečević, Moritz Willig, Devendra Singh Dhami, and Kristian Kersting. Causal parrots: Large language models may talk causality but are not causal. *Transactions on Machine Learning Research*, 2023.

APPENDIX

A RICHER NOTIONS OF WORLD MODEL

There are richer notions of having a world model. On some views, a world model must include a representation of the task's dynamics—in this case, the rules for flipping pieces and skipping turns—so that the network can simulate the consequences of a move rather than merely recognize the current state. On others, a world model must be usable for planning, such as choosing a move by anticipating the opponent's replies, or for counterfactual reasoning, such as working out what the board would be had a different move been played. These richer notions are more demanding, so any evidence that Othello-GPT fails the minimal notion is also evidence that it fails them. We focus on the minimal notion because it is the weakest notion on which the question is still interesting.

B OTHELLO-MLP

This section gives the full construction summarized in Subsection 4.2.

Hidden layer. Let $\mathcal{P}$ denote the set of flanking patterns, $|\mathcal{P}| = 960$, where each pattern $p \in \mathcal{P}$ is a triple $p = (\mathrm{sq}_p, \mathrm{dir}_p, \mathrm{len}_p)$ specifying a target square, a direction, and a flank length. For each p, we trained a decision tree T_p to predict, from a played (unordered) encoding of the move history, whether pattern p is present. Each tree T_p was capped at depth 15 and 50 leaves. Training was class balanced by weighting positive instances because they were only a small fraction of positions.

Each leaf ℓ of T_p corresponds to a conjunction of literals on the encoding,

$$c_\ell(x) = \bigwedge_{(i,k)\in J_\ell} \mathbb{1}[x_{i,k} = v_{i,k}], \qquad v_{i,k} \in \{0,1\},$$

for some index set $J_\ell \subseteq \{1,\dots,60\} \times \{1,2\}$ of features tested along the root-to-leaf path. We instantiated one hidden unit u_ℓ per leaf, with activation

$$u_\ell(x) = c_\ell(x) \in \{0,1\}.$$

Pooling the leaves of all 960 trees, $\mathcal{L} = \bigcup_{p\in\mathcal{P}} \mathrm{leaves}(T_p)$, gives a fixed hidden layer of

$$|\mathcal{L}| = \sum_{p\in\mathcal{P}} |\mathrm{leaves}(T_p)| \le 48{,}000$$

since each tree has at most 50 leaves; a tree stops earlier when no split further improves its fit. Othello-MLP had 47,061 units. Its units were constructed rather than learned.

Output. Using gradient descent, we then trained a dense linear readout from the hidden layer to one score per pattern,

$$z_p(x) = \sum_{\ell\in\mathcal{L}} w_{p,\ell}\, u_\ell(x) + b_p, \qquad \hat{y}_p(x) = \sigma\big(z_p(x)\big) \in (0,1),$$

where σ is the sigmoid, giving a probability-like score for each pattern. Othello-MLP has one output $\hat{y}_p$ for each of the 960 flanking patterns. The trees fix which conjunctions the hidden layer computes, but the readout is free to combine them however best predicts each pattern. Only this readout is trained; the hidden layer is not. The predicted move is the target square of the highest-scoring pattern,

$$\hat{m}(x) = \mathrm{sq}_{p^*}, \qquad p^* = \arg\max_{p\in\mathcal{P}} \hat{y}_p(x).$$

C OTHELLO-MLP USES A COLLECTION OF HEURISTICS

By construction, each hidden unit computes a fully interpretable conjunction of move features, for example that B5 has not been played and that C5 and D5 were both played by the opponent. The readout maps these to a vote for or against each flanking pattern. On its face, then, the network is simply a collection of heuristics. Four considerations support this reading. Each was mentioned in Section 4.3; we expand on them here.

Othello-MLP fails completeness. Each hidden unit detects a conjunction of move features; their outputs are linearly combined and passed through a sigmoid, one per flanking pattern. That is the whole network. At no point are the conjunctions assembled into a board, and at no point are the flanking patterns read off such a board. The predictions are derived from 47,061 independently computed features.

