Title

Interpreting Othello-Playing Transformers via Sparse Dictionary Learning

ISBN

9798277436578

Author

Zhang, Zhiyuan

Publication Date

2026-03-10

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Interpreting Othello-Playing Transformers via Sparse Dictionary Learning

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Statistics

by

Zhiyuan Zhang

2026

ABSTRACT OF THE THESIS

Interpreting Othello-Playing Transformers
via Sparse Dictionary Learning

by

Zhiyuan Zhang
Master of Science in Statistics
University of California, Los Angeles, 2026
Professor Yingnian Wu, Chair

Mechanistic interpretability aims to reverse-engineer the internal computations of neural networks into human-readable algorithms. We study a decoder-only Transformer trained to predict legal moves in 6×6 Othello from move sequences alone, achieving 99.62% top-k legal-move accuracy. We train two sparse dictionary learning models using end-to-end cross-entropy training: JumpReLU transcoders (replacing MLP sublayers) and JumpReLU sparse autoencoders (reconstructing residual streams), preserving top-k accuracy at 99.3–99.6% across all layers.

To measure each feature's interpretability, we fit decision trees over hand-crafted boolean game features and evaluate the F_1 score. A clear interpretability trend emerges across depth: in transcoder Layer 0, 58.1% of features achieve $F_1 > 0.99$, whereas this fraction declines to 1.8% in Layer 5. The SAE exhibits a similar but smoother trend, from 35.8% in Layer 0 to 6.8% in Layer 5. We conduct causal intervention experiments to confirm that these sparse features drive the model's predictions rather than merely correlating with game state. Comparing random forests against single decision trees, we find that the primary

bottleneck for unexplained features is feature engineering (13–20% of features lack adequate boolean descriptors) rather than classifier capacity (>70% are fully explained by a single tree). These results demonstrate that sparse dictionary learning can extract large numbers of unambiguous, human-readable rules from a Transformer trained in a well-controlled domain.

The thesis of Zhiyuan Zhang is approved.

Xiaowu Dai

Guang Cheng

Yingnian Wu, Committee Chair

University of California, Los Angeles

2026

To my family and LLMs

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor, Professor Yingnian Wu, for his guidance and support throughout this work. I would like to thank my committee members Professor Guang Cheng and Professor Xiaowu Dai as well, for their valuable feedback. I am also grateful to Shu Wang and Danni Liu for helpful discussions during the course of this work.

VITA

2026 M.S. (Statistics), University of California, Los Angeles.

CHAPTER 1

Introduction

1.1 Motivation

Mechanistic interpretability aims to reverse-engineer neural network computations into human-understandable algorithms. Recent work on Circuit Tracing [ALP25, Ant25] has demonstrated the benefits of this path, as it enables our deeper understanding how large language model (LLM) work internally and enables practical applications of model steering, where we can control the behavior of LLMs by manipulating the interpreted features.

Sparse dictionary learning has emerged as a promising tool for decomposing neural networks into interpretable, sparse features. However, applying these methods to LLMs faces a foundamental issue: the computational cost makes it almost impossible to train a sparse surrogate to achieve the same cross-entropy loss as the original Transformer. When a sparse model fails to capture some behavior, it is impossible to determine whether the failure stems from the surrogate's architecture, insufficient training, or non-sparse representations within the original model. Resolving this ambiguity is our key motivation for this work, and a primary reason we turn to a small, controlled model.

Board games offer verifiable ground truth absent in natural language, thus are ideal for interpretability research [TWL22, MKT22]. Othello has a finite state space (yet large enough to be tackled with a Transformer), well-defined rules, and allows direct verification of game state and move predictions. Li et al. [LHB23] showed that a Transformer trained on move sequences develops emergent board-state representations recoverable by nonlinear

probes, and that intervening on these representations changes model predictions. Nanda et al. [NLW23] found that board state is linearly encoded as relative tile (mine/yours/empty) and performed causal interventions using these linear directions.

More recent work has provided more evidence for this picture. Yuan and Søgaard [YS25] and jylin04 [jyl24] found evidence that OthelloGPT relies primarily on local heuristics rather than a unified world model. Singh et al. [SWM25] applied decision trees to discover rule-based MLP neurons, while Du et al. [DHI25] used SAEs and found features such as tile stability. These studies suggest that Othello Transformers encode rich, structured information—but a comprehensive, causally validated analysis using modern sparse dictionary learning has yet to be carried out.

