WHEN GENERATIVE MODELS ACT AS SIMULATORS

Anonymous authors
Paper under double-blind review

ABSTRACT

A simulator summarizes an environment in a state that users can read and change. Generative models learn representations of the world without explicit specification, but offer no built-in way to edit that world's state. We investigate the conditions under which a generative model can act as a simulator through *editability*: whether changing a model's latent representation produces the next-step outputs we get by editing a simulator's state. Plainly stated, our goal is to find a mapping between a generative model's latent state and the simulator state, where the model is trained on observations alone and the mapping is learned afterward. We test two kinds of mappings, both learned from observations paired with simulator states. In one, we train probes to decode the simulator state from the model, then steer along them. In the other, we learn the mapping directly with an inverse map, which can succeed even where steering fails. We study both in two contrasting environments: the discrete board game Othello, and Rayworld, our own continuous world of moving discs seen through partial renderings, closer to physical scenes and video. We find that which mapping succeeds depends strongly on the environment, even though the state is decodable in every variant. In Othello, editability falls smoothly as the game is simplified, and even small rule changes have large consequences. In Rayworld, probes fail to steer along the simulator's continuous state, yet an inverse map sets the model from that state anyway. A model can therefore be driven by a simulator it may have never learned.[1]

1 INTRODUCTION

A simulator represents an environment by holding a state, advancing it by a rule, and rendering it into observations. In conventional simulators, this state is explicit and interpretable, allowing it to be set while preserving commitments to that environment and its rules (Todorov et al., 2012). Yet what these systems can express is constrained by what has been specified. Generative models,

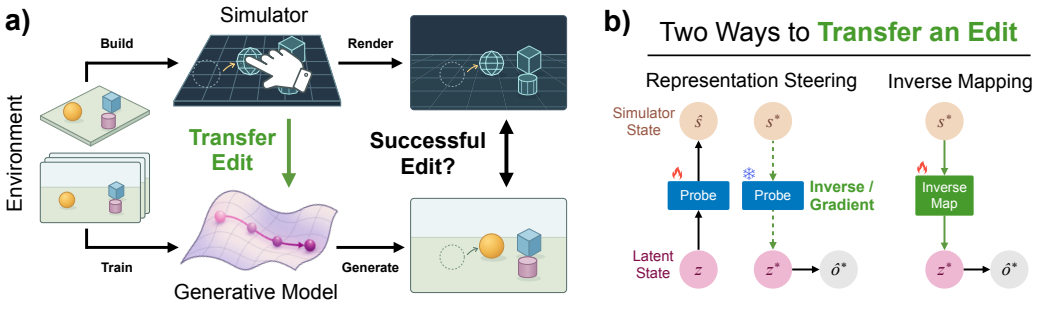

Figure 1: We examine when a generative model's latent state can be set from interpretable simulator variables. (a) We train a generator on an environment's observations alone, then test whether an edit to a ground-truth simulator transfers into it, so its generation matches the simulator's rendering. (b) In representation steering, we train a probe to read out simulator state, then invert it or follow its gradients to steer the latent state. In inverse mapping, we train a map from simulator state to latent state. Neither approach is trained on the observations its edits should produce. See fig. 2 for details.

[1]Code at `anonymous.4open.science/r/generative-models-as-simulators-C32D`.

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.

by contrast, learn from observations and can supply plausible details where that specification is incomplete (Brooks et al., 2024; Bruce et al., 2024). If an observer has seen only half of a room and then turns around, a generative model can propose what lies behind them, consistent with what they have seen (Lu et al., 2025). Their internal representations, however, offer no built-in means to set the environment's state; one must be found or engineered (Shen et al., 2020; Tewari et al., 2020).

Combining the two would provide interpretable control with learned detail. In forecasting (Lam et al., 2023) and planning (Hafner et al., 2025), this could pair predictive accuracy with the ability to specify precise conditions and counterfactuals. In virtual environments, a generative model could continuously extend the scene while a simulator keeps its state and rules persistent. Both require *editability*: the ability to set a model's latent state so that its next prediction follows from a target simulator state. Crucially, editability is a question about the simulator-like functionality we can access, not about whether the model represents or advances state as a conventional simulator does.

We investigate which conditions make a variable of the environment editable within an unsupervised generative model. Prior work shows that decodability alone does not reveal editability (Hazineh et al., 2023). To avoid confounds from the architecture, training, or evaluation, we hold all three fixed and vary only the training environment. Changing the rules governing a variable, or the observations through which it is revealed, then isolates their effect on its editability in the learned model.

We study two classes of environment: Othello, whose discrete board state is fully determined by its move history and rules (Li et al., 2023), and Rayworld, whose continuous disc positions are rendered into partial 1D observations, closer to physical scenes and video. We vary the game rules in Othello and the observation process in Rayworld, and also compare continuous and categorical descriptions of the discs' state. Both environments have known ground truth simulators, so the correct response to each edit can be checked directly. We score each edit by whether it moves the predicted observation toward what the environment produces after the edit, and by its overall fidelity.

Given observations paired with their simulator states, we compare two ways of editing a model. The first, representation steering, comes from mechanistic interpretability. Probes (linear or MLP) are trained to decode the state from the model's latent representations, revealing how the model encodes it, and editors then move the latent state along the probes' directions (Li et al., 2023; Nanda et al., 2023). A failed steering edit, however, shows only that the probes expose no control, not that the model cannot act as a simulator. The second, inverse mapping, tests this directly. A map trained on the same data, from simulator state to latent state, overwrites the latent state at a single layer. The model's remaining layers must then carry that write through to its output. If it succeeds where steering fails, we can still drive the model with simulator states, even if the model does not represent them as the simulator does.

Every model decodes its state well, yet in Othello editability falls smoothly as the rules are simplified. This holds even for a variant whose next move still depends on tracking the full board, where steering along the probes already degrades sharply. In Rayworld, the probes fail to steer unless their target is discretized, and steer best when it mirrors the observation, suggesting the model never forms a single abstract representation of where the discs are. Yet the inverse map succeeds from continuous positions, even though it predicts the model's latent state poorly. A generative model can therefore be driven from a simulator's state even when its probes cannot steer it.

Our main contributions are as follows:

(1) **An evaluation framework for editability across diverse environments;**

(2) **Evidence that simplifying an environment's rules can smoothly reduce its editability;**

(3) **A demonstration that a model can be driven from simulator states using inverse maps when probe-derived editors fail, even when these maps predict latent states poorly.**

2 Background and Related Work

World Models and Simulators. World models range from learned environments for agent training (Ha & Schmidhuber, 2018) to video generators presented as world simulators (Brooks et al., 2024; Bruce et al., 2024; NVIDIA, 2025a). Whether they represent environment state and use it in prediction (Li et al., 2025) is distinct from whether that state can be set. Vafa et al. (2024) test whether models predict the same futures for the same state, and different futures for different states.

For video generators, Kang et al. (2025) find accurate physical predictions within the training distribution but poor extrapolation, while What-If World tests whether prompted changes to physical variables produce the expected consequences (Cai et al., 2026).

Controllable generation often relies on additional structure: trained conditioning interfaces (Zhang et al., 2023; Wang et al., 2024; NVIDIA, 2025b), geometric manipulation of latent features for camera control without further training (Hou & Rupprecht, 2026), or generative priors that infer material properties for an explicit physical simulator (Zhang et al., 2024). The available edits thus depend on the structure provided. We instead ask whether a model trained only to predict observations can accept simulator state edits through its learned representations, producing the observations expected from a target simulator state.

Probing and Steering Learned Representations. In mechanistic interpretability, a common approach to studying learned world models is to decode environment variables and then intervene on their representations to test whether the model uses them. In Othello, Li et al. (2023) decode board states from transformers trained on move sequences and use probe gradients to guide predictions toward edited boards. Representing token ownership as mine/theirs permits linear decoding and edits through vector arithmetic (Nanda et al., 2023), and similar interventions produce moves legal under edited chess boards (Karvonen, 2024). Yet accurate board decoding can coexist with ineffective edits in late layers and shallow Othello models (Hazineh et al., 2023). Chawla et al. (2026) train models on pairs of rule variants and find that probes can guide edits across them, whereas we train a separate model per variant and hold the training and analysis fixed.

Decoding accuracy must be interpreted with care, since a probe can itself learn the structure being recovered (Hewitt & Liang, 2019). Amnesic probing tests whether a model uses certain decoded information by removing it from activations and measuring the effect on behavior (Elazar et al., 2021). In video encoders, Joseph et al. (2026) edit motion representations and use a separate probe to check whether they now indicate the target direction. Esmati et al. (2026) study whether video-model activations distinguish physically plausible from implausible events. In Atari models, Zhang (2026) also shows that interventions along probe directions change predictions. Because our simulators can be edited directly, we can instead compare a model's edited output with the output of the corresponding simulator edit. As in image editing (Shen et al., 2020) and language-model steering (Zou et al., 2023; Rimsky et al., 2024), success also depends on preserving properties outside the target (Canby et al., 2025). Our metrics measure both the targeted change and prediction quality.

A direct mapping into latent space can also set a model's state, without first finding a steerable representation. StyleRig maps face parameters and the current latent into StyleGAN using editing losses (Tewari et al., 2020), while our map uses state alone. Closer to our setting, Liu & Chen (2026) compare edited rollouts against simulator counterfactuals, but need counterfactual trajectories to train both model and edits. Our inverse mapping is fitted only on ordinary state–activation pairs, and we test whether it produces the intended output when probe-derived editors cannot.

Causal abstraction tests whether interventions on a model's internal representations correspond to interventions on variables in a higher-level causal model (Geiger et al., 2021), typically by swapping activations between inputs and, in distributed alignment search, learning the subspace where such swaps succeed (Geiger et al., 2025; 2024). We share this intervention-based criterion, but our high-level model is the simulator we want to drive the model with, not a hypothesis about its computation. Our editors are fit only to paired latent and simulator states, without counterfactuals or optimization for intervention success, which lets us test whether decodable representations are also editable.

3 Experimental Setup

Our experiments hold the model and its analysis fixed while the environment changes, so a difference between two rows of our tables is a fact about the environment. We use a common scheme for both environments, shown in fig. 2. A simulator holds a state s_t and renders it into an observation o_t, a move in Othello and a frame in Rayworld. The encoder G_E maps o_t and earlier latent states $z_{1:t-1}$ to z_t, the residual stream at every point, and the decoder G_D, the output layer, reads the last point to predict $\hat{o}_{t+1}$. An edit changes s_t to a target s_t^*, an editor proposes a matching latent state z_t^*, and G_D maps it to $\hat{o}_{t+1}^*$. The reference is o_{t+1}^*, rendered from s_t^* advanced one step, with the unedited o_{t+1} as a baseline. We vary the rules in Othello and what the observation reveals in Rayworld.

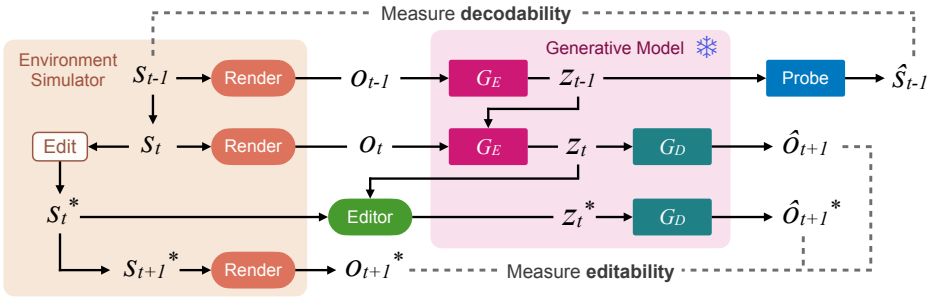

Figure 2: Overview of our evaluation procedure. The environment simulator renders state s_t into observation o_t. The frozen generative model encodes the observations into latent state z_t with G_E, and G_D maps it to the next-step prediction $\hat{o}_{t+1}$, where z_t holds the residual activations at every point and G_D reads the last. We measure **decodability** by training probes to recover simulator state from the latent state. To measure **editability**, we edit the simulator state to obtain a target s_t^*, use an editor to produce a corresponding latent state z_t^* and pass it through G_D for the edited prediction $\hat{o}_{t+1}^*$. We compare this prediction with o_{t+1}^*, rendered from the edited state one step forward, using the unedited observation o_{t+1} (not shown) and prediction as references (eq. (4)).

3.1 Othello

Othello is played on an 8×8 board. Players alternate placing tokens, and a legal move must enclose a line of opponent tokens, which then flip to the mover's color. We follow the setup of Li et al. (2023), whose generator draws every move uniformly from the legal set, so the optimal prediction is known exactly: a uniform distribution over the legal moves (section A.2). Following Nanda et al. (2023), instead of using black and white we describe the board as mine and theirs relative to the player about to move, the form in which it is linearly decodable.

The simulator state s_t is the board after each move, with every cell being empty, mine, or theirs. An edit flips the color of one token after a real 20-move opening and is judged against the legal moves of the board before and after the flip, according to each variant's own placement rules. The flipped board need not be reachable by any legal game, a property inherited from the implementation of Li et al. (2023). We split our results by reachability in section A.4.

Othello Variants. We introduce three variants of Othello. One modifies where a token may be placed, another removes the flipping of enclosed tokens, and the third does both. The two rules play distinct roles: placement keeps color relevant to predicting legal moves, while flipping makes the board hard to recover from the move sequence. In `adjacent-flip` a token may be placed anywhere adjacent to a token of its own color, including diagonally, with flipping active. Flips still change the legal moves but become rarer, at 0.27 flipped tokens per move against 2.2 in the `standard` game. `adjacent-noflip` uses the same placement rule with flipping off, so a placed token never changes color. `standard-noflip` keeps the standard placement rule without flipping, so each game fills the same checkerboard in a different order, and a token's color is fully determined by the cell it occupies. Figure 3 illustrates the setup and its variants.

3.2 Rayworld

Rayworld is a 2D world seen through a single fan of rays. An observer at the origin looks into a trapezoidal frustum in which two discs move at constant velocity for 40 frames. Trajectories in which a disc would hit a wall or the other disc are rejected, so the dynamics are exactly linear. Each frame is rendered into a 1D observation by casting 128 rays, spaced to cross the far edge of the frustum at equal intervals. A ray returns the reflectivity of the nearest disc it hits (0.4 or 0.8 and fixed per disc) or 0 if it hits neither, giving one noiseless pixel per ray. Depth is never observed directly. It enters only through the number of rays a disc covers, which falls with distance as the rays spread out, and through occlusion.