The information needed to reconstruct the board is still present, spread across those features, but that is not sufficient for a world model. After all, many collections of heuristics implicitly contain enough information to reconstruct a fairly complete description of the task: a set of turn-by-turn routes can suffice to reconstruct a map, yet following memorized directions is not using a map. More generally, what matters is not whether the board can be recovered from the network, but whether the network uses a representation of the board to derive its predictions. That requires assembling the information about each square into one representation, and that doesn't happen in Othello-MLP. The information stays distributed across move features.

Othello-MLP fails structural relations. Nothing in the network represents how the squares, or the patterns, are arranged into an 8×8 grid. Grant for the sake of argument that each hidden unit implicitly represents the spatial relations among the squares in its own conjunction; it still doesn't represent a relation between those squares and any others. Grant likewise that each output unit represents the relations among the squares in its own pattern; there is still no representation of relations between patterns, including which lie on the same line, which share a square, which sit adjacent. The patterns are the pieces of an unassembled jigsaw, with no representation of how, or whether, they fit together into a grid.

Othello-MLP's input does not entail a unique board state. The input contains no information about move order. As a result it does not determine the board: by move 53, roughly a thousand distinct positions are on average consistent with a given input (Fig. 2c). A network whose input does not determine the board cannot be reading its predictions off the actual board. It could in principle represent a distribution over the consistent boards, or guess; nothing in the construction does either.

If Othello-MLP has a world model, almost every network does. It is not clear how we could construct a cleaner example of a network using mere heuristics: each hidden unit is an interpretable conjunction of move features, and the network is nothing but a weighted combination of them. If this network has a world model, it is unclear how *any* network could be shown not to have one. Any network combining a bank of features would qualify: an image classifier combining edge and object detectors, a spam filter combining word frequencies, a fraud detector combining a transaction's amount, time, and location. The notion would then stop tracking what made it interesting, namely whether a network understands its task and how far we can trust it in unfamiliar situations, and the claim that Othello-GPT has a world model would be correspondingly trivial—true of almost any network that solves its task. But Othello-GPT is taken to matter precisely because, unlike these networks, it looks like a compelling case of a network using one.

D FULL DISTRIBUTION

This section expands on the full-distribution results in Section 5.1. Othello-GPT outputs a softmax distribution over the 60 squares. Othello-MLP outputs 960 independent sigmoids, one per flanking pattern, that we can interpret as the probability that the pattern is present on the board. We combine the 15–18 patterns naming a given square into a single score for that square using the greatest output for that square (argmax). The top-K results depend only on the ranking of these 60 scores. To calculate the probability on illegal moves, we renormalize the 60 scores to sum to one. This puts Othello-MLP at a disadvantage because it wasn't calibrated for this distribution.

Metric	Othello-GPT	Othello-MLP	Randomized Othello-GPT
Top-1 legal	99.9%	98.2%	16.3%
Top-3 legal	99.9%	94.6%	16.4%
Top-5 legal	99.9%	90.9%	16.5%
Games >5% on one illegal move	15.8%	100%	86.1%
Games >10% on one illegal move	8.7%	99.6%	0.1%
Games >5% on all illegal moves	16.7%	100%	100%
Games >10% on all illegal moves	10.5%	100%	100%

Table 1: Legal-move prediction and illegal probability mass on held-out synthetic games, moves 5–53 ($N = 392{,}000$ positions from 8,000 games, all three models scored on the identical game set). Top-K is the fraction of the top $\min(K, n_{\text{legal}})$ predicted squares that are legal. The next two rows report the fraction of games with at least one illegal move exceeding 5% and 10%. The last two rows report the fraction of games with at least one position where total illegal move probability exceeds 5% and 10%. The randomized column is the mean of three initializations.