1.2 Research Question

This work aims to analyze the Transformer's inner computations using sparse dictionary learning. To be specific, we train JumpReLU transcoders to replicate the MLPs and JumpReLU SAEs to reconstruct the residual stream.

Our Transformer is trained on randomly sampled games, so it only learns the rules of Othello and has no concept of strategies. This choice removed strategy information from the dataset, which made interpretability easier and more focused.

We have two questions that guides our investigation. First, do the learned sparse features correspond to human-understandale game rules? Second, can we understand how the Transformer uses input tokens to reconstruct boards, and predict next moves?

1.3 Contributions

This thesis makes four contributions.

Extracting unambiguous rules at scale. We decompose both MLP computations (via transcoders) and residual-stream representations (via SAEs), yielding large numbers of features that are almost perfectly explained by rules ($F_1 > 0.99$). Because reconstruction is near-lossless (top-k accuracy almost unchanged), we can rule out insufficient training and attribute any remaining uninterpretable features to the representations themselves.

Causal validation of sparse features. We conduct six causal intervention experiments, four on transcoders and two on SAEs, spanning move identity, board state, sequence history, and compound features, to establish that the discovered sparse features are causally used by the model. While Nanda et al. [NLW23] performed causal interventions using linear probe directions, our interventions target individual sparse dictionary features, providing finer-grained causal evidence.

Interpretability trend across depth. The proportion of features with $F_1 > 0.99$ declines monotonically from 58.1% (Layer 0) to 1.8% (Layer 5) for transcoders, and from 35.8% to 6.8% for SAEs, which implies that early layers encode simple features while deeper layers form complex representations.

Feature engineering as the bottleneck. Comparing random forests against single decision trees across, we find that a single tree already works well for over 70% of neurons. The true bottleneck is not classifier capacity but feature engineering: 13–20% of neurons are "Feature Insufficient", which means that these concepts are not well captured by our hand-crafted boolean features.

1.4 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 reviews related work. Chapter 3 describes the Transformer architecture, the JumpReLU transcoder and SAE, and the

decision tree framework. Chapter 4 presents experiment settings and results. Chapter 5 discusses implications and limitations. Chapter 6 summarizes our findings and outlines future work.

CHAPTER 2

Related Work

2.1 Mechanistic Interpretability and Othello

Othello is a good interpretability testbed as it combines a deterministic game environment with a large enough complexity. During the game, two players place pieces on an $n \times n$ board; opponent pieces flanked in a straight line are flipped to the current player's color. Our variant on a smaller board (6×6) preserves the core mechanics, such as corner control and flips, while remaining computationally manageable for our thorough analysis.

Li et al. [LHB23] demonstrated that a transformer trained on Othello move sequences can develop an emergent internal representation, or, a world model, of the board state, which can be probed and causally intervened using MLP probes. Their work established OthelloGPT as a good interpretability setting where ground truth board states can be used to provide an objective metric.

Nanda et al. [NLW23] advanced this by showing that board state is linearly encoded as relative tile ownership (mine/yours/empty) rather than absolute color, and performed causal interventions by modifying these linear directions in the residual stream to verify that the model's predictions change accordingly.

More recent work has revisited the emergent world model hypothesis. Yuan and Søgaard [YS25] argued that OthelloGPT relies primarily on local heuristics rather than a coherent global model, and jylin04 [jyl24] characterized its computation as "a bag of heuristics". These findings suggest that the model might not have a clean world model where board state is

strictly tracked, which is directly relevant to our findings that interpretability drops in later layers.

Singh et al. [SWM25] trained decision trees on raw MLP neuron activations, finding that nearly 50% of neurons in layer 5 can be described by a depth-4 tree with $R^2 > 0.7$. Without sparse decomposition, however, superposition limits their results to $R^2 \approx 0.8$. By first sparsifying the superpositioned features, our method reaches $F_1 > 0.99$, suggesting that this ceiling is largely an artifact of polysemantic neurons. Du et al. [DHI25] applied SAEs to OthelloGPT and discovered a few interpretable features, such as chess stability.