The Rayworld state $s_t \in \mathbb{R}^8$ holds the position and velocity of both discs and evolves as $s_{t+1} = A\, s_t$, where A advances each position by its velocity, and the frame $o_t = h(s_t) \in [0, 1]^R$ over R rays is

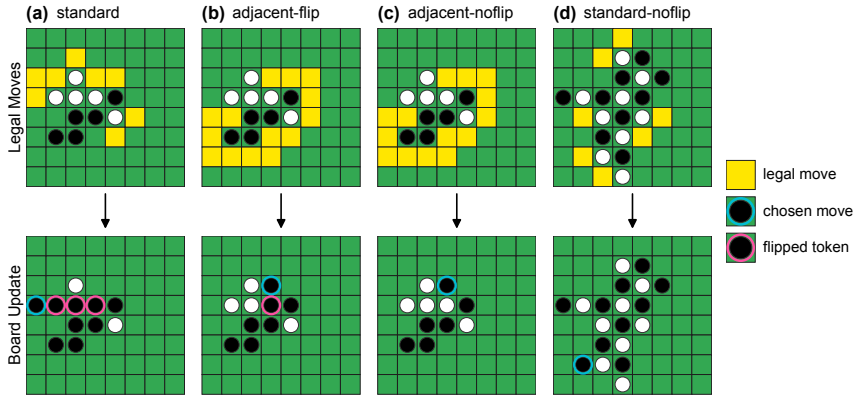

Figure 3: (a) In `standard` Othello, players alternate placing tokens that must enclose opponent tokens, which then flip color. (b) `adjacent-flip` instead allows placement anywhere adjacent (or diagonal) to one's own color, (c) `adjacent-noflip` also turns flipping off, and (d) `standard-noflip` keeps the original placement without flipping, so every game fills the same checkerboard in a different order. In all examples it is black's turn to move. See section 3.1.

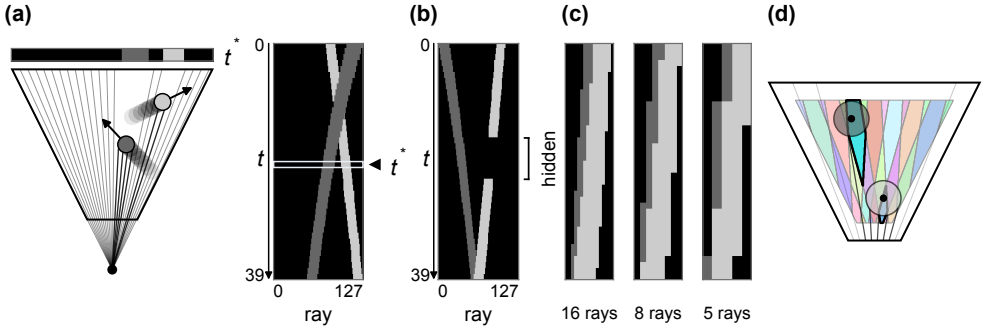

Figure 4: (a) In `standard` Rayworld, two discs move at constant velocity in a 2D frustum without colliding. Each frame is rendered into a 1D observation by casting rays into the frustum, shown stacked over time. (b) In `blink`, discs vanish for signaled periods of several frames, and (c) the `N-ray` family coarsens the observation by casting fewer rays. (d) The appearance target locates each disc with categorical bins, sets of positions that produce the same observation. See section 3.2.

given by the rendering function h. The best possible prediction of the next frame averages $h(A\, s_t)$ over all possible states, s_t, which fit the current observations. Nothing here forces a predictor to represent the state itself, and whether it does is what the probes and editors investigate. We probe the state in the Cartesian coordinates A acts in. An edit teleports a single disc, moving it to a random position inside the frustum that is clear of the walls and of the other disc, with both velocities left unchanged. Unlike an Othello edit, every edited state is a valid configuration of the world, with both discs inside the frustum and apart.

The state description we probe for can reveal what implementation the model has learned to achieve its task. To test this, we give Rayworld's continuous positions a categorical counterpart, matching the form of Othello's board. One option is to split the frustum into a uniform grid, with every bin labeled empty, bright disc, or dim disc, depending on which disc's center it contains. Alternatively, we let the observations draw the bins. In an observation, each disc lights a group of adjacent rays, described by its width and lateral position rather than its 2D coordinates. A small move changes this appearance only when the disc crosses into a different ray, so we treat all positions that produce the same appearance as one bin, and call the result the appearance target (fig. 4). An edit then moves a disc between bins, as an Othello edit changes one cell. This places the target in observation space rather than in positions. To keep the number of classes small, the target contains width and lateral position separately, rather than making each combination a separate class. Section A.8 compares it with the uniform grid, and we return to it in the discussion.

Rayworld Variants. We introduce two variants of Rayworld, each modifying what the observation reveals. In `blink`, a disc vanishes at random for 1 to 12 frames, with the frame before the blackout and its last hidden frame marked on an edge ray, so its position must be carried across the gap rather than read off the current observation. The `N-ray` family coarsens the observation by casting fewer rays, for $N \in \{5, 8, 16, 128\}$, to examine how the level of detail affects what a probe can find. The disc radius is doubled so a disc always covers at least one ray. We include `128-ray`, which matches `standard`'s ray count but uses the larger discs, to check that disc size alone does not explain our results. Figure 4 illustrates the setup and variants.

3.3 Model, Training, and Probes

Every model is the same eight-block transformer of width 512 with about 25M parameters (Vaswani et al., 2017), differing between environments only in its input and output layers. Rayworld uses linear input and output layers with a squared error loss, and Othello uses a token embedding and a softmax over moves with a cross entropy loss. To control for this difference in interface, we also train an `8-ray` Rayworld model with Othello's interface, reading and predicting whole frames as tokens (section A.5). Each model trains for 780,000 steps at batch size 256 on 18M sequences, with another 2M used for validation. The main results tables come from one seed per variant, and retraining each variant from three seeds at a shorter budget of 512,000 steps shows limited variation. We report the spread in section A.3.

A decoding probe reads the simulator state out of the model's latent state. The latent state z_t comprises the residual stream, the vector every block adds to and passes on, at every point (Elhage et al., 2021). Its value after block k is residual point k, with point 0 as the embedding. At every point we fit a linear probe, as a single affine map (Alain & Bengio, 2017), and an MLP probe with one hidden layer of 128 units, the shape used by Li et al. (2023). Both are fit on about 1.2M pairs of latent state and simulator state, with inputs standardized. Regression targets are fit by least squares and categorical targets by cross entropy. The targets are the mine/theirs board in Othello, the eight-dimensional state in Rayworld, and the appearance target of section 3.2, whose probes use a larger corpus of 200,000 sequences to match their larger output.

3.4 Editors

The editors differ only in how they produce z_t^*. With $f_{\mathrm{Lin}}(z) = Wz + b$ the linear probe, W^+ the pseudoinverse of W, and f_{MLP} the MLP probe at the edited residual point,

$$\text{PI:} \quad z_t^* = z_t + \alpha\, W^+ \big(s_t^* - f_{\mathrm{Lin}}(z_t)\big), \tag{1}$$

$$\text{GS:} \quad z_t \leftarrow z_t - \eta\, \nabla_{z_t} \mathcal{L}\big(f_{\mathrm{MLP}}(z_t),\, s_t^*\big), \tag{2}$$

$$\text{IM:} \quad z_t^* = g(s_t^*). \tag{3}$$

The first two editors steer z_t through the same probes that decode the state, so their success or failure indicates what those probes have found. Pseudoinverse injection (PI) makes the smallest change to the standardized latent state that moves the linear probe's reading onto the target, scaled by α so that $\alpha = 1$ lands exactly, and writes once at a single residual point. Gradient steering (GS) instead takes 100 gradient steps of size η on the latent state, minimizing the MLP probe's loss $\mathcal{L}$ toward the target while weighting entries that should not change by 0.2. Following Li et al. (2023), it writes at a starting point and again at every later point.

The inverse map (IM) runs the probe in reverse, testing whether the model can act as a simulator even when steering fails. We fit g from simulator state to latent state, an MLP of the same size and training setup as the forward probe. At edit time, we overwrite the latent state with $g(s_t^*)$ at one residual point. To rule out memorization, we compare against overwriting with the mean latent state of the ten nearest training states, which IM beats on every Rayworld variant, while in Othello the nearest neighbors never successfully edit (section A.9).

Each variant is scored on 1000 edit cases, each a held-out sequence with one variable changed at step 20. We skip cases whose edit leaves the observation unchanged and sweep each editor over residual points. For PI and GS we also sweep the step size (α and η, respectively).

3.5 METRICS

We introduce three metrics, one for decodability and two for editability, each defined relative to a reference so that environments with different outputs can be compared directly.

We measure decodability with Probe Skill $= 1 - L_{\text{probe}}/L_{\text{trivial}}$, a baseline-relative skill score in the sense of Murphy (1988), where L_{probe} is the probe's held-out loss and L_{trivial} that of a predictor that ignores the model. For continuous targets we use squared error with the training mean as the trivial predictor, so Probe Skill reduces to (out-of-sample) R^2. For categorical targets we use the error rate, with the most common class of each cell as the trivial predictor. We also fit the same probes to a randomly initialized model (Li et al., 2023) and to the raw observation sequences, separating what training put into the model from what a probe could extract regardless.

For editability we introduce two measurements, since an edit can implement the target change while degrading the rest of the output, or improve the output without implementing the change. Edit Index measures whether the edit moves the prediction in the right direction, measured only on the part of the output the edit should change: the rays whose values differ between the two worlds in Rayworld, and the cells whose legality the flip changes in Othello. It runs from -1, matching the unedited observation, to $+1$, matching the edited one. Edit Fidelity measures whether the output as a whole remains accurate, relative to the unedited prediction: 0 is no better than making no edit, negative values are worse, and 1 reproduces the edited observation exactly. With RMSE_Δ the error over the edit's support,

$$d_{\text{edit}} = \mathrm{RMSE}_\Delta(\hat{o}^*_{t+1}, o^*_{t+1}), \qquad d_{\text{uned}} = \mathrm{RMSE}_\Delta(\hat{o}^*_{t+1}, o_{t+1}),$$

$$\text{Edit Index} = \frac{d_{\text{uned}} - d_{\text{edit}}}{d_{\text{uned}} + d_{\text{edit}}}, \qquad \text{Edit Fidelity} = 1 - \frac{\mathrm{RMSE}(\hat{o}^*_{t+1}, o^*_{t+1})}{\mathrm{RMSE}(\hat{o}_{t+1}, o^*_{t+1})}. \tag{4}$$

We report each editor at its highest Edit Index among settings with Edit Fidelity of at least 0, falling back to the highest Edit Fidelity where none qualifies. Selecting on Edit Fidelity instead leads to the same conclusions (section A.11).

4 RESULTS

Tables 1 and 2 report decodability and editability for every variant, from one training run each. Retraining each variant from two further seeds gives a seed standard deviation of at most 0.004 for every trained probe's Probe Skill, and below 0.04 for the Edit Index of every editor that lands. Cells whose spread exceeds 0.1 carry a dagger, and all but IM's fidelity on `adjacent-flip` are failing editors whose selected setting flips between seeds at the fidelity cutoff (section A.3). Every model predicts its environment well before any probing or editing begins (section A.2).

Decodability. The simulator state is decodable in every variant, with the MLP probe reaching a Probe Skill of at least 0.88 and the linear probe close behind, so what separates the variants is the baselines. In Othello, training adds something the baselines lack only when flipping is on. Once flipping is removed the board can be read off the move sequence itself, so the `adjacent-noflip` model only matches its observation baseline, and every probe on `standard-noflip` scores a perfect 1.000, even on a randomly initialized model. In Rayworld, a randomly initialized model already comes within 0.03 of the trained one under the MLP probe on every variant except `blink`, where a hidden disc must be carried forward rather than located in the current frame.

Editability. Every editor edits `standard` Othello, consistent with Li et al. (2023) and Nanda et al. (2023), each carrying the predicted move distribution most of the way to the flipped board. Every change to the rules breaks this. With flipping off, no editor lands on either no-flip variant, even on `adjacent-noflip`, where legal moves still depend on token color. `adjacent-flip` is where the editors separate: IM still lands, while PI and GS reach at most $+0.35$ at low fidelity.

In Rayworld this separation becomes the rule. On continuous positions, the probe-derived editors fail on every variant while IM lands everywhere, including three further variants in section A.6. PI's write reaches its target in probe space, since at $\alpha = 1$ the linear probe reads the edited state exactly, yet the predicted frame does not move. A latent state that renders the teleported disc therefore exists

		Probe Type							
		Observation		Random init		Trained		Inverse Map MLP	
Class	Variant	Lin.	MLP	Lin.	MLP	Lin.	MLP	Best Point	Edit Point
Othello	`standard`	0.792	0.809	0.577	0.583	0.975	0.976	0.884	0.827
	`adjacent-flip`	0.888	0.914	0.674	0.700	0.947	0.970	0.968	0.959
	`adjacent-noflip`	0.988	0.983	0.581	0.588	0.988	0.990	0.978	0.855
	`standard-noflip`	1.000	1.000	1.000	1.000	1.000	1.000	0.947	0.855
Rayworld	`standard` (128 rays)	0.305	0.885	0.738	0.945	0.872	0.973	0.741	0.339
	`blink` (128 rays)	0.216	0.809	0.608	0.870	0.907	0.979	0.613	0.338
	`128-ray` (big discs)	0.377	0.919	0.859	0.970	0.914	0.981	0.652	0.321
	`16-ray`	0.360	0.907	0.890	0.949	0.892	0.965	0.554	0.516
	`8-ray`	0.329	0.877	0.883	0.919	0.886	0.932	0.585	0.580
	`5-ray`	0.277	0.829	0.855	0.871	0.850	0.883	0.617	0.469

Table 1: Decodability (higher is better, lower is worse), measured as Probe Skill at the trained model's best residual point, alongside random-initialization and observation baselines fit with the same probes on the same sequences. The inverse map columns give its R^2 in predicting latent state from simulator state, at its best residual point and at the point where it edits best. See section 4.

			Editor					
		Unedited	PI		GS		IM	
Class	Variant	Index	Index	Fid.	Index	Fid.	Index	Fid.
Othello	`standard`	−0.93	+0.82	0.70	+0.83	0.72	+0.81	0.62
	`adjacent-flip`	−0.96	+0.35†	0.17	−0.06	0.21	+0.66	0.49†
	`adjacent-noflip`	−0.96	−0.23†	0.19	−0.96	−0.01	−0.03†	0.26
	`standard-noflip`	−0.98	−0.95	0.00	−0.97	−0.01	−0.93	−0.01
Rayworld (continuous)	`standard` (128 rays)	−0.93	−0.10	0.04	−0.16	0.04	+0.59	0.66
	`blink` (128 rays)	−0.92	+0.02	0.01	−0.09	0.02	+0.51	0.55
	`128-ray` (big discs)	−0.94	−0.02	0.02	−0.10	0.03	+0.57	0.68
	`16-ray`	−0.91	+0.16	0.04	−0.10	0.10	+0.66	0.71
	`8-ray`	−0.90	+0.21	0.00	−0.07	0.12	+0.71	0.73
	`5-ray`	−0.89	+0.14	0.15	−0.10	0.14	+0.81	0.77
Rayworld (categorical)	`128-ray` (3736 bins)	−0.94	−0.31	0.02	+0.31	0.21	+0.64	0.71
	`16-ray` (77 bins)	−0.91	−0.06	0.04	+0.28	0.15	+0.82	0.71
	`8-ray` (30 bins)	−0.90	+0.38	0.05	+0.46	0.46	+0.87	0.73
	`5-ray` (14 bins)	−0.91	+0.51	0.30	+0.60	0.54	+0.91	0.75

Table 2: Editability (higher is better, lower is worse). For each variant and editor, we report the highest Edit Index across residual points and step sizes among settings with Edit Fidelity ≥ 0, with the Edit Fidelity at that setting. Where no setting reaches 0 we select for the highest Edit Fidelity. Scores are averaged over 1000 edit cases. Edit Index compares the edited prediction against the edited and unedited ground truth over only the rays or cells that change. Edit Fidelity measures whole-output improvement over the unedited prediction, from 1 for a perfect match to 0 for no improvement. Cells marked † have a seed standard deviation above 0.1 (section A.3). See section 4.

and can be written, but not along any direction the probes provide. Discretizing the target changes this. It raises GS on the `128-ray` model, and as the rays coarsen both probe-derived editors rise steadily, to +0.51 and +0.60 at 5 rays. Since the number of bins falls with the ray count, this trend cannot separate coarseness from target size. At 8 rays, evenly spaced bins that ignore the observation also improve on continuous positions, but the appearance bins improve most (section A.8). Figure 5 shows representative edits qualitatively.