Probability on illegal moves. A given set of heuristics will cover some positions better than others, so a network using heuristics should assign substantial probability to illegal moves in positions its heuristics cover poorly. Othello-GPT does: in 8.7% of games it places more than 10% probability on a single illegal move, comparable to what a legal move receives in ordinary play, about 12% (Section 5.4). Othello-MLP shows the same pattern in more extreme form, at 99.6% of games. By this metric, Othello-GPT's behavior is suggestive of heuristics.

Probability on all legal moves. Following Li et al. (2023), Section 5.1 measures task performance by whether the move assigned the most probability is legal. This ignores the probability assigned to other legal moves. We therefore ask what fraction of a network's k most probable moves are legal. When a position has fewer than k legal moves, we just consider those moves. Othello-GPT's top-3 and top-5 rates both round to 99.9%, matching its top-1 rate, while Othello-MLP's fall to 94.6% and 90.9%. By this metric, Othello-GPT's behavior is suggestive of a world model. But a richer set of heuristics might produce the same result. High top-1 accuracy requires detecting only one legal move per position, and most positions include at least one clear flanking pattern. High top-k accuracy requires detecting k, including hard cases requiring higher-order patterns in the move history. Those might be the 5–9% of cases where Othello-GPT succeeds and Othello-MLP fails. The gap is small enough that Othello-GPT may simply have better heuristics for those cases.

Taken together, the evidence from the full output distribution is ambiguous.

E Othello-MLP's pattern of errors

Othello-MLP makes the same kinds of errors as Othello-GPT, though they differ in degree (Fig. 6).

F Othello-MLP's decoded board

The input does not determine the board. Othello-MLP's input discards move order, so a single input is compatible with many boards. We estimated how many: For moves 5–53 we sampled 100 movesets and, for each, drew 2,000 random orderings, keeping those that form a legal game; Fig. 2c plots the resulting mean number. The ambiguity increases across moves: a moveset of 5 moves is consistent with 1.2 boards on average, one of 20 moves with 66.5, and one of 53 moves with 970.7. The mean across moves 5–53 is 322.0 boards per moveset (median 125, maximum 1,979). Because we sample orderings, these are lower bounds.

This imposes a ceiling on decoding. A probe reading Othello-MLP's hidden layer outputs one board, but, as we just saw, the input is consistent with many more. Scoring each board against every board consistent with the same input gives a ceiling: the accuracy of the board most consistent with all the other boards. The ceiling decreases with move number, from 99.85% at move 5 to 80.05% at move 53. The mean ceiling across moves 5–53 is 92.06%, meaning that an average of 5.1 of the 64 squares are wrong. Because we sample orderings, these ceilings are probably overestimates: the

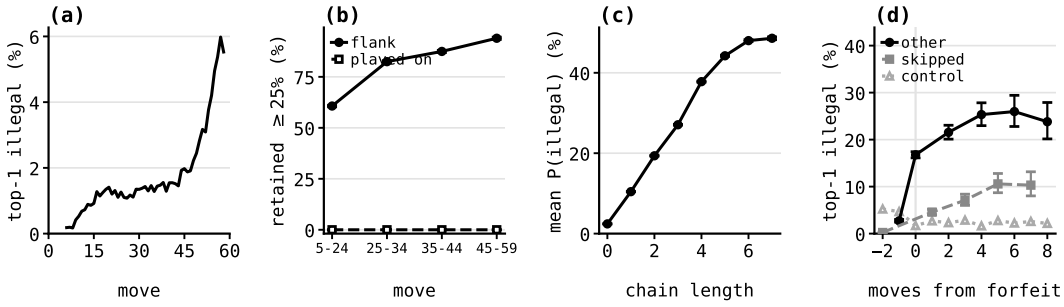


Figure 6: Behavioral analysis of Othello-MLP, matching the analysis of Othello-GPT in Fig. 3. **(a)** Error rate (percentage of top-1 predictions that are illegal) by move number. **(b)** Percentage of moves that were legal on the previous move but are now illegal and still retain $\geq$25% of their previous probability, split by whether a piece was played on the square (played on) or a move elsewhere disrupted the flanking pattern (flank). **(c)** Mean probability of an illegal move by the length of an adjacent opponent chain that is uncapped. **(d)** Error rate by moves since a forfeit, for the player who skipped (skipped), the other player (other), and matched games without a forfeit (control). Note the different y-axis scales from Fig. 3.

best board is fit to the sampled boards, and the boards we miss are likely to disagree with it, so the gaps reported below are, if anything, overstated.