Researchers had similar findings in other chess games as well. Toshniwal et al. [TWL22] showed that Transformers trained on move sequences learn to track chess board state, and McGrath et al. [MKT22] found that AlphaZero has human-understandable concepts as emergent representations.

2.2 Sparse Dictionary Learning

Superposition is a phenomenon that has blocked researchers from interpreting neurons [EHO22]. Because neural networks can encode more features than their dimensions, individual neurons carry polysemantic meanings and are not directly interpretable. Sparse dictionary learning methods try to address this by using an over-complete dictionary. Ideally, each dictionary item should correspond to a single interpretable feature.

Sparse Autoencoders

Bricken et al. [BTB23] and Templeton et al. [TCM24] trained SAEs on modern LLMs (Claude 3 series) and found thousands of interpretable features, such as code bugs and the Golden Gate Bridge. Cunningham et al. [CER24] also found that SAE features could be interpreted as to monosemantic concepts.

Transcoders

Dunefsky et al. [DCN24] introduced transcoders to replace MLP sublayers. Transcoders decomposes the MLP's computation using a sparse dictionary so that each dictionary feature represents something to be read and written into the residual stream, instead of extracintg the information that is present in the residual stream. They further showed that transcoder features can be composed into integreated into circuits tracing methods for multi-token, multi-layer tracking.

Circuit Tracing

Ameisen et al. [ALP25] introduced attribution graphs that trace information flow between sparse features across layers, subsequently extended to attention layers [Ant25]. However, this requires that every neuron in the circuit must be interpretable. If neurons lack clear meaning, the circuit graph becomes uninterpretable. As we report in Chapter 4, interpretable features drop sharply with depth, suggesting that complete circuit tracing on our model remains an open challenge.

Activation Functions: JumpReLU versus TopK

Rajamanoharan et al. [RCS24] proposed the JumpReLU SAE, where each dictionary item has a learnable threshold θ, enabling feature-level sparsity regularization. Gao et al. [GTT24] proposed TopK SAEs, which allow exactly k biggest activations to pass through.

In our early experiments, TopK's fixed activation count harms interpretability severly, as simple early-game positions and easy late-game moves activate k features regardless of how little information they carry, introducing noise. But JumpReLU avoids this by using an adaptive threshold for different neurons, and the number of active features can vary largely on different inputs. We use it for the rest of this work.

CHAPTER 3

Methods

3.1 Transformer Architecture

We use a standard, modern, decoder-only Transformer and train it on Othello move sequences. We used:

- **RoPE** [SAL24]: positional encoding via rotary queries and keys.
- **Elementwise Gated Attention** [QWZ25]: a sigmoid gate that modulates the attention output element-wise.
- **Pre-norm**: LayerNorm before each sub-layer.
- **GELU activation** in the MLP.

We tried a few architectures, and this combination proved to converge the fastest. We provide detailed hyperparameter values Chapter 4.

3.1.1 Tokenization

Board position indices are flattened and directly mapped to integers $\{0, \ldots, n^2 - 1\}$. The pass move is tokenized as n^2. With $\textsc{bos} = n^2 + 1$ and $\textsc{eos} = n^2 + 2.$, each game is tokenized as $[\textsc{bos},\ m_1,\ \ldots,\ m_T,\ \textsc{eos}]$.

3.1.2 Training Objective

The Transformer is trained with standard next-token cross-entropy loss.

Because multiple moves are legal at each position, we evaluate using *top-k accuracy*: for each position with k legal moves, we take the model's top-k predictions and measure what fraction of them are indeed legal.

3.2 JumpReLU Transcoder

3.2.1 Architecture

For each layer, we train a JumpReLU transcoder to replace the MLP, mapping MLP input $\mathbf{x} \in \mathbb{R}^d$ to an approximation $\hat{\mathbf{y}} \in \mathbb{R}^d$ of the MLP output using a sparse dictionary.