5 Discussion

In Othello, editability falls smoothly as we simplify the game's rules, with each modification breaking it in a specific way. When the placement rule is exchanged for adjacency the probes no longer find steerable representations, yet the inverse map succeeds. This suggests the model represents the board in a form the probes miss. But this variant's settable state still extends beyond the boards it

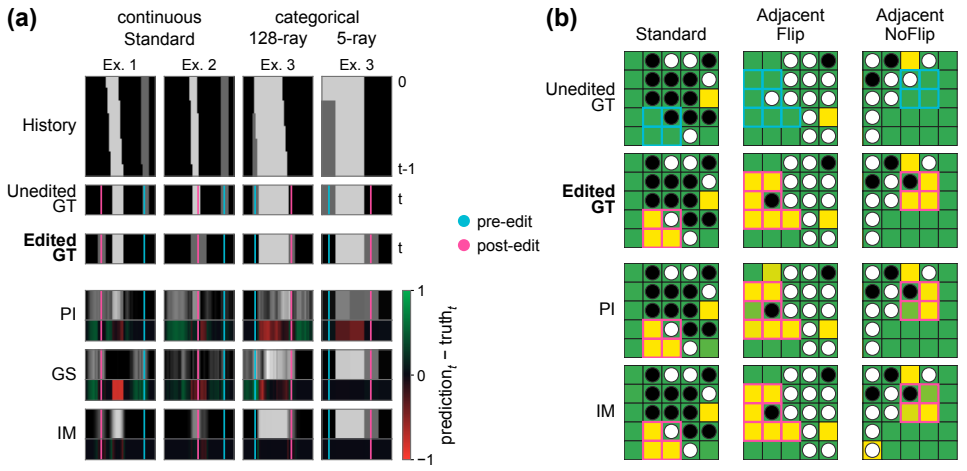

Figure 5: Qualitative examples of edited predictions in (a) Rayworld and (b) Othello, across environment variants and editors. Quantitative results in table 2. Cyan and magenta coloring highlight the disc or cells that changed as a result of the edit.

has seen, as the inverse map sets unreachable boards nearly as well as reachable ones. When token flipping is turned off, this generalization appears to degrade. `adjacent-noflip` can no longer be edited to flip a single token's color, where every target board is unreachable, but in section A.4 we show that it can be edited to reachable boards with two tokens flipped, using IM. With standard placements and no flipping, no editor can set the board at all. Access to the model's state thus narrows gradually as the game is simplified, rather than disappearing all at once.

In Rayworld, what matters is how the state is described. The probes find no steerable direction in continuous positions in any variant we tested, including with Othello's token interface (sections A.5 and A.6). Steering appears once the target is discretized, and works best when the state is defined in the observation space itself (section A.8). The inverse map instead sets continuous positions at almost every residual point, and still partially succeeds when the discs move as a rigid pair and an edit breaks their fixed spacing (section A.6). The inverse map thus lets us keep the simulator we prefer even where the model appears to have bypassed it. In Othello, by contrast, IM lands only at points 4 and 5 (section A.7), as if the model funnels its history through a board-like bottleneck that the Rayworld model never forms. Whether a model must form one depends on what the environment demands of prediction, resembling how a task may force a minimal sufficient statistic of the past (Shalizi & Crutchfield, 2001). Our Othello variants vary exactly this demand, which may be why editability falls as the game is simplified.

The inverse map's fit and its edits also separate in Rayworld, where it explains far less of the latent variance (R^2 of 0.32 to 0.58) yet lands everywhere. Because its edits still land despite the weak fit, much of the latent variance appears to have little bearing on the prediction. The edited predictions are visibly smoother than the model's own (fig. 5), consistent with the inverse map producing a simplified latent that omits some of this variance.

6 Conclusion

We investigated when a generative model can be driven as a simulator by setting its latent state, and found that the answer depends on the environment in ways decodability does not predict. In Othello, access to the model's state narrows gradually as the game is simplified. In Rayworld, probe-derived editors cannot steer the continuous state, yet an inverse map sets the model from it anyway, even while predicting its latent state poorly. Our editability concerns a single step, whereas a simulator also carries its state forward under consistent rules (see section A.1 for further limitations). Rewriting the history lets an edit persist through a rollout (section A.10), and the next step is to drive a model continuously from a simulator, or from different simulators at different times, and to discover which environments make that possible.

7 AI USE STATEMENT

Artificial intelligence was used to assist in the execution of this research. Large language models were used to develop code for experiments and simulations, aid in the writing and polishing of drafts, retrieve and discover related work, and assist with ideation.

8 REPRODUCIBILITY STATEMENT

We release the code for every environment, model, probe, editor, and metric at `https://anonymous.4open.science/r/generative-models-as-simulators-C32D`. The trained weights of every model, including the seed replicates, and the evaluation and edit datasets of every variant are at `https://huggingface.co/datasets/AnonymousCompute/generative-models-as-simulators`. The training corpora are not hosted, but the released scripts regenerate each one from its recorded seeds. Section 3 describes the setup, section A.12 lists the remaining settings, and section A.3 reports the spread over training seeds and probe refits.

REFERENCES

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2017. URL https://openreview.net/forum?id=ryF7rTqgl.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. URL https://openai.com/research/video-generation-models-as-world-simulators.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments, 2024. URL https://arxiv.org/abs/2402.15391.

Kunlin Cai, Rui Song, Jinghuai Zhang, Kaiyuan Zhang, Pranav Bodapati, Alicia Yu, Fnu Suya, Mohammad Rostami, Jiaqi Ma, and Yuan Tian. What-if world: A causal benchmark for general world models in embodied scenarios, 2026. URL https://arxiv.org/abs/2605.27589.

Marc E. Canby, Adam Davies, Chirag Rastogi, and Julia Hockenmaier. How reliable are causal probing interventions? In Kentaro Inui, Sakriani Sakti, Haofen Wang, Derek F. Wong, Pushpak Bhattacharyya, Biplab Banerjee, Asif Ekbal, Tanmoy Chakraborty, and Dhirendra Pratap Singh (eds.), *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 857–878, Mumbai, India, December 2025. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics. ISBN 979-8-89176-298-5. doi: 10.18653/v1/2025.ijcnlp-long.47. URL https://aclanthology.org/2025.ijcnlp-long.47/.

Aviral Chawla, Galen Hall, and Juniper Lovato. Metaothello: A controlled study of multiple world models in transformers, 2026. URL https://arxiv.org/abs/2602.23164.

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. doi: 10.1162/tacl_a_00359. URL https://aclanthology.org/2021.tacl-1.10/.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and

Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Parsa Esmati, Somjit Nath, Katja Hofmann, Derek Nowrouzezahrai, Samira Ebrahimi Kahou, and Majid Mirmehdi. The invisible hand of physics: When video diffusion models know more than they show, 2026. URL https://arxiv.org/abs/2606.05328.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 9574–9586. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/4f5c422f4d49a5a807eda27434231040-Paper.pdf.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah Goodman. Finding alignments between interpretable causal variables and distributed neural representations. In Francesco Locatello and Vanessa Didelez (eds.), *Proceedings of the Third Conference on Causal Learning and Reasoning*, volume 236 of *Proceedings of Machine Learning Research*, pp. 160–187. PMLR, 01–03 Apr 2024. URL https://proceedings.mlr.press/v236/geiger24a.html.

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, and Thomas Icard. Causal abstraction: A theoretical foundation for mechanistic interpretability. *Journal of Machine Learning Research*, 26(83):1–64, 2025. URL http://jmlr.org/papers/v26/23-0058.html.

David Ha and Jürgen Schmidhuber. World models. 2018. doi: 10.5281/ZENODO.1207631. URL https://zenodo.org/record/1207631.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, Apr 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-08744-2. URL https://doi.org/10.1038/s41586-025-08744-2.

Dean S. Hazineh, Zechen Zhang, and Jeffery Chiu. Linear latent world models in simple transformers: A case study on othello-gpt, 2023. URL https://arxiv.org/abs/2310.07582.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *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, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1275. URL https://aclanthology.org/D19-1275/.

Chen Hou and Christian Rupprecht. Probing into camera control of video models, 2026. URL https://arxiv.org/abs/2605.14815.

Sonia Joseph, Quentin Garrido, Randall Balestriero, Matthew Kowal, Thomas Fel, Shahab Bakhtiari, Blake Richards, and Mike Rabbat. Interpreting physics in video world models, 2026. URL https://arxiv.org/abs/2602.07050.

Bingyi Kang, Yang Yue, Rui Lu, Zhijie Lin, Yang Zhao, Kaixin Wang, Gao Huang, and Jiashi Feng. How far is video generation from world model: A physical law perspective. 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*, volume 267 of *Proceedings of Machine Learning Research*, pp. 28991–29017. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/kang25g.html.

Adam Karvonen. Emergent world models and latent variable estimation in chess-playing language models, 2024. URL https://arxiv.org/abs/2403.15498.

Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, Alexander Merose,

Stephan Hoyer, George Holland, Oriol Vinyals, Jacklynn Stott, Alexander Pritzel, Shakir Mohamed, and Peter Battaglia. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023. doi: 10.1126/science.adi2336. URL https://www.science.org/doi/abs/10.1126/science.adi2336.

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. *ICLR*, 2023. URL https://par.nsf.gov/biblio/10473173.

Kenneth Li, Fernanda Viégas, and Martin Wattenberg. What does it mean for a neural network to learn a "world model"?, 2025. URL https://arxiv.org/abs/2507.21513.

Yang Liu and Yuming Chen. Low-rank dynamics-effective latent carriers for counterfactual rollout in learned world models, 2026. URL https://arxiv.org/abs/2608.15156.

Taiming Lu, Tianmin Shu, Junfei Xiao, Luoxin Ye, Jiahao Wang, Cheng Peng, Chen Wei, Daniel Khashabi, Rama Chellappa, Alan Yuille, and Jieneng Chen. Genex: Generating an explorable world, 2025. URL https://arxiv.org/abs/2412.09624.

Allan H. Murphy. Skill scores based on the mean square error and their relationships to the correlation coefficient. *Monthly Weather Review*, 116(12):2417–2424, 1988. doi: 10.1175/1520-0493(1988)116⟨2417:SSBOTM⟩2.0.CO;2. URL https://journals.ametsoc.org/view/journals/mwre/116/12/1520-0493_1988_116_2417_ssbotm_2_0_co_2.xml.

Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. In Yonatan Belinkov, Sophie Hao, Jaap Jumelet, Najoung Kim, Arya McCarthy, and Hosein Mohebbi (eds.), *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 16–30, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.blackboxnlp-1.2. URL https://aclanthology.org/2023.blackboxnlp-1.2/.

NVIDIA. Cosmos world foundation model platform for physical ai, 2025a. URL https://arxiv.org/abs/2501.03575.

NVIDIA. Cosmos-transfer1: Conditional world generation with adaptive multimodal control, 2025b. URL https://arxiv.org/abs/2503.14492.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.828. URL https://aclanthology.org/2024.acl-long.828/.

Cosma Rohilla Shalizi and James P. Crutchfield. Computational mechanics: Pattern and prediction, structure and simplicity. *Journal of Statistical Physics*, 104(3-4):817–879, August 2001. ISSN 1572-9613. doi: 10.1023/a:1010388907793. URL http://dx.doi.org/10.1023/A:1010388907793.

Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the Latent Space of GANs for Semantic Face Editing . In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9240–9249, Los Alamitos, CA, USA, June 2020. IEEE Computer Society. doi: 10.1109/CVPR42600.2020.00926. URL https://doi.ieeecomputersociety.org/10.1109/CVPR42600.2020.00926.

Ayush Tewari, Mohamed Elgharib, Gaurav Bharaj, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zöllhofer, and Christian Theobalt. Stylerig: Rigging stylegan for 3d control over portrait images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, june 2020.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Keyon Vafa, Justin Y. Chen, Ashesh Rambachan, Jon Kleinberg, and Sendhil Mullainathan. Evaluating the world model implicit in a generative model, 2024. URL https://arxiv.org/abs/2406.03689.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Zhouxia Wang, Ziyang Yuan, Xintao Wang, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. Motionctrl: A unified and flexible motion controller for video generation, 2024. URL https://arxiv.org/abs/2312.03641.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023. URL https://arxiv.org/abs/2302.05543.

Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y. Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T. Freeman. Physdreamer: Physics-based interaction with 3d objects via video generation, 2024. URL https://arxiv.org/abs/2404.13026.

Xinyu Zhang. What do world models learn in rl? probing latent representations in learned environment simulators, 2026. URL https://arxiv.org/abs/2603.21546.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to ai transparency, 2023. URL https://arxiv.org/abs/2310.01405.

A APPENDIX

Contents

A.1 Limitations . . . 14
A.2 Predictive Loss of Trained Models . . . 14
A.3 Metric Variance and Confidence Intervals . . . 17
A.4 Comparing Reachable vs Unreachable Target States in Othello . . . 17
A.5 Matching GPT-style Prediction in Rayworld . . . 18
A.6 Additional Rayworld Variants . . . 19
A.7 Inverse Mapping Trends Across Residual Points . . . 20
A.8 Appearance Bins Against Evenly Spaced Bins . . . 22
A.9 Comparing Inverse Mapping to Nearest Neighbor Replacement . . . 22
A.10 Persistent Transformer Edits with History Rewriting . . . 23
A.11 Selecting for Edit Fidelity Instead of Edit Index . . . 24
A.12 Implementation Details . . . 25
A.13 Additional Qualitative Editability Visualizations . . . 26

A.1 LIMITATIONS

Our editability is a statement about only one step. A simulator also carries its state forward under consistent rules, and we test only whether a single write changes the next prediction as it should. Rewriting the history so that an edit persists through a rollout is one route, which we demonstrate in section A.10, and extending the metrics to rollouts is the natural next step. Every model here is one transformer at one depth, so some of what we find may be a property of that architecture or that depth rather than of the environments, and recurrent or diffusion models may behave differently. The environments also differ in difficulty in ways we cannot fully control, and a harder environment may be less editable for reasons unrelated to how its state is represented.