Othello-MLP falls just below the ceiling. Table 2 compares the probe to this ceiling. The gap never exceeds 1.9 points and plateaus after move 30. The probe is close to the best available answer at every stage of the game.

move	consistent boards	ceiling	Othello-MLP	gap
5	1.2	99.85%	99.56%	−0.29
10	2.7	99.28%	98.38%	−0.90
20	66.5	96.33%	95.10%	−1.24
30	306.5	92.83%	91.04%	−1.79
40	540.5	88.02%	86.24%	−1.77
50	760.4	82.25%	80.37%	−1.88
53	970.7	80.05%	78.17%	−1.87
pooled	322.0	92.06%	90.49%	−1.57

Table 2: Board-state decoding against the ceiling set by input ambiguity, moves 5–53. "Ceiling" is the accuracy of the best single board, scored against every board consistent with the same input.

G TRAINED VARIANTS OF OTHELLO-MLP

Section 5.5 reports four further MLPs, trained by gradient descent. This section describes them and explains why we take them, like Othello-MLP, to be collections of heuristics.

Architecture and training. Each network has a single hidden layer and, like Othello-MLP, outputs one sigmoid score per flanking pattern, trained with a per-pattern binary cross-entropy loss. Unlike Othello-MLP, its input-to-hidden weights are learned rather than constructed from decision trees. We trained hidden widths of 512 and 4,096. Also unlike Othello-MLP, which approximates a mine/theirs encoding by alternating the "played" encoding between even and odd moves, we kept the encoding fixed and trained two networks of each width, one for even and one for odd positions. Because both approaches rely on parity, both misidentify the mover for the rest of a game after a forfeit.

Played encoding. Two networks use the same played (unordered) encoding as Othello-MLP (Appendix B). Because their input weights are learned, each hidden unit computes a weighted combination of input features rather than a simple conjunction. A unit might, for example, weight heavily that C5 and D5 were played by the opponent and that B5 was not, while weighting dozens of other features slightly. Such units are harder for us to interpret, but that is a fact about our understanding, not about what they compute, and it does not bear on whether the network is a collection of heuristics. The four considerations listed in Appendix C apply to these networks as well. In one respect it is arguably even more plausible that at least some of these networks are collections of heuristics: the H=512 variants using the played (unordered) encoding have roughly half a million parameters (so one million between the even and odd parity networks), compared to Othello-GPT's twenty-five million.

Signed one-hot encoding. The other two networks use a signed one-hot encoding: a 60×60 matrix over moves and squares, with each move's entry set to ±1 by the color of the player who made it. This encoding preserves move order, so the board is recoverable from the input in principle, though it is hard to see how a network with a single layer could recover boards that involve multiple flips. The other three considerations listed in Appendix C still apply. We therefore take these networks to be plausibly, though less obviously, collections of heuristics.

Noisy-OR. Each network assigns a score to the 960 flanking patterns. We use the highest-scoring pattern to predict the next move (argmax). But several patterns can make the same square legal, and taking only the highest ignores the others. As an alternative, we also treat the pattern scores as probabilities and combine those targeting the same square by taking the probability that at least one is present, assuming independence: $1 - \prod_p (1 - \hat{y}_p)$ over the patterns targeting that square. The predicted move is then the square with the highest combined score. Since combining independent pattern detectors introduces nothing resembling a world model, this aggregation narrows the gap to Othello-GPT without changing how we should interpret the network.

Results. Table 3 reports legal-move prediction and board-state decoding accuracy for each network.