The computation proceeds in three stages:

$$z_i = (\mathbf{W}_{\text{enc}}\mathbf{x} + \mathbf{b}_{\text{enc}})_i, \tag{3.1}$$

$$h_i = z_i \cdot \mathbf{1}[z_i > \theta_i], \tag{3.2}$$

$$\hat{\mathbf{y}} = \mathbf{W}_{\text{dec}}\mathbf{h} + \mathbf{b}_{\text{dec}}, \tag{3.3}$$

where $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{D\times d}$ is the encoder weight matrix, $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{d\times D}$ is the decoder weight matrix, and $\boldsymbol{\theta} \in \mathbb{R}^D$ is a vector of learnable per-feature thresholds. The activation function in Equation (3.2) is the JumpReLU [RCS24]: it passes the pre-activation z_i through only when it exceeds the threshold θ_i, and zeros it out otherwise, producing a sparse activation vector $\mathbf{h}$. The straight-through estimator is used to backpropagate the gradient through the JumpReLU [RCS24]. The dictionary dimension is $D = 8192$, which is $64\times$ the model's embedding dimension $d = 128$.

Following previous work, the columns of $\mathbf{W}_{\text{dec}}$ are constrained to unit norm [BTB23, RCS24], so each dictionary feature defines a direction in activation space and its magnitude is determined by the corresponding coefficient h_i. We use log-parameterized thresholds

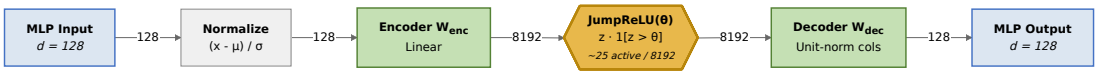

Figure 3.1: Architecture of the JumpReLU transcoder. The MLP input $\mathbf{x}$ goes through a linear encoder, then goes through a JumpReLU activation with learnable per-feature thresholds θ_i, finally the activations $\mathbf{h}$ are projected back to MLP output space by a unit-norm linear decoder.

($\theta_i = \exp(\log \theta_i)$) to ensure positivity.

To tackle the non-differentiable L_0, we approximate the gradient following Rajamanoharan et al. [RCS24]:

$$\frac{\partial L_0}{\partial \theta_i} \approx -\frac{1}{n\epsilon} \sum_{j=1}^{n} K\left(\frac{z_i^{(j)} - \theta_i}{\epsilon}\right), \tag{3.4}$$

where K is a rectangular kernel and ϵ is the bandwidth. The bandwidths for L_0 loss and JumpReLU are set to $\epsilon = 0.001$ [RCS24].

3.2.2 Normalization

Inputs are standardized using $\tilde{\mathbf{x}} = \mathbf{x}/\sigma$, where σ is the root mean square (RMS) norm computed from a fixed number of batches [BTB23, RCS24].

3.2.3 Loss Function and Training

The transcoder is trained end-to-end with the frozen Transformer using:

$$\mathcal{L} = \mathcal{L}_{\text{CE}}(p_{\text{model}}, y_{\text{next}}) + \lambda \cdot L_0, \tag{3.5}$$

with no separate MSE pre-training phase (unlike large models which typically train with reconstruction loss alone or with CE end-to-end fine-tuning [BTB23, TCM24]). Only transcoder parameters are optimized.

We fix $\lambda = 1.0$ for all layers. Because the thresholds $\boldsymbol{\theta}$ receive gradients only from L_0,

Adam's first order momentom normalizes each parameter's gradient by its own gradient norm and cancelled out λ. As a result, tuning λ has almost no effect, and each layer's activation counts converge to their natural levels.

A critical detail is setting $\beta_1 = 0$ in AdamW for *all* parameter groups. Rajamanoharan et al. [RCS24] noted small benefits of $\beta_1 = 0$ for thresholds alone, but in our setting it proved essential across the board: with $\beta_1 > 0$, momentum causes thresholds to overshoot, killing too many neurons and dropping end-to-end accuracy by roughly one percentage point (from 99.5% to ∼98.4% in average). Mixed configurations (momentum for encoder/decoder but not $\boldsymbol{\theta}$) also underperformed and a systematic exploration is left for future work.

3.2.4 Sparsity Control and Early Stopping

We define the dead neuron ratio as the fraction of dictionary elements that never activate. Although dead neurons represent unused capacity, a high dead ratio is actually desirable here, as it indicates that the dictionary has converged to a compact set of non-redundant features.

Training is stopped when both the activation count (L_0) and the dead ratio plateau.

3.3 Sparse Autoencoder on Residual Stream

The SAE reuses the identical JumpReLU architecture (Section 3.2) with input_dim = output_dim = $d = 128$, placed at the output of each layer after the residual connection.