A.2 PREDICTIVE LOSS OF TRAINED MODELS

Every model predicts its environment nearly as well as the environment allows, so none of our editing results can be blamed on a weak predictor. Table 3 places each model's held-out loss between two references. The trivial predictor ignores the history and always makes the same guess, the overall frequency of each move in Othello and the average frame in Rayworld. The Bayes floor is the loss of the best possible predictor, which knows the environment's rules and is limited only by what the history leaves uncertain. The Othello floor is exact, since the generator draws each move uniformly from the legal moves. In Rayworld, a sequence is fixed by the discs' starting positions and velocities, but different starting states can light the same rays, so the frames seen so far do not pin down the next one. The best predictor averages the next frame over every starting state consistent with those frames, weighted by how likely the generator was to draw it. We approximate this average with sampled starting states, dropping each one as soon as it fails to reproduce an observed frame. The floor rises as the rays coarsen and leave more of the state undetermined. Every model closes at least 99% of the gap from the trivial predictor to its floor. Figures 6 and 7 show this accuracy directly. On `5-ray` a rollout can keep a disc on the wrong ray for several steps. In Othello, boards shaded by the model's predicted distribution are nearly indistinguishable from boards shaded by the true legal moves.

Class	Variant	Trivial	Bayes floor	Model	Excess
Othello (nats per move)	`standard`	4.094	2.011	2.029	0.018
	`adjacent-flip`	4.094	2.295	2.301	0.005
	`adjacent-noflip`	4.094	2.433	2.435	0.002
	`standard-noflip`	4.094	1.679	1.680	0.001
Rayworld (MSE $\times 10^{-3}$)	`standard` (128 rays)	72.6	0.80 ± 0.04	1.06	0.26 ± 0.04
	`blink` (128 rays)	66.0	0.98 ± 0.03	1.26	0.28 ± 0.03
	`128-ray`	92.4	0.75 ± 0.03	0.94	0.19 ± 0.03
	`16-ray`	97.2	3.76 ± 0.13	4.00	0.23 ± 0.13
	`8-ray`	100.7	5.62 ± 0.14	5.74	0.12 ± 0.14
	`5-ray`	105.5	6.86 ± 0.18	7.05	0.19 ± 0.18
Rayworld (`8-ray` tokens)	next-token CE (nats per frame)	5.176	0.441 ± 0.011	0.464	0.023 ± 0.011
	mean frame (MSE $\times 10^{-3}$)	100.7	5.62 ± 0.14	5.73	0.11 ± 0.14

Table 3: Held-out predictive loss of every trained model against the Bayes floor of its environment and a trivial history-blind predictor. Details in section A.2.

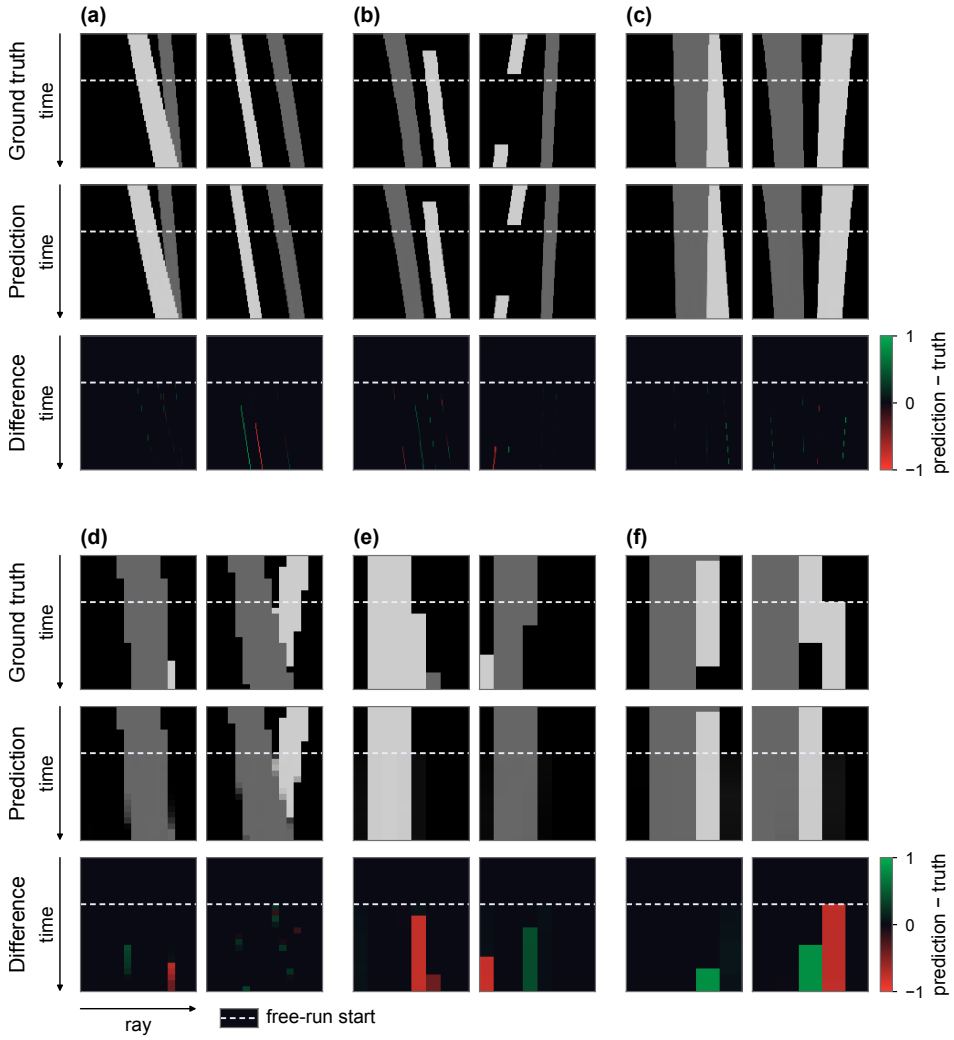

Figure 6: Randomly sampled demonstrations of Rayworld predictive accuracy in (a) `standard`, (b) `blink`, (c) `128-ray`, (d) `16-ray`, (e) `8-ray`, and (f) `5-ray`.

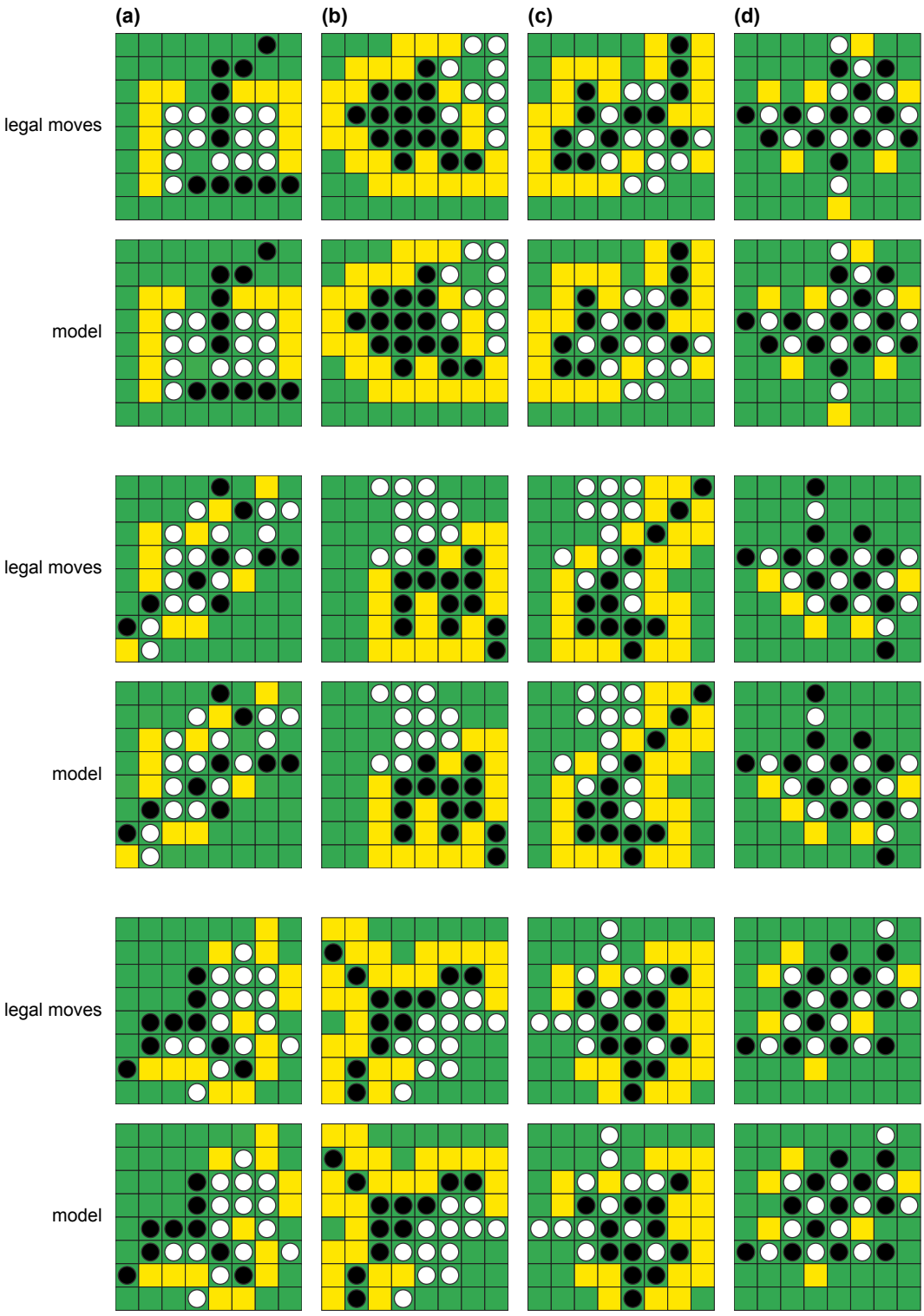

Figure 7: Randomly sampled demonstrations of Othello predictive accuracy in (a) `standard`, (b) `adjacent-flip`, (c) `adjacent-noflip`, and (d) `standard-noflip`.

A.3 Metric Variance and Confidence Intervals

The seed spread is small compared to the differences between variants. We retrain every variant from two further seeds and compare the three members of each set at 512,000 steps, namely the two new models and the main run's own checkpoint at that step, each scored exactly as in the main tables. Table 4 gives the standard deviation over the three seeds.

No Probe Skill varies by more than 0.004, and the Edit Index of every successful editor varies by at most 0.034. Each step of IM's increasing editability as Rayworld's rays coarsen is at least six standard deviations. Four cells exceed 0.1 and carry a dagger in table 2. Three are failing editors on the two adjacency variants, PI on both and IM on `adjacent-noflip`, and their spread comes mostly from selecting a different setting on each seed rather than from the models. Holding the main run's setting fixed, the Edit Index of both PI cells varies by only 0.05 to 0.07. IM on `adjacent-noflip` stays unstable even then, at 0.32, and on one seed no setting passes the cutoff, so it selects for highest fidelity instead and reports a much lower index. We keep the per-seed selection in table 4, since it is what rerunning the full pipeline would give. The fourth dagger, IM's Edit Fidelity on `adjacent-flip`, is not due to selection. All three seeds write at the same point with an Edit Index of 0.62 ± 0.03, while the fidelity ranges from 0.40 to 0.59.

Outside the daggered cells, the seed means sit within 0.05 of each Edit Index in table 2, which reports the same runs at 780,000 steps. Edit Fidelity can differ by up to 0.26 where the selected setting moves between the two checkpoints. PI on `standard` Othello, for example, is selected at a larger step size at 512,000 steps, with a mean Edit Fidelity of 0.48 against 0.70.

Refitting the probes on a fixed model has an even smaller spread. Refitting the linear probe with six to ten seeds, which change its initialization and held-out split, moves Probe Skill by at most 0.0007 and PI's Edit Index by at most 0.016 on `standard` Othello, the three `adjacent-flip` models, and the categorical target of the `8-ray` model. The MLP probes implementing GS were not refit.

		Probe Skill		PI		GS		IM	
Class	Variant	Lin.	MLP	Index	Fid.	Index	Fid.	Index	Fid.
Othello	`standard`	0.003	0.002	0.015	0.054	0.012	0.007	0.034	0.038
	`adjacent-flip`	0.000	0.001	0.245†	0.031	0.073	0.038	0.028	0.108†
	`adjacent-noflip`	0.001	0.001	0.196†	0.085	0.005	0.010	0.422†	0.077
	`standard-noflip`	0.000	0.000	0.001	0.001	0.002	0.002	0.026	0.002
Rayworld (continuous)	`standard` (128 rays)	0.002	0.000	0.042	0.010	0.027	0.026	0.005	0.020
	`blink` (128 rays)	0.002	0.001	0.060	0.016	0.018	0.020	0.012	0.007
	`128-ray` (big discs)	0.004	0.000	0.009	0.010	0.027	0.023	0.006	0.008
	`16-ray`	0.002	0.000	0.021	0.020	0.010	0.009	0.007	0.004
	`8-ray`	0.002	0.000	0.022	0.041	0.030	0.017	0.006	0.008
	`5-ray`	0.002	0.000	0.036	0.051	0.015	0.016	0.009	0.007
Rayworld (categorical)	`128-ray` (3736 bins)	0.001	0.001	0.024	0.020	0.014	0.019	0.003	0.005
	`16-ray` (77 bins)	0.002	0.000	0.014	0.019	0.006	0.065	0.002	0.004
	`8-ray` (30 bins)	0.000	0.000	0.006	0.027	0.014	0.009	0.005	0.001
	`5-ray` (14 bins)	0.000	0.000	0.017	0.029	0.012	0.016	0.003	0.000

Table 4: Seed standard deviation of the trained models' Probe Skill and of each editor's Edit Index and Edit Fidelity, over three training seeds per variant at 512,000 steps. Rows and settings as in tables 1 and 2. Cells marked † exceed 0.1.

A.4 Comparing Reachable vs Unreachable Target States in Othello

An Othello edit flips one token of a real board, and nothing guarantees that any game could reach the result. We therefore split the edit cases by whether some legal game ends on exactly the target board, which we decide by an exhaustive search over games (table 5). On `standard` Othello every editor edits both kinds about equally well. On `adjacent-flip`, IM lands on both but does better on reachable boards. Neither no-flip variant has any reachable flipped board, so the failures in table 2 there are all on unreachable targets.

To test whether a model that fails on single flips can be edited toward a reachable board, we flip two tokens of opposite color at once, which keeps each color's count (table 6). On `standard` and `adjacent-flip` the editors behave as they do for single flips. On `adjacent-noflip`, IM lands on the reachable pairs at $+0.88$ with an Edit Fidelity of 0.78, comparable to `standard` Othello, while PI and GS fail. On the unreachable pairs IM still moves the changed cells toward the target, at $+0.65$, but an Edit Fidelity of 0.08 shows it degrades the rest of the prediction. On `standard-noflip` no two-token edit is reachable, since every token's color there is fixed by its cell, and no editor lands.