	Legal moves						Decoding
	Argmax			Noisy-OR			
Model	Top-1	Top-3	Top-5	Top-1	Top-3	Top-5	Linear
Othello-GPT	99.9%	99.9%	99.9%	—	—	—	99.6%
Othello-MLP, played (unordered), H=47,061	98.2%	94.6%	90.9%	98.7%	95.8%	92.7%	90.5%
trained, played (unordered), H=512	97.9%	94.1%	90.4%	98.5%	95.5%	92.4%	90.4%
trained, played (unordered), H=4,096	98.5%	95.3%	92.1%	98.9%	96.5%	93.7%	90.9%
trained, one-hot (ordered), H=512	97.4%	94.1%	90.9%	98.0%	95.4%	92.5%	92.0%
trained, one-hot (ordered), H=4,096	98.0%	95.7%	93.4%	98.4%	96.5%	94.4%	93.3%
Randomized Othello-GPT	16.3%	16.4%	16.5%	—	—	—	34.4%

Table 3: Legal-move prediction and linear decoding accuracy on held-out synthetic games, moves 5–53 (N = 392,000 positions from 8,000 games). Top-K is the fraction of the top $\min(K, n_{\text{legal}})$ predicted squares that are legal. For the MLPs, the predicted move is either the target of the highest-scoring pattern (argmax) or the square whose patterns combine to the highest score (noisy-OR). Othello-GPT was probed at layer 6. H is the hidden width.

H Causal Interventions

Intervention success in Othello-GPT and Othello-MLP varies with the kind of edit, the square targeted, and the number of squares edited at once.

Intervention layer. As in Nanda et al. (2023), interventions on Othello-GPT were most effective at layer 4, so we intervene there, using a probe trained at that layer.

Intervention success varies with kind of edit. Across four kinds of edit (adding an opponent's piece, adding one's own piece, flipping a piece, and removing a piece), the probability an intervention adds to a newly legal move ranges from 0.05% to 0.38% in Othello-GPT ($7.6\times$) and from 0.54% to 1.85% in Othello-MLP ($3.4\times$). The probability it removes from a newly illegal move ranges from 0.86% to 3.75% in Othello-GPT ($4.4\times$) and from 1.85% to 7.86% in Othello-MLP ($4.2\times$) (Table 4).

	Othello-GPT		Othello-MLP	
Kind of edit	Newly legal	Newly illegal	Newly legal	Newly illegal
Add opponent's piece	+0.19%	−3.03%	+0.69%	−2.53%
Add own piece	+0.38%	−3.75%	+1.37%	−2.81%
Flip a piece	+0.05%	−0.86%	+0.54%	−1.85%
Remove a piece	+0.38%	−0.96%	+1.85%	−7.86%

Table 4: Intervention effects by kind of edit.

Intervention success varies with targeted square. In Othello-GPT, the probability an intervention adds to a newly legal move varies by a factor of 7.4 between the 10th- and 90th-percentile squares, and the probability it removes from a newly illegal move by a factor of 2.2 (Table 5). In Othello-MLP, the probability added varies by a factor of 1.7.

	Median	10th pct.	90th pct.	Ratio	Min (square)	Max (square)
Promotion	0.146%	0.072%	0.537%	7.4×	0.020% (B8)	1.562% (E6)
Suppression	2.050%	1.402%	3.120%	2.2×	0.995% (F8)	3.603% (E6)

Table 5: Intervention effects on Othello-GPT across target squares. Ratio is the 90th- over the 10th-percentile square. Promotion is the probability an intervention adds to a newly legal move; suppression is the probability it removes from a newly illegal move.

Intervention success varies with number of edits. We measure success as the fraction of edited squares whose probe readout of Othello-GPT matches the intended state. It falls from 98.2% with one edit to 94.6% and 93.9% with two and three. Edits also change the probe readout of squares they do not target: 0.45 squares on average with one edit, rising to 0.95 and 2.28 with two and three. That is a fivefold increase for three edits.