Training is identical to the transcoder: frozen Transformer, per-layer, end-to-end cross-entropy with L_0 penalty.

End-to-end training is feasible because our model is small; on large models, SAEs are typically trained with reconstruction loss alone [BTB23, TCM24].

The key distinction between the transcoder and SAE is that the former decomposes MLP

computations (input $\rightarrow$ output), while the latter decomposes the residual stream *representations*. Sharing the same architecture and training framework makes their results directly comparable.

3.4 Decision Tree Interpretability Analysis

To understand what each sparse feature computes, we fit shallow decision trees that predict feature activations from board-state conditions. This approach is inspired by Singh et al. [SWM25], who applied decision trees to raw MLP neurons but were limited to $R^2 \approx 0.8$ due to superposition. By analyzing sparse dictionary features instead, we exploit the disentanglement provided by sparsity, which pushes F_1 scores above 0.99.

3.4.1 Feature Engineering

All features are binary descriptors of the board state (occupancy, flips, move position), deliberately excluding strategic or evaluative metrics since our model is trained on random play.

The feature set consists of two categories: **Board-state features** capture the raw game state:

- *Absolute board occupancy*: binary features for black and white pieces on each of the n^2 squares ($2n^2$ features).
- *Relative board occupancy*: binary features for the current player's pieces and the opponent's pieces on each square ($2n^2$ features).
- *Flipped positions*: binary features for the squares that were flipped by the current move.
- *Move one-hot*: a one-hot encoding of the current move's board position.

- *History and sequence features*: features encoding the game's move history.
- *Previous move one-hot*: one-hot encodings of the k most recent moves, with k ranging from 1 to 6.

Derived features include complex, high-level properties, such as:

- *Flip directions*: which of the 8 directions were involved in flipping.
- *Move row and column*: the row and column index of the current move.
- *Neighbor of move*: which positions next to the current move are occupied.
- *Direction lines*: occupancy patterns along lines connecting the move.
- *Game phase*: whether it's early, middle, or late game.
- *Board and color aggregates*: summary statistics such as the total number of pieces on the board.
- *Legal move aggregates*: counts and spatial distribution of currently legal moves.
- *Corner state and region aggregates*: occupancy information for strategically important board regions (corners, edges).

In total, the feature vector for a 6×6 board with `prev_move_steps` $= 6$ comprises $\sim$700 boolean features.

3.4.2 Data Collection and Tree Training

To collect data for decision tree training, we first run the full dataset through the model and get the sparse activations. Then, we first sample balanced positive and negative examples, then replay the games to reconstruct the board states and compute derived features.

For each feature, we train a `DecisionTreeClassifier` (max depth 10) with a depth sweep: we select the shallowest tree achieving validation $F_1 \geq 0.99$, or the best F_1 if none reaches this threshold. In practice, most features with $F_1 > 0.99$ have a depth ≤ 4.

3.4.3 Interpretability Scoring

We classify features by validation F_1: **highly interpretable** (> 0.99), **interpretable** (0.95–0.99), **partially interpretable** (0.90–0.95), and **not interpretable** (≤ 0.90).

We evaluate interpretability quality using an activation-frequency-weighted mean F_1, so that frequently active features contribute more to the score.

3.5 Causal Intervention

High F_1 scores tells that features correlate well with game state conditions, but it does not mean that the model uses them for prediction. To test this, we directly modify feature activations during inference and check whether the model's outputs change as expected.

Our approach is similar to Nanda et al. [NLW23]'s linear-probe interventions by replacing probe directions with sparse dictionary features. Because dictionary features correspond to individual rules, they can provide finer-grained intervention targets, but this specificity also means that each intervention must be specifically examined and validated to a particular feature on a specific token in a specific sequence. We therefore present representative case studies rather than aggregate statistics.

For interventions (both transcoder and SAE), we modify some items of $\mathbf{h}$ (zeroing, swapping, or setting values), and the changes propagate through subsequent layers and subsequent tokens.