				Editor					
			Unedited	PI		GS		IM	
Variant	Target	n	Index	Index	Fid.	Index	Fid.	Index	Fid.
`standard`	legal	441	-0.93	$+0.79$	0.70	$+0.84$	0.73	$+0.83$	0.62
	illegal	555	-0.93	$+0.84$	0.71	$+0.82$	0.71	$+0.79$	0.62
`adjacent-flip`	legal	335	-0.95	$+0.37$	0.27	$+0.09$	0.24	$+0.78$	0.51
	illegal	614	-0.97	$+0.35$	0.12	-0.12	0.19	$+0.61$	0.48
`adjacent-noflip`	legal	0	–	–	–	–	–	–	–
	illegal	1000	-0.96	-0.23	0.19	-0.96	-0.01	-0.03	0.26
`standard-noflip`	legal	0	–	–	–	–	–	–	–
	illegal	1000	-0.98	-0.95	0.00	-0.97	-0.01	-0.93	-0.01

Table 5: Editability on Othello split by whether the target board is legal, meaning some game under the variant's rules ends on exactly the flipped board with the same player to move, or illegal, meaning no game does. Both are decided exactly by an exhaustive search over games, and the cases it could not settle within its budget are left out (4 in `standard`, 51 in `adjacent-flip`). Each editor is read at its setting from table 2. Without flipping, a token keeps its placer's color and players pass only when they cannot move, so each color's count is fixed by the number of moves and no board with a single flipped token is legal. The illegal rows of both no-flip variants are therefore the full bench of table 2. Columns as in table 2.

				Editor					
			Unedited	PI		GS		IM	
Variant	Target	n	Index	Index	Fid.	Index	Fid.	Index	Fid.
`standard`	legal	1000	-0.92	$+0.74$	0.51	$+0.78$	0.73	$+0.78$	0.71
	illegal	1000	-0.93	$+0.72$	0.61	$+0.76$	0.69	$+0.76$	0.69
`adjacent-flip`	legal	1000	-0.94	$+0.31$	0.03	$+0.16$	0.15	$+0.71$	0.56
	illegal	1000	-0.95	$+0.18$	0.02	$+0.05$	0.08	$+0.64$	0.54
`adjacent-noflip`	legal	1000	-0.94	-0.04	0.18	-0.94	-0.00	$+0.88$	0.78
	illegal	1000	-0.95	-0.95	0.00	-0.95	-0.00	$+0.65$	0.08
`standard-noflip`	legal	0	–	–	–	–	–	–	–
	illegal	1000	-0.97	-0.97	0.00	-0.95	-0.01	-0.86	-0.08

Table 6: Editability on Othello for edits that flip two tokens, one black and one white, so that each color's count is unchanged. We generate 1000 reachable and unreachable cases for each variant, except for `standard-noflip`, as no such edit is legal there. Each editor is read at the setting the rule of table 2 picks within each group. Standard errors of the Edit Index are at most 0.02. Columns as in table 5.

A.5 Matching GPT-style Prediction in Rayworld

Othello and Rayworld models differ in their interface as well as their environment. The Othello model reads each move as a token and predicts a distribution over the next one, in the manner of a GPT-style language model. The Rayworld model reads and predicts frames as vectors under a squared error loss. To make sure this interface is not what separates the two environments, we train an `8-ray` Rayworld model with Othello's interface. With fixed reflectivities and no noise, every pixel takes one of three values depending on which disc it hits, or if it misses, so each frame is one of a finite set of patterns. Every pattern that occurs in the dataset becomes a token, giving a vocabulary of 421 frames plus a reserved unknown token. This allows us to match the setup exactly to the Othello model, with a token embedding in, a softmax over next frames out, and a cross-entropy loss,

trained with the same recipe but on the 8-ray frame model's sequences. Probes and editors act on the residual stream and carry over unchanged. Only scoring needs a choice, since the model outputs a distribution over frames rather than a frame. The frame-set reading scores the model's distribution over frames the way Othello scores its distribution over moves, with each world's true next frame in place of the set of legal moves. An edit scores $+1$ when all the probability moves onto the edited world's frame and -1 when it stays on the unedited one. The mean-frame reading averages the distribution into one frame and scores it exactly as the regular Rayworld models are scored.

The interface changes almost nothing. The token model decodes the state as well as the frame model, with every trained probe within 0.013 of it (table 7), and its mean frame predicts as accurately (table 3). Its editability also follows the same pattern under either reading (table 8). On continuous positions PI and GS fail while IM lands, and on the categorical target the probe-derived editors improve while IM lands. The frame-set reading gives lower scores to the editors that land, since it credits only probability on the exact edited frame, and a prediction that places the disc one ray off earns nothing. Whatever keeps the probes from steering Rayworld's continuous positions, it is not the vector interface.

		Probe Type							
		Observation		Random init		Trained		Inverse Map MLP	
State	Model	Lin.	MLP	Lin.	MLP	Lin.	MLP	Best Point	Edit Point
Continuous	8-ray frames	0.329	0.877	0.883	0.919	0.886	0.932	0.585	0.580
	8-ray tokens	0.329	0.877	0.876	0.885	0.899	0.935	0.611	0.611
Categorical	8-ray frames	0.477	0.935	0.922	0.935	0.935	0.943	0.618	0.615
	8-ray tokens	0.477	0.935	0.921	0.925	0.940	0.944	0.641	0.641

Table 7: Decodability of the 8-ray frame model and the 8-ray token model, which sees the same sequences as a vocabulary of whole-frame tokens and predicts a distribution over next frames. Columns as in table 1. The observation baseline is shared, since both models see the same frames.

			Editor					
		Unedited	PI		GS		IM	
State	Model	Index	Index	Fid.	Index	Fid.	Index	Fid.
Continuous	8-ray frames	-0.90	$+0.21$	0.00	-0.07	0.12	$+0.71$	0.73
	8-ray tokens, frame set*	-0.75	$+0.00$	0.25	-0.01	0.24	$+0.65$	0.70
	8-ray tokens, mean frame	-0.90	$+0.14$	0.19	-0.08	0.15	$+0.84$	0.75
Categorical	8-ray frames	-0.90	$+0.38$	0.05	$+0.46$	0.46	$+0.87$	0.73
	8-ray tokens, frame set*	-0.73	$+0.23$	0.44	$+0.35$	0.53	$+0.78$	0.80
	8-ray tokens, mean frame	-0.90	$+0.46$	0.32	$+0.48$	0.43	$+0.92$	0.75

Table 8: Editability of the 8-ray frame model and the 8-ray token model through the continuous and categorical state targets. Columns as in table 2. Rows marked $*$ score the token model's next-frame distribution against the frame sets of the edited and unedited worlds, the legal-set construction used for Othello. Mean-frame rows take the expectation of that distribution and score it as an 8-ray frame model with the ray-zone construction, choosing each editor's setting by the rule of table 2 on that readout.

A.6 Additional Rayworld Variants

Two further variants investigate whether the probes fail on continuous positions because the observation shows position too crudely. In standard Rayworld a disc lights every ray it covers at its full reflectivity, so a frame changes only when a disc's edge crosses a ray, and a model might predict well without ever tracking a precise position. The smooth variant shades each disc like a dome. A ray reads the disc's full reflectivity where it passes through the center and fades smoothly to zero toward the edge, following $(1-(d/r)^2)^2$ for a ray at distance d from the center of a disc of radius r. Any small shift of the disc then changes the brightness of every ray it lights, not just the rays at its edges, while the set of rays it covers stays the same. The obs5 variant keeps coarse rays but adds multiple viewpoints. Five 8-ray observers stand evenly spaced on a ring around a circular arena, and each frame joins their five views into 40 values. A disc hidden from one observer behind the other disc stays visible to the rest, so together the views pin down its position far better than any one

of them. Each variant trains the same model with the same recipe on data of its own, and we probe the same continuous Cartesian state.

Both variants make position easier to read, and neither makes it easier to steer (table 9). On `smooth`, training lifts the linear probe from 0.62 on a randomly initialized model to 0.96, the largest gap of any Rayworld variant, so the model does learn a linearly readable position. On `obs5` an MLP reads position from the raw frames alone at 0.98, and both probes on the trained model exceed 0.97. Yet PI and GS fail on both variants, with Edit Indices between -0.19 and $+0.02$ at an Edit Fidelity of 0.02, while IM lands on both at least as well as on `standard`. Therefore, Rayworld's outcomes are not due to a frame that reveals position too coarsely, since the smooth profile makes every lit ray track the exact position, nor are they due to a single viewpoint that leaves position ambiguous, since five viewpoints pin it down.

A third variant, `pair`, adds an invariant to the world. Its two discs move as a rigid pair, at a fixed distance of 2.0, with a shared velocity. However, the edit bench still teleports a single disc as usual, breaking the rigid pair. We expected these edits to fail, since the model never sees a disc move apart from its partner. In fact, the world state is now redundant, as the positions can be represented equally well with three dimensions (one 2D coordinate, plus an angle). Yet IM still lands from the Cartesian input, at $+0.41$ with an Edit Fidelity of 0.30 against $+0.59$ and 0.66 on `standard`, which suggests that the environment's invariants influence editability but do not dictate it.

(a) Probe Skill

	Probe Type							
	Observation		Random init		Trained		Inverse Map MLP	
Variant	Lin.	MLP	Lin.	MLP	Lin.	MLP	Best Point	Edit Point
`standard`	0.305	0.885	0.738	0.945	0.872	0.973	0.741	0.339
`smooth`	0.152	0.829	0.615	0.949	0.962	0.994	0.885	0.746
`obs5`	0.575	0.981	0.934	0.990	0.979	0.997	0.641	0.488
`pair`	0.502	0.890	0.800	0.969	0.922	0.994	0.765	0.373

(b) Edit Index and Edit Fidelity

		Editor					
	Unedited	PI		GS		IM	
Variant	Index	Index	Fid.	Index	Fid.	Index	Fid.
`standard`	-0.93	-0.10	0.04	-0.16	0.04	$+0.59$	0.66
`smooth`	-0.97	-0.09	0.02	-0.19	0.02	$+0.67$	0.74
`obs5`	-0.89	$+0.02$	0.02	-0.10	0.02	$+0.60$	0.66
`pair`	-0.95	-0.10	0.01	-0.15	0.01	$+0.41$	0.30

Table 9: Decodability and editability of the three additional Rayworld variants, `smooth` (the rendering profile falls off continuously from each disc's center), `obs5` (five `8-ray` observers spaced around the arena) and `pair` (the two discs move as a rigid pair), beside `standard` Rayworld for reference. All four use the continuous position target. Panel (a) reports Probe Skill and panel (b) reports Edit Index and Edit Fidelity.

A.7 Inverse Mapping Trends Across Residual Points

In Othello the inverse map successfully edits at only two residual points, and the way its fit changes with depth suggests why. Table 10 tracks editability of the `standard` Othello and `standard` Rayworld models through all nine residual points. In Othello, the first five blocks look like a compression phase. The inverse map's R^2 climbs from 0.39 at the embedding to 0.83 at point 5, as the board comes to determine more and more of the latent state, then dips to 0.73 at point 7. IM lands only at the top of this climb, at points 4 and 5, and at every other point its write leaves the prediction worse than no edit. A reconstruction test further supports this. Overwriting the latent state with the inverse mapping of the *unedited* board state, which should change nothing, keeps the output's error low only at points 4 and 5, and raises it substantially elsewhere. We read this as a bottleneck. The board is a sufficient statistic of the history for the next move, since the legal moves depend on the history only through it, and with flipping it is a complicated function of that history. The model appears to assemble the board in its early blocks, hold it near points 4 and 5, and then expand it into

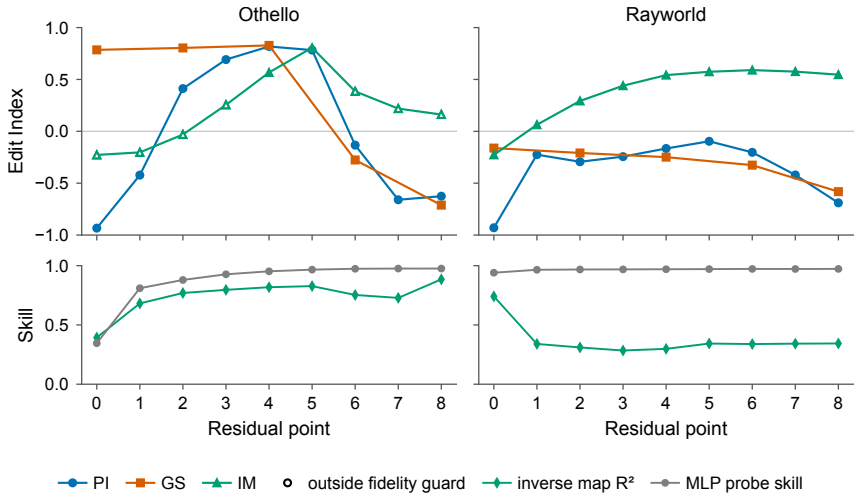

Figure 8: Editability across residual points on `standard` Othello (left) and `standard` Rayworld (right). The top row gives the Edit Index of each editor's best setting at each point, with GS placed at the point where it starts writing and hollow markers where no setting at that point reaches an Edit Fidelity of 0. The bottom row gives the inverse map's R^2 and the MLP probe's skill at each point.

the features of its prediction. A board written earlier meets blocks that are still gathering one from the moves, and a board written later lacks the detail those blocks have added since. The inverse R^2 alone does not locate the bottleneck, since it is higher at point 8 than at point 5 while IM fails there.

Rayworld shows no such bottleneck. The inverse map's R^2 is highest at the embedding, where the latent state is a direct function of the current frame, and falls to about 0.3 from point 1 on, where it stays. Yet IM improves the prediction at every point, and its Edit Index climbs steadily to $+0.59$ at point 6, at an Edit Fidelity of 0.66, and stays above $+0.54$ through point 8. No single depth holds a compact state that the rest of the network depends on, and the write works at most depths. The contrast fits what each environment demands of prediction. Othello's rules make the board both necessary and costly to compute, which gives a model reason to form it once and route everything through it, as a minimal sufficient statistic of the past (Shalizi & Crutchfield, 2001). Rayworld's frames already show most of the state, and its dynamics are linear, so its model faces no such pressure. The probe-derived editors follow a similar split in fig. 8. On Othello, PI and GS peak at point 4, just before IM, and fail from point 6 on, while on Rayworld they fail at every point.