Larger interventions corrupt the board. Larger interventions produce larger shifts in Othello-GPT, but always at a cost. Consider interventions meant to make a move illegal. At the calibrated magnitude, they leave 8.68% of the original 10.83%. At $4\times$ and $10\times$ that magnitude, they remove 86.5% and 93.4%. But the number of untargeted squares whose probe readout changes rises from 0.03 to 0.45 and then 3.78. None of the magnitudes we tested suppresses the move while leaving the rest of the board intact.

I TRANSFER EXPERIMENT

Question. If Othello-GPT represents the board's geometry, it should find it easier to learn an extension whose new squares continue that geometry than one whose new squares are attached arbitrarily. We ask whether it does.

Two extensions. We test this by creating two extensions of Othello, one that extends Othello's 8×8 geometry and one that does not (see Fig. 5a). In the coherent extension another row is added, creating a 9×8 board, and the flanking rules are those the geometry dictates: along the new row, up the columns, and along the diagonals. In the incoherent extension eight new squares are also added, but their flanking rules jump arbitrarily around the board, so the new squares cannot be coherently located on it (see Fig. 5b).

Number and length of new flanking rules. The rules in the two extensions are matched in number and in length, where a rule's length is the number of squares in its chain, counting the flanked squares together with the flanking disc that caps them; lengths range from 2 to 8. Othello has 960 flanking patterns. The coherent extension adds 122 more: 42 that involve only the new squares, running along the new row, and 80 that reach into the existing board up the columns and along the diagonals. The incoherent extension adds the same 122. The 42 among the new squares are replicated exactly, so that both extensions have the same number of patterns involving only new squares. The 80 reaching into the existing board are generated in matched pairs, so that for every coherent rule of a given length there is exactly one incoherent rule of the same length, except that it includes randomly chosen squares rather than squares along a line.

Controls on training. Games are generated by uniform-random legal play, as in the synthetic database used to train Othello-GPT. Each extension's training set is grown until it contains the same number of positions in which at least one new square is a legal move. Learning parameters are identical across both extensions.

Controls on evaluation. For each extension, the evaluation set is 5,000 positions in which at least one new square is legal, with at most one position per game, so the positions are independent of one another. We compare the two extensions only on positions with the same rule length and the same number of legal moves n_{legal}. Because the training games are generated by uniform-random legal play, a fully trained model would place $1/n_{\text{legal}}$ on each legal move, so matching n_{legal} ensures that the two extensions have the same target probability on the newly legal squares.

Measuring learning speed. We measure learning speed by the number of positions in which a new square is legal rather than by the number of games. Measuring by games would give the incoherent extension an unfair advantage because the coherent rules are anchored in a new row at the bottom of the board, and adjacent squares usually aren't occupied until late in a game. The incoherent rules draw their squares uniformly from the whole board, and those squares are occupied earlier, so the rules are applicable earlier and more often in a game. Equalizing the number of new-square-legal positions controls for this difference. Since both extensions are trained to the same number of such positions, Fig. 5c equivalently plots the probability assigned to new squares against the number of new-square-legal positions.

Results. Othello-GPT learns the incoherent extension at least as fast as the coherent extension. At the end of training the two extensions place indistinguishable probability on the newly legal squares (0.114 coherent, 0.112 incoherent), and the curves are within noise of each other throughout the second half of training.

Discussion. This result is evidence that Othello-GPT is not using the board's 8×8 geometry, and thus that its representation does not satisfy the structural-relations requirement on world models.

Why we do not use Othello-MLP as a baseline on this experiment. This experiment does not need an external baseline because it compares Othello-GPT to itself: whether it learns the coherent extension faster than the incoherent extension. Moreover, it is trivial that Othello-MLP learns both extensions just as easily. It takes an equal number of positions to learn a decision tree for any new square, and once those trees are built into the network it takes an equal number of positions to learn the readout to the new output node. By design, each of these elements is learned independently, so the network's learning speed does not depend on geometrical relations between the squares.