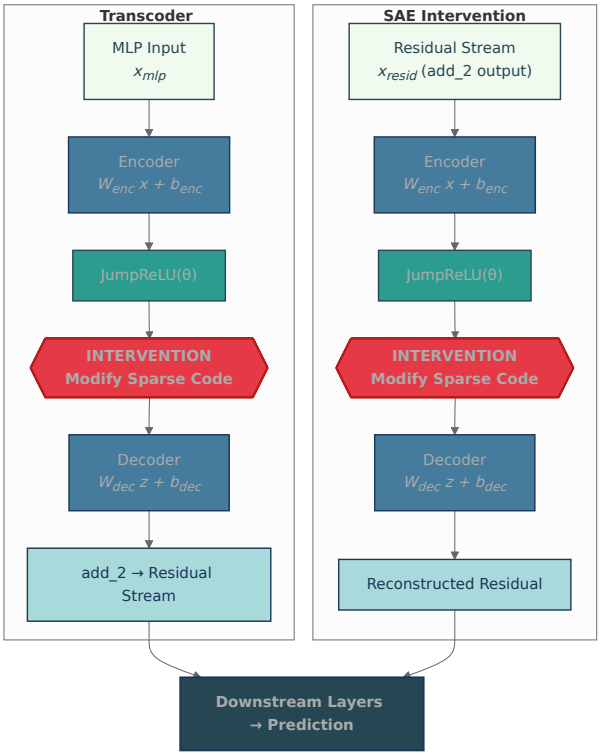

Figure 3.2: Causal intervention methodology for the transcoder and the SAE. For the transcoder (left), specific items of the activation vector $\mathbf{h}$ are modified (zeroed, swapped, or set to a target value) before the decoder, and replaces the MLP so that the changes propagates through subsequent layers. For the SAE (right), intervention works similarly but SAEs are injected into the residual stream.

CHAPTER 4

Experiments and Results

In this chapter, we present the complete experiments together with results, describing each experiment setup alongside its outcomes. All training was conducted on single NVIDIA A6000 GPUs.

4.1 Data Generation

All experiments use a 6×6 Othello board with games generated by uniform random play. We generated a total of approximately 5.7 million games averaging 32 moves each ($\sim$184M tokens). The dataset is split 90/10 for training/validation.

4.2 Transformer Training and Performance

4.2.1 Configuration

We summarize the OthelloGPT hyperparameters in Table 4.1: $d = 128$, 6 layers, 8 attention heads, $4\times$ MLP expansion with GELU, Elementwise Gated Attention [QWZ25], and RoPE [SAL24].

Our training uses AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 0.01) with cosine-annealed learning rate ($10^{-3} \rightarrow 10^{-6}$), gradient clipping at 0.5, bfloat16 mixed precision, batch size 3,072, and early stopping (patience 5, monitoring top-k accuracy).

Table 4.1: Architecture and training hyperparameters for the OthelloGPT model. The model uses a 128 embedding dimension with elementwise gated attention and RoPE positional encoding, trained with cosine-annealed AdamW and early stopping on top-k validation accuracy.

Hyperparameter	Value
Embedding dimension (d)	128
Layers	6
Attention heads	8 ($d_{\text{head}} = 16$)
MLP expansion	4× (hidden = 512)
MLP activation	GELU
Attention variant	Elementwise Gated
Positional encoding	RoPE (base = 1000)
Batch size	3,072
Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$)
Learning rate	10^{-3} (cosine decay to 10^{-6})
Gradient clipping	0.5
Precision	bfloat16 mixed
Early stopping	patience = 5, monitor = `val_acc_topk`

4.2.2 Results

The trained OthelloGPT achieves 99.62% top-k validation accuracy, confirming near-perfect rule learning from move sequences alone. Figure 4.1 shows the training curve.

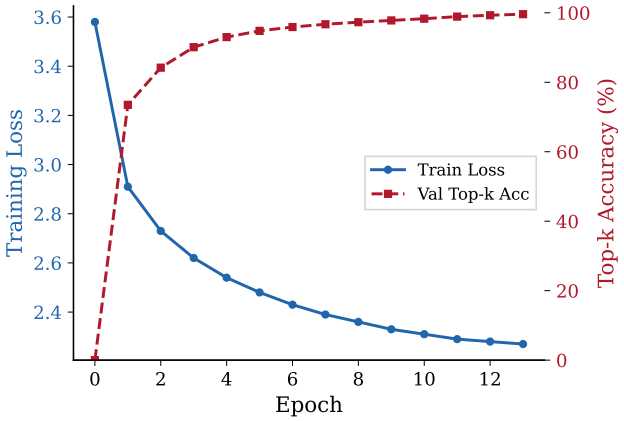

Figure 4.1: Training curve of the OthelloGPT model, showing top-k validation accuracy over training epochs. The model exhibits rapid initial learning followed by gradual refinement, converging to 99.62% accuracy before early stopping triggers.