	Othello `standard`				Rayworld `standard`			
	Probe	Inverse	IM		Probe	Inverse	IM	
Residual point	Skill	R^2	Index	Fid.	Skill	R^2	Index	Fid.
0 (embedding)	0.346	0.393	−0.23	−1.07	0.940	0.741	−0.23	0.25
1	0.810	0.682	−0.20	−0.99	0.965	0.340	+0.06	0.43
2	0.879	0.769	−0.03	−0.64	0.968	0.310	+0.29	0.54
3	0.927	0.796	+0.26	−0.14	0.969	0.284	+0.44	0.61
4	0.952	0.818	+0.57	0.51	0.970	0.298	+0.54	0.67
5	0.966	0.827	+0.81	0.62	0.971	0.343	+0.57	0.66
6	0.974	0.753	+0.39	−0.21	0.972	0.339	+0.59	0.66
7	0.975	0.728	+0.22	−1.03	0.972	0.342	+0.58	0.64
8	0.976	0.884	+0.16	−2.08	0.973	0.344	+0.55	0.60

Table 10: Decodability and inverse-map editability at every residual point of the `standard` Othello and `standard` Rayworld models. Probe Skill is the MLP probe's, the inverse R^2 is the held-out fit of g from simulator state to latent state at that point, and the IM columns give the Edit Index and Edit Fidelity of the inverse map overwrite $z^* = g(s^*)$ written at that point. Point k is the residual stream after block k.

A.8 Appearance Bins Against Evenly Spaced Bins

For the probe-derived editors, categorical targets seem to help even when their bins ignore the observation, and they help most when the bins are matched to it. We compare the appearance target on the `8-ray` model against evenly spaced grids over the frustum coordinates, matched to it at 30 bins and increased beyond it at 128. Every bin is labeled empty, bright disc, or dim disc (table 11). Every one of these grids improves GS from the continuous position target's Edit Index of -0.07 at an Edit Fidelity of 0.12, reaching $+0.35$ to $+0.39$ at Edit Fidelities of at least 0.33. Labeled the same way, the appearance bins improve it further, to $+0.59$ at an Edit Fidelity of 0.56, and their factorized form from the main tables also beats every uniform grid. Much of the gain is therefore in binning at all, and the rest in choosing bins the observation already defines. PI trends the same way, albeit less cleanly. None of this tracks Probe Skill, since the grids decode worse than the position target and still edit better than it.

Snapping each position to the center of its appearance bin while keeping the regression read-out returns GS to -0.05 at an Edit Fidelity of 0.07, the continuous result, even though the partition there is the one that works for the categorical state. The inverse map tends to edit well in every Rayworld categorical grid variant, but similarly does best when the grid matches the observation space.

	Probe Skill		Editor					
			PI		GS		IM	
Probe target (`8-ray` model)	Lin.	MLP	Index	Fid.	Index	Fid.	Index	Fid.
Position, regression	0.886	0.932	$+0.21$	0.00	-0.07	0.12	$+0.71$	0.73
Appearance (30 bins)	0.893	0.904	$+0.44$	0.11	$+0.59$	0.56	$+0.88$	0.73
Appearance, factorized	0.935	0.943	$+0.38$	0.05	$+0.46$	0.46	$+0.87$	0.73
Grid 6×5 (30)	0.466	0.535	$+0.30$	0.02	$+0.35$	0.33	$+0.64$	0.54
Grid 10×3 (30)	0.443	0.496	$+0.37$	0.09	$+0.35$	0.35	$+0.70$	0.57
Grid 16×8 (128)	0.156	0.212	-0.05	0.06	$+0.39$	0.34	$+0.73$	0.64
Snapped regression (30)	0.959	0.991	$+0.29$	0.00	-0.05	0.07	$+0.70$	0.69

Table 11: Probe targets on the Rayworld `8-ray` model, each a new probe fit on the same trained model with the same editors run through it on 1000 cases. Grids are evenly spaced bins over the frustum coordinates, as size-matched and resolution-matched controls for the appearance bins. Snapped regression replaces each position by the center of its appearance bin and keeps the regression pipeline. Columns as in tables 1 and 2.

A.9 Comparing Inverse Mapping to Nearest Neighbor Replacement

An inverse map fit on about 1.2 million pairs could succeed by memorizing them, recalling the latent state of a training example whose simulator state lies close to the target. We test this against a nearest-neighbor lookup with nothing to fit. Given a target simulator state, it finds the ten training examples whose simulator states are closest to it, takes the model's latent states on those examples, and writes their average into the model in place of the current latent state. Closeness is the number of differing cells between Othello boards, and the standardized Euclidean distance between Rayworld states. It draws on the same fit set as the inverse map, is swept over the same residual points, and is scored the same way. To compare how well each map fits, we also predict the latent states of held-out examples from their simulator states and report each map's latent R^2 at the point where the inverse map edits.

In every variant the inverse map beats the nearest neighbor at predicting latent states. It is also a better editor on every single Rayworld variant, while in Othello the nearest neighbor replacement never successfully edits. The inverse map therefore does more than recall nearby examples.

		Latent R^2		Editor			
				Inverse Map		Nearest Neighbor	
Class	Variant	IM	NN	Index	Fid.	Index	Fid.
Othello	`standard`	0.827	0.726	+0.81	0.62	−0.09	−0.81
	`adjacent-flip`	0.959	0.901	+0.66	0.49	−0.70	−0.27
	`adjacent-noflip`	0.855	0.750	−0.03	0.26	−0.82	−0.05
	`standard-noflip`	0.855	0.762	−0.93	−0.01	−0.90	−0.07
Rayworld	`standard`	0.339	0.068	+0.59	0.66	+0.33	0.24
	`blink`	0.338	0.059	+0.51	0.55	+0.27	0.22
	`128-ray` (big discs)	0.321	0.143	+0.57	0.68	+0.38	0.37
	`16-ray`	0.516	0.357	+0.66	0.71	+0.49	0.45
	`8-ray`	0.580	0.424	+0.71	0.73	+0.54	0.47
	`5-ray`	0.469	0.263	+0.81	0.77	+0.60	0.54

Table 12: The inverse map g against the nearest-neighbor control, which overwrites the latent state with the mean latent state of the nearest training states to the target. Latent R^2 is each map's held-out fit of the latent state at the residual point where the inverse map edits.

A.10 Persistent Transformer Edits with History Rewriting

Transformers make persistent edits hard, because they carry no state from one step to the next. An edit written at the last position changes the next prediction, but at the following step the model reads the original history again, and all that remains of the edit is the one predicted frame now appended to it. That frame shows the disc in its new place while every frame before it shows the disc on its old path, and the model sides with the history. On the `standard` Rayworld model, a single inverse-map write lands at the edit frame, at $+0.61$ with an Edit Fidelity of 0.64 on the 32 cases we use here, yet one step later the rollout has largely returned to the unedited world, at -0.63. Scoring rollouts would therefore mix two questions: whether the model reads the written state and whether it carries that state forward against a contradicting history. The main paper asks only the first, and scores every edit on the next prediction.

We introduce history rewriting to make the history agree with the edit. The edited state keeps the moved disc's velocity, and Rayworld's dynamics are reversible, so integrating backward from the target gives the counterfactual state at every earlier frame, with the other disc on its original path. We then rebuild the history one frame at a time. At each step the model reads the frames rewritten so far, we overwrite its latent state at the last position with g of the counterfactual state, and the model's prediction becomes the next rewritten frame. The first frame is kept, since nothing precedes it. The rollout then starts from the rewritten history. The simulator never renders an observation in this process. Every rewritten frame comes from the model, driven by written states alone.

The rewritten history successfully holds an edit through its rollout (fig. 9). For a transformer, the history is part of the state. An edit persists only when the history agrees with it, and history rewriting achieves this with nothing but the inverse map and the model itself. It points toward driving a generative model continuously from a simulator, writing the simulator's state into the model at every step and using the model's own predictions as the observations, so that the simulator supplies the state and the model the detail. Two obstacles remain. Firstly, the backward integration relies on Rayworld's simple, reversible dynamics. Secondly, the rewritten frames also drift from the true render, so errors may compound over longer horizons. A model that carries its state forward explicitly, as a recurrent model does, might avoid rewriting the history at all.

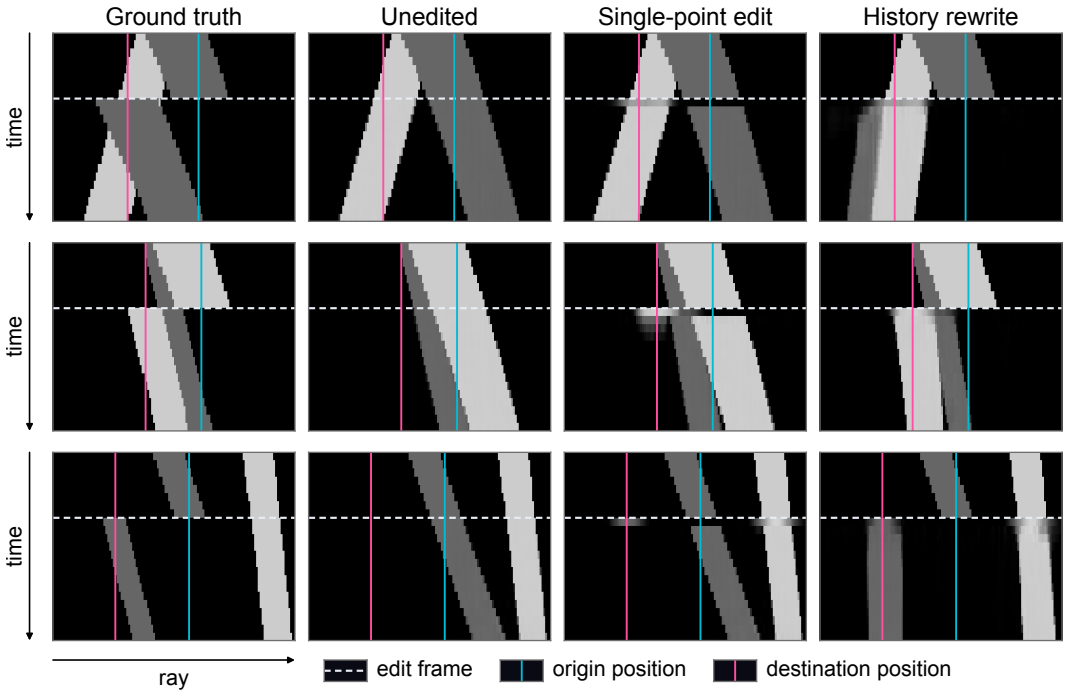

Figure 9: Persistent edits on `standard` Rayworld through history rewriting, one edit per row with time running downward. Every column shows the same last eight observed frames above the dashed line. Below it are the edited world's true continuation and 15-step rollouts from the unedited model, from a single inverse-map write at the edit frame, and from the rewritten history with the same write. Cyan and magenta lines mark the edited disc's origin and destination at the edit frame.

A.11 Selecting for Edit Fidelity Instead of Edit Index

Our main tables choose each editor's setting by its Edit Index, among the settings that leave the prediction at least as accurate as no edit. The rule rewards the strongest edit that does no harm, but it could obscure an editor sitting right at that cutoff, or make our conclusions depend on where the cutoff lies. Table 13 applies an alternative rule, choosing the setting with the highest Edit Fidelity outright and reporting the Edit Index reached there. This rule prioritizes accuracy over change. PI and GS mostly move to smaller steps, trading Edit Index for Edit Fidelity, while IM, which has no step size, moves by at most 0.10 in Edit Index.

Every conclusion of the main text survives the change of rule. All three editors still edit `standard` Othello. On `adjacent-flip` only IM lands, and on the no-flip variants no editor does. In Rayworld, PI and GS fail on continuous positions in every variant, while IM lands in all of them and rises as the rays coarsen. On the categorical target both probe-derived editors again rise as the rays coarsen, and IM lands throughout.

			Editor					
		Unedited	PI		GS		IM	
Class	Variant	Index	Index	Fid.	Index	Fid.	Index	Fid.
Othello	`standard`	−0.93	+0.82	0.70	+0.78	0.73	+0.81	0.62
	`adjacent-flip`	−0.96	+0.03	0.36	−0.06	0.21	+0.56	0.61
	`adjacent-noflip`	−0.96	−0.23	0.19	−0.96	−0.01	−0.03	0.26
	`standard-noflip`	−0.98	−0.95	0.00	−0.97	−0.01	−0.93	−0.01
Rayworld (continuous state)	`standard`	−0.93	−0.45	0.13	−0.59	0.11	+0.54	0.67
	`blink`	−0.92	−0.36	0.14	−0.49	0.13	+0.49	0.55
	`128-ray` (big discs)	−0.94	−0.47	0.16	−0.29	0.16	+0.55	0.68
	`16-ray`	−0.91	−0.15	0.19	−0.30	0.15	+0.66	0.71
	`8-ray`	−0.90	−0.16	0.22	−0.38	0.16	+0.71	0.74
	`5-ray`	−0.89	+0.03	0.26	−0.33	0.16	+0.81	0.77
Rayworld (categorical state)	`128-ray` (3736 bins)	−0.94	−0.63	0.10	+0.10	0.38	+0.63	0.73
	`16-ray` (77 bins)	−0.91	−0.42	0.14	+0.12	0.36	+0.78	0.72
	`8-ray` (30 bins)	−0.90	+0.06	0.35	+0.40	0.46	+0.84	0.74
	`5-ray` (14 bins)	−0.91	+0.38	0.42	+0.58	0.55	+0.90	0.75

Table 13: Table 2 changed to the highest-fidelity selection rule. For each editor we report the residual point and step size with the highest Edit Fidelity, and give the Edit Index reached at that setting, instead of the highest Edit Index among settings whose Edit Fidelity is at least 0. Rows, columns, and cases as in table 2.

A.12 Implementation Details

Environments. The Rayworld frustum spans depths 3 to 12, and its width at each depth equals that depth. Each disc's initial depth is drawn, then its lateral position across the frustum at that depth, and it moves in a random direction at a speed between 0.05 and 0.12 per frame. The depth, lateral position, velocity direction, and speed are all drawn uniformly. A sequence is redrawn if either disc ever leaves the frustum or the two discs' centers ever come within 1.6 diameters of each other, so every trajectory is a straight line at constant speed. Discs have radius 0.5, or 1.0 in the `N-ray` family, which casts $N + 2$ rays and drops the two that run along the walls. In `blink`, after the first three frames each disc begins a blackout with probability 0.05 per frame, lasting a geometric number of frames with mean 7, capped at 12. Othello games come from the generator of Li et al. (2023) under each variant's rules. A player passes only when they have no legal move, and a game ends when neither player can move. Each edit recolors one occupied square outside the four central ones, and a case is rejected if the flip leaves the legal moves unchanged or empty. Training, probe, evaluation, and edit sequences come from disjoint seed ranges.

Models and training. The transformer is the minGPT model of Li et al. (2023), with 8 heads, learned absolute position embeddings, and dropout of 0.1. It reads 39 frames in Rayworld and 59 moves in Othello, whose vocabulary has 61 tokens. We train with AdamW at a learning rate of 10^{-3} and weight decay of 10^{-4}, clip gradients at norm 1, and hold the learning rate constant after 2,000 warmup steps. Main runs use seed 0 and the replicates use seeds 1 and 2. Each model trains on a single NVIDIA RTX 4090 or 5090, in 8 to 12 hours for Rayworld and 19 to 26 hours for Othello.

Probes and inverse map. Probes are fit at every position of each sequence, with an 80/20 split by sequence into fit and held-out sets. Linear regression probes are solved in closed form. The MLP probes, the linear categorical probes, and the inverse map train with Adam at a learning rate of 10^{-3} and batch size 4,096, for 200 epochs, or 50 epochs on the larger corpus of the categorical Rayworld targets. The inverse map minimizes squared error in the standardized latent space. Its input is the eight-dimensional state in Rayworld, the one-hot mine/theirs board in Othello, and for the categorical target the one-hot bin labels together with both discs' velocities. The nearest-neighbor control averages the latent states of the 10 fit states nearest the target, by standardized Euclidean distance on continuous states and Hamming distance on categorical ones. The random-init baseline uses the same budget as the trained model's probes. The observation baseline instead reads the whole input history, a much wider input that overfits this budget and so understates what the input reveals, so we fit it on 250,000 Rayworld sequences, 200,000 for the categorical targets, or 170,000 Othello games, for 50 epochs, where its gap between fit and held-out skill nearly vanishes.

Across the runs in our tables the fit and held-out skills differ by at most 0.003 for the linear probes and 0.005 for the MLP probes, and the inverse map's two R^2 values agree to three decimals.

Editors and evaluation. Every editor writes only at the last position of the history. On continuous targets PI sweeps $\alpha \in \{0.1, 0.25, 0.5, 0.75, 1, 1.5, 2, 3, 5, 8, 12, 20, 35, 60, 100, 175\}$ and GS sweeps $\eta \in \{0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.35, 0.5, 0.7\}$. On categorical targets, Othello's included, PI sweeps $\alpha \in \{0.25, 0.5, 1, 1.5, 2, 3, 5, 10, 20, 35, 60, 100\}$ and GS sweeps $\eta \in \{0.05, 0.1, 0.2, 0.35, 0.7, 1.5\}$. GS starts at residual point 0, 2, 4, 6, or 8, while PI and IM are swept over all nine points. Edit Index is averaged over cases. Edit Fidelity divides errors that are each averaged over all cases, rather than averaging a ratio per case.

A.13 Additional Qualitative Editability Visualizations

The following figures show more edits in the style of fig. 5. In each Othello figure, every column is one randomly drawn edit for one rule variant. The top two boards show the legal moves before and after the flip, and the rest show the model's prediction after each editor's write, with shading that saturates at 2%. Outlines mark the flipped token and the cells whose legality it changes. Where no setting reaches an Edit Fidelity of 0 the figures show the setting with the highest Edit Index instead of the table's fallback. Each Rayworld panel shows one world under every variant, with single next-step predictions rather than rollouts, each above a strip of its error against the edited ground truth. We drew only teleports that visibly change the `5-ray` frame. The categorical IM is blank for `standard` and `blink`, which have no categorical inverse map.

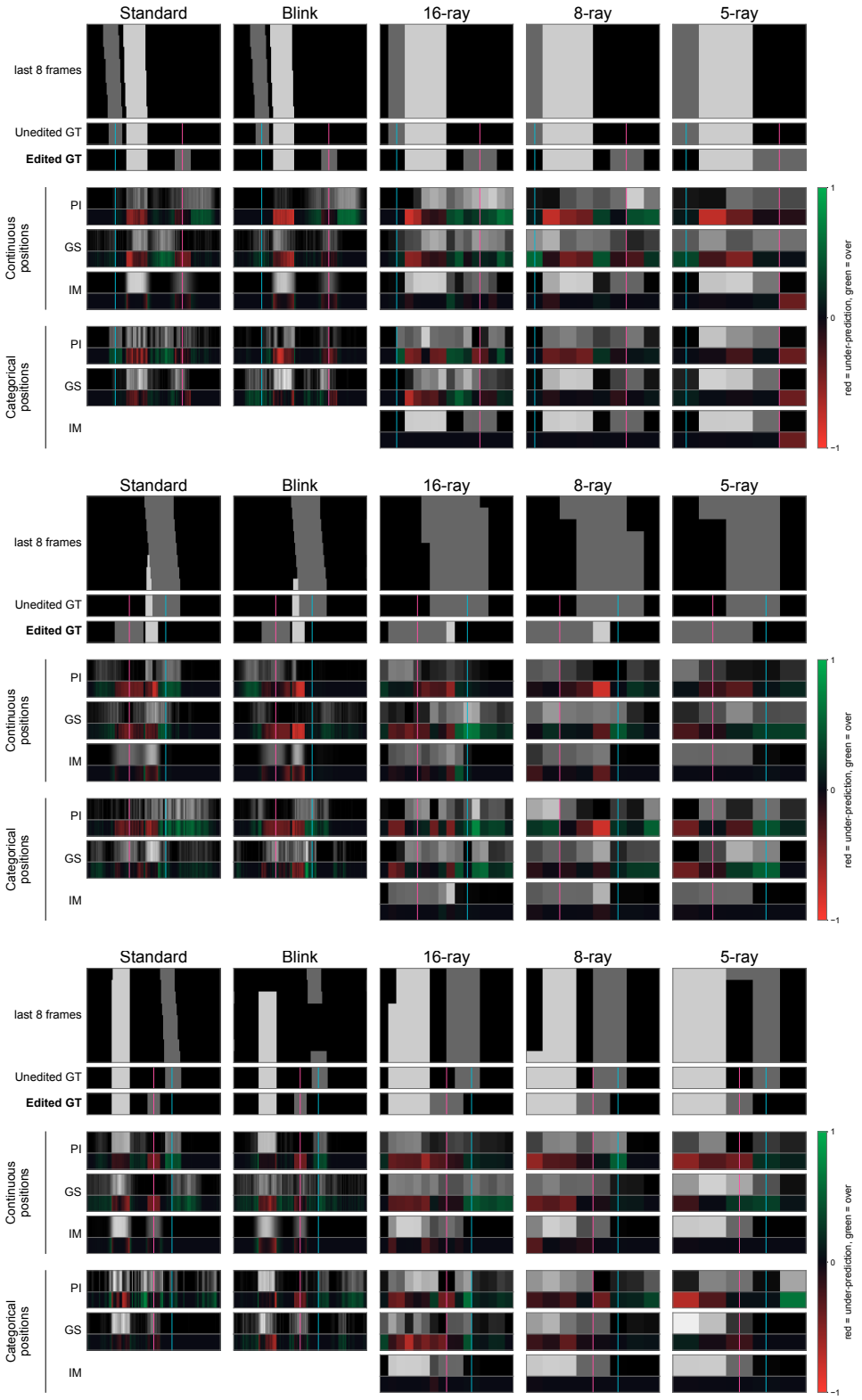

Figure 10: Additional qualitative visualizations of Rayworld edits.

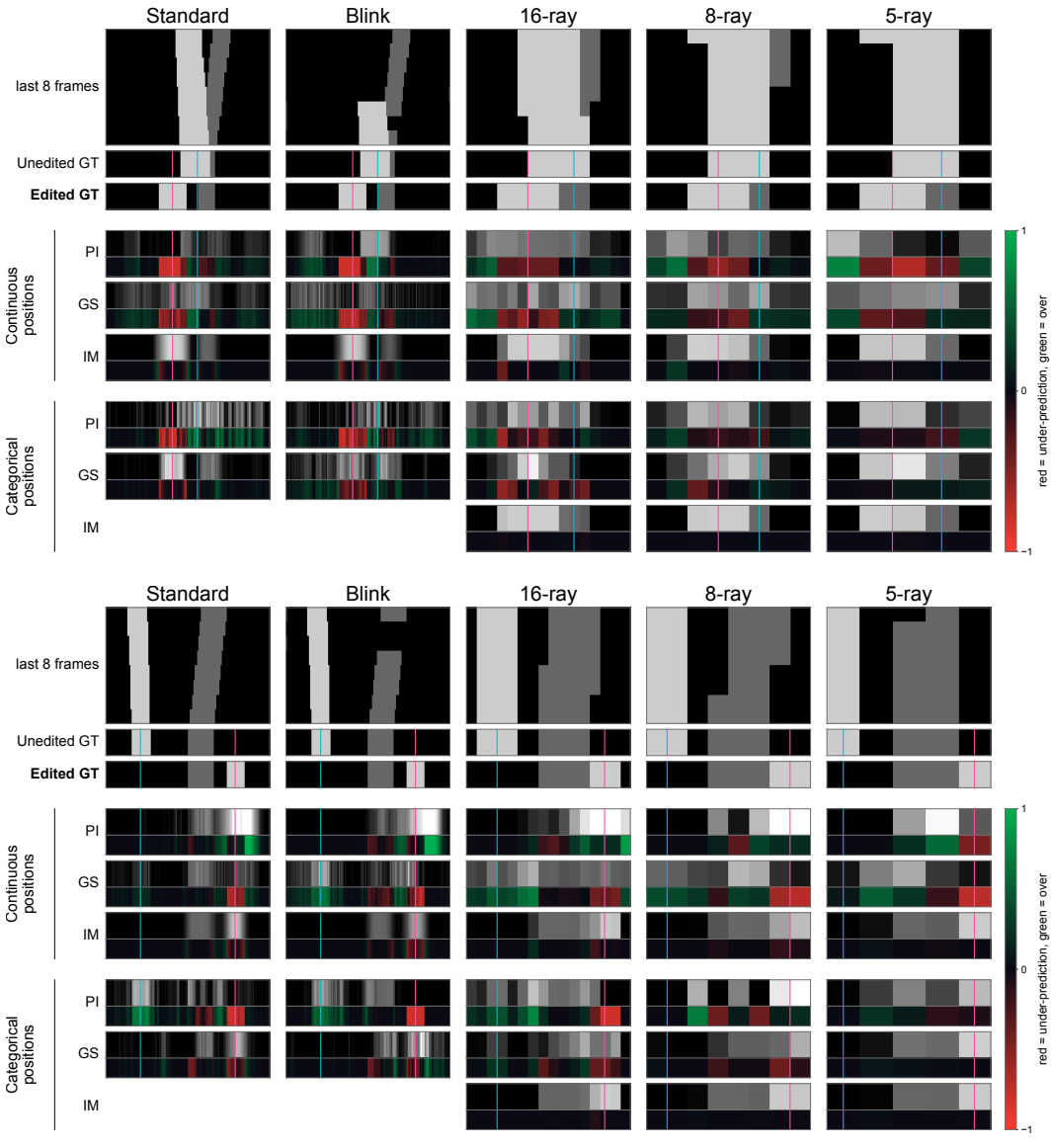

Figure 11: Additional qualitative visualizations of Rayworld edits.

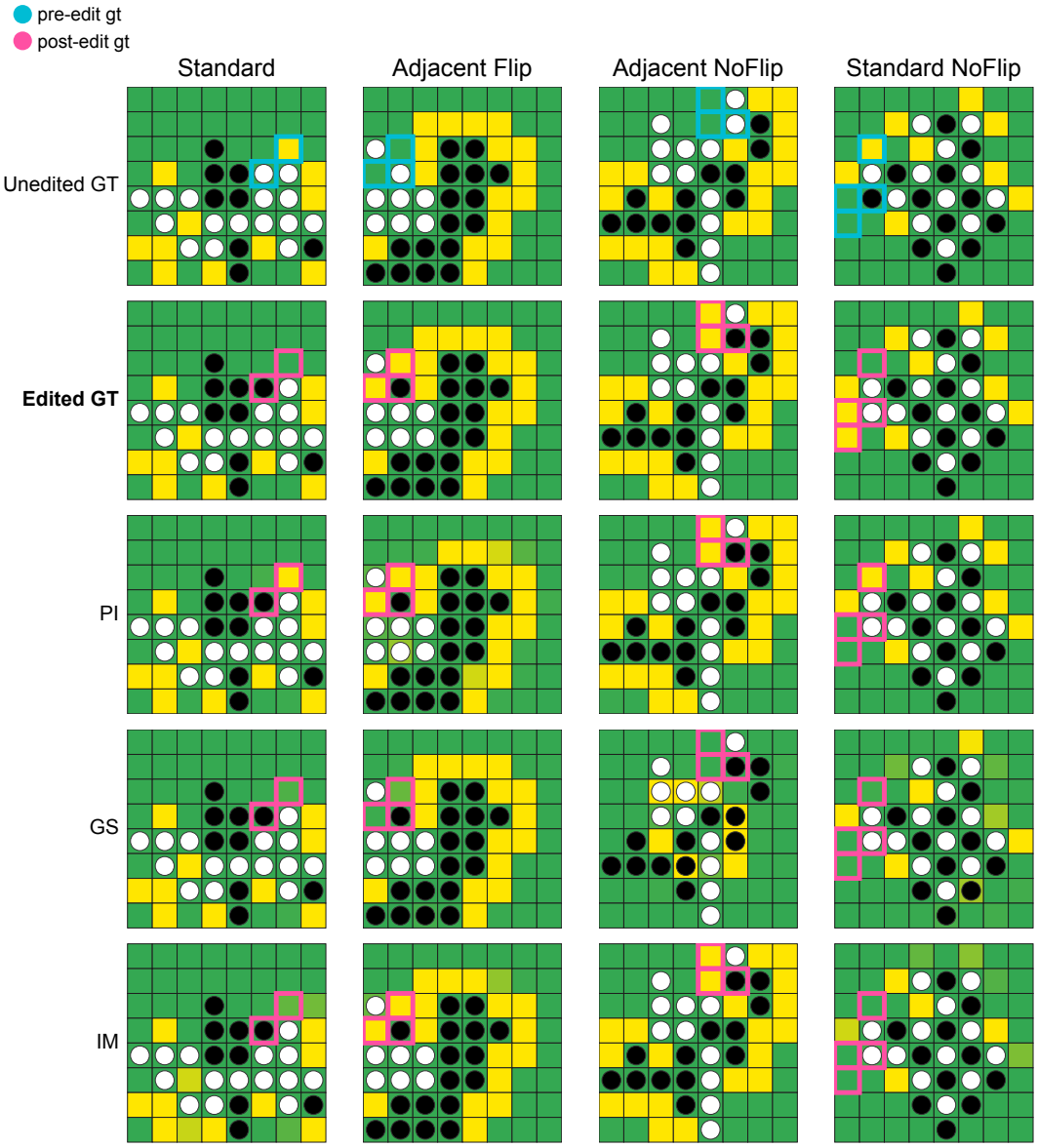

Figure 12: Additional qualitative visualization of Othello edits.

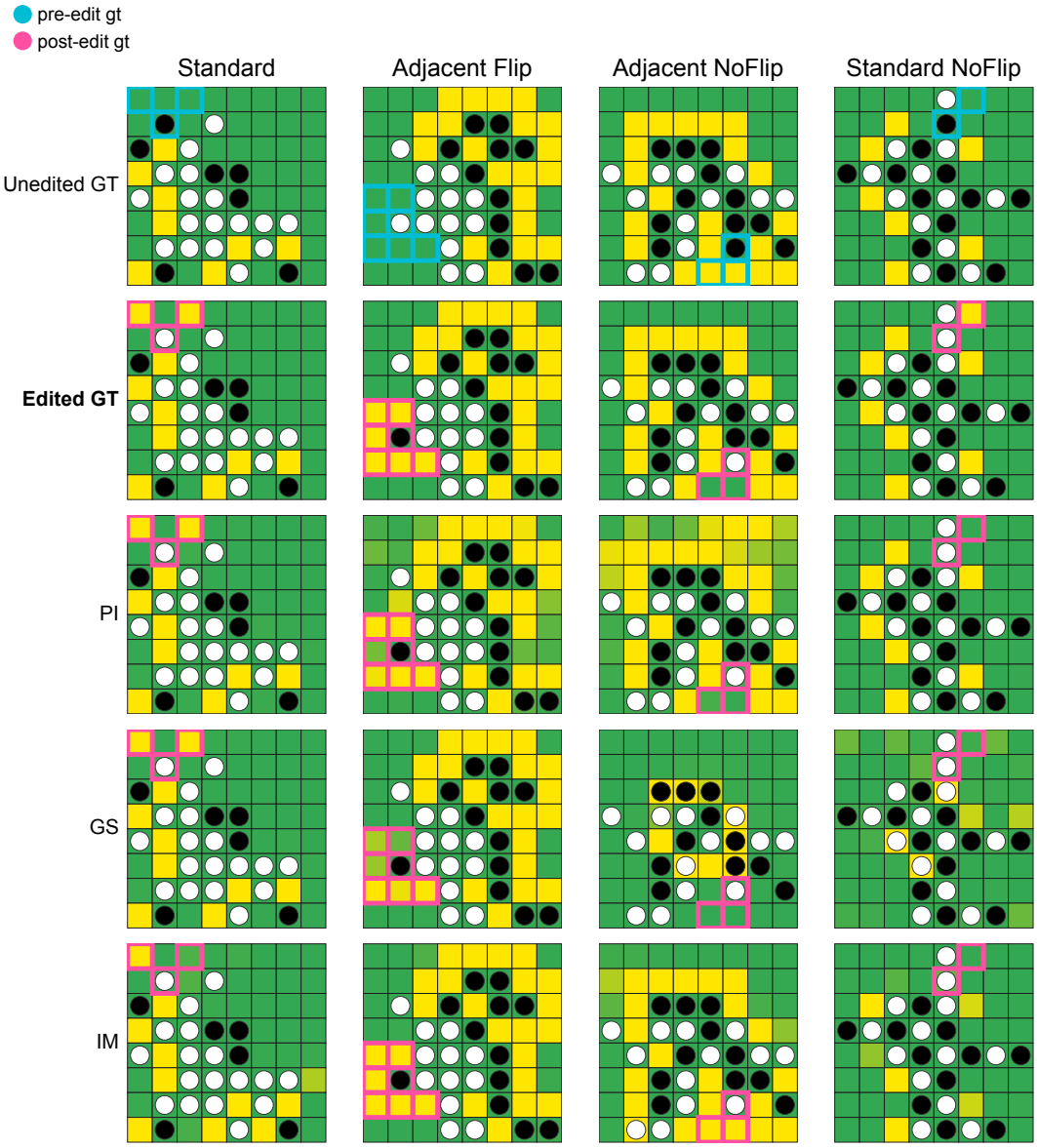

Figure 13: Additional qualitative visualization of Othello edits.

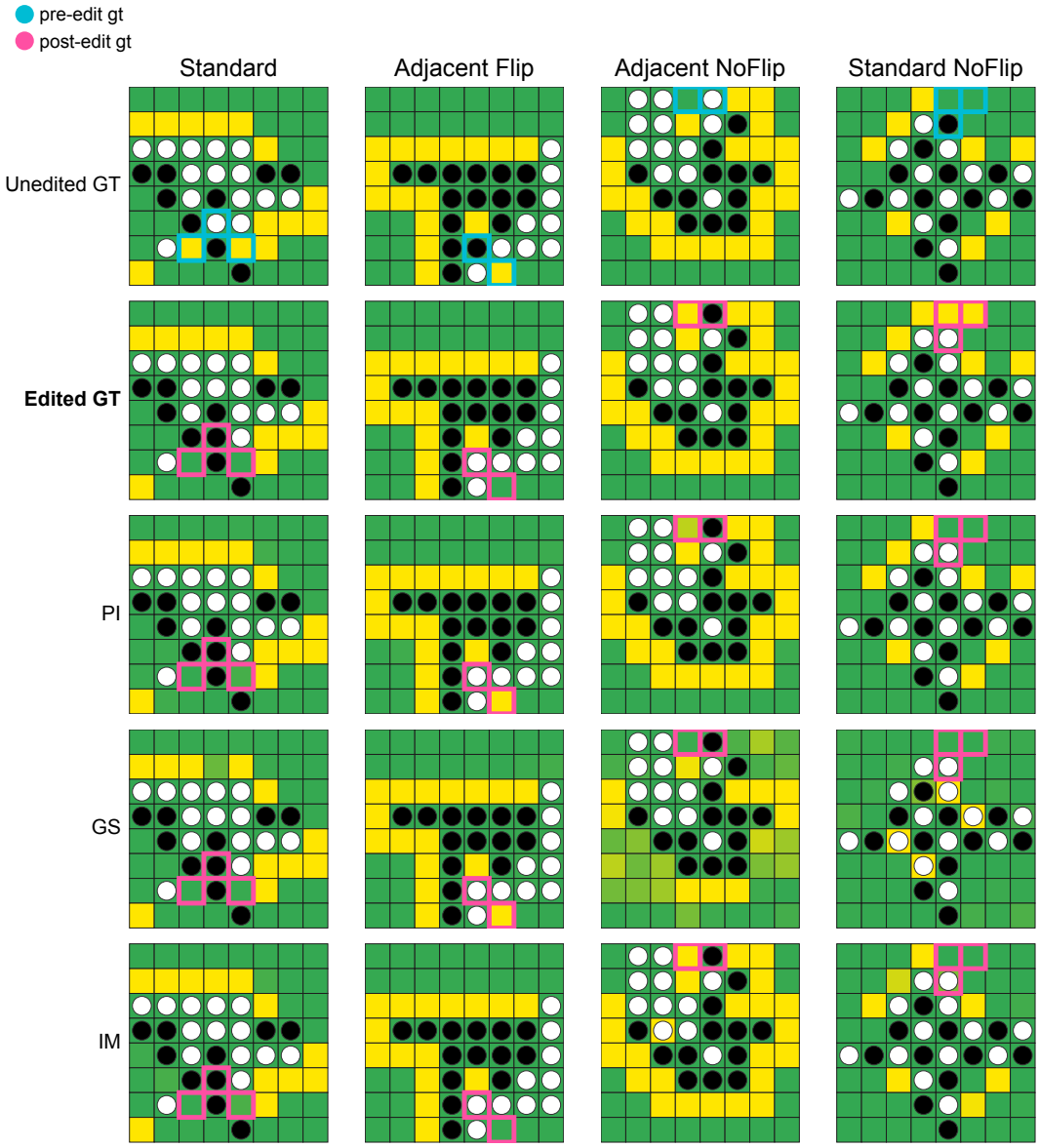

Figure 14: Additional qualitative visualization of Othello edits.

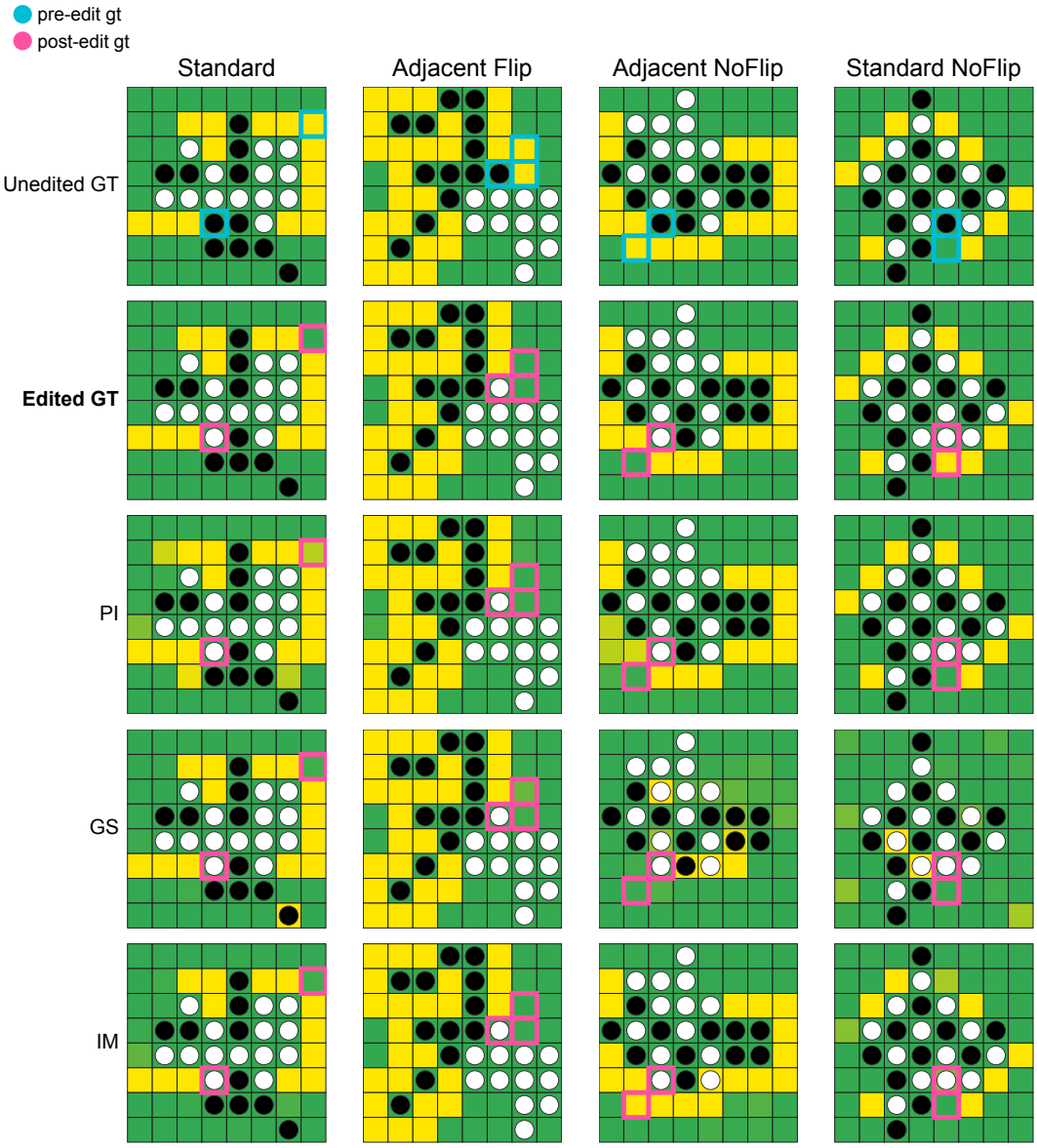

Figure 15: Additional qualitative visualization of Othello edits.

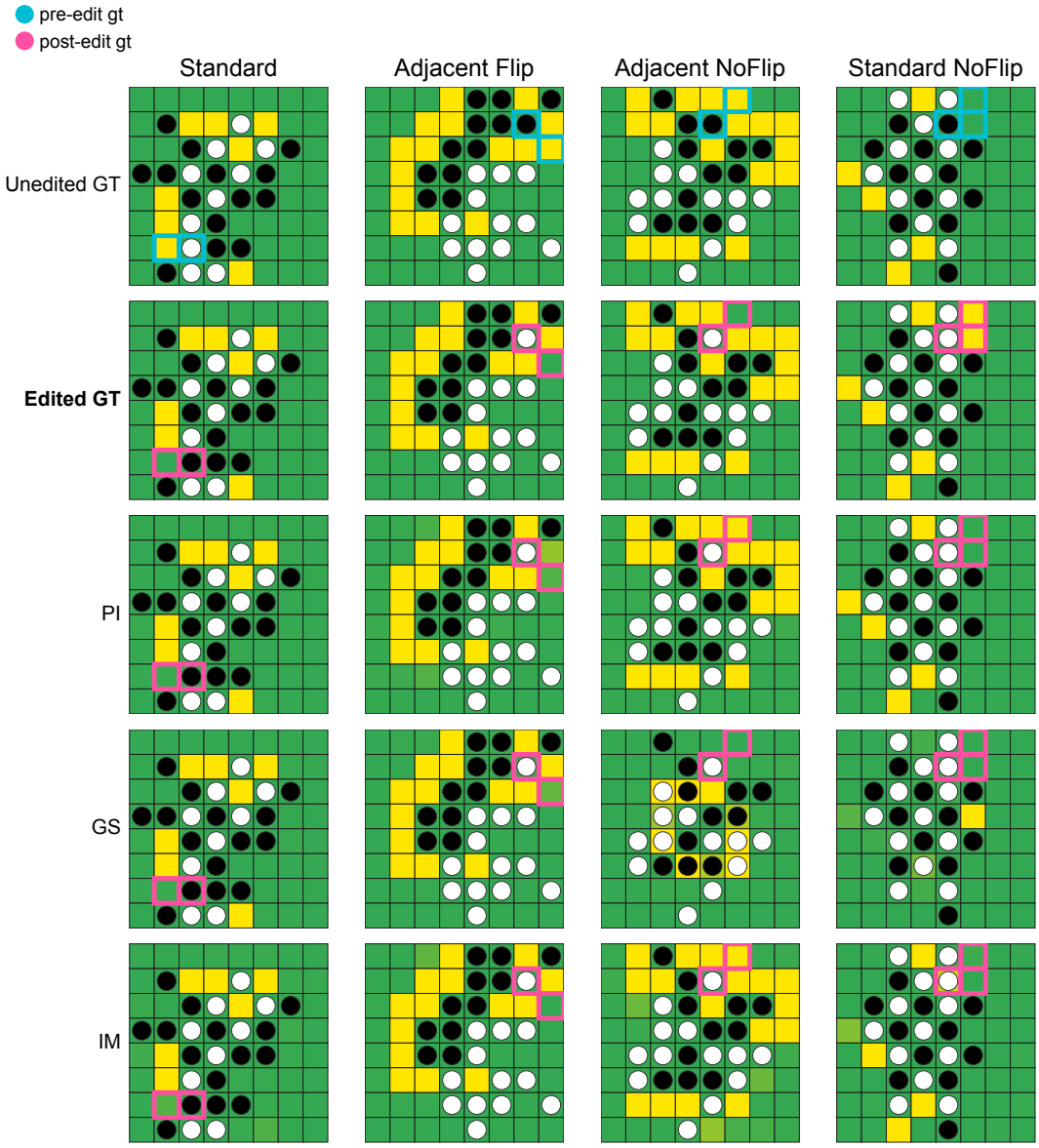

Figure 16: Additional qualitative visualization of Othello edits.