4.3 Linear Probing: Board State Representation

Linear probing confirms that our 6×6 OthelloGPT develops board-state representations in its residual stream, which is consistent with prior result on a 8×8 board [LHB23, NLW23]. We probed residual stream activations (dim 128) from 5,000 games ($\sim$160,000 tokens) using both linear and MLP probes using relative (mine/opponent/empty) or absolute (black/white/empty) board states.

Figure 4.2 shows the results. Under relative encoding, the linear probe reaches 98.0% at Layer 4, increasing monotonically from 79.0% (Layer 0) with a slight decline at Layer 5 (96.8%). Relative encoding dramatically outperforms absolute (98.0% vs. 75.5%), confirming that the model adopts a current-player perspective [NLW23]. The MLP probe adds at most

0.5 percentage points (pp), confirming that the encoding is approximately linear. These results validate our model and provide a correlational baseline for the causal experiments in Section 4.7.

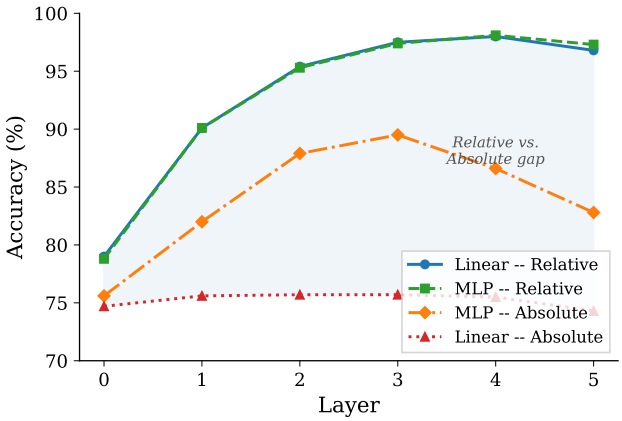

Figure 4.2: Probe accuracies for Transformer layers for both encoding schemes and probe types. Accuracies for relative board states increases from Layer 0 through Layer 4 before decreasing slightly at Layer 5, suggesting that the final layer begins to specialize for output prediction rather than continue to maintain a clean relative board-state representation.

4.4 Transcoder Training and Sparsity

4.4.1 Configuration

Transcoders and SAEs share identical hyperparameters, differing only in insertion point (MLP replacement vs. residual stream interception at `add_2`). Per-layer training uses frozen Transformer weights and end-to-end cross-entropy loss.

Key hyperparameters: $D = 8{,}192$ (64× expansion), AdamW with learning rate 0.1, $\beta_1 = 0$, $\beta_2 = 0.999$, $\lambda = 1.0$, batch size 2,048, gradient clipping 1.0, unit-norm decoder columns. The $\beta_1 = 0$ setting is critical (see Chapter 3): with momentum, thresholds overshoot and accuracy degrades from 99.5% to 98.4%.

4.4.2 Sparsity and Reconstruction Results

All twelve sparse models preserve top-k accuracy within 0.33 percentage points (pp) of the original 99.62%, with the best within 0.06pp. Because this reconstruction is near-perfect—unlike the substantial cross-entropy gaps typical of large-model SAEs—any remaining uninterpretable features reflect genuine representation properties rather than training artifacts.

Living neurons range from 543 (Layer 0 TC, 6.6%) to 5,156 (Layer 5 SAE, 62.9%), indicating that early layers need far fewer dictionary elements.

The L_0 values reveal a depth pattern. For transcoders, L_0 rises steadily from 13.6 (Layer 0) to 29.9 (Layer 4), then jumps to 227.9 at Layer 5; SAEs show a similar trend (22.9–45.4 for Layers 0–4, then 256.1 at Layer 5), reflecting that the last layer handles dense output representations.

Within each layer, JumpReLU's adaptive sparsity is also evident: simple positions activate ∼15 neurons while complex mid-game positions activate ∼40. Figures 4.3 and 4.4 show the training dynamics and per-layer comparisons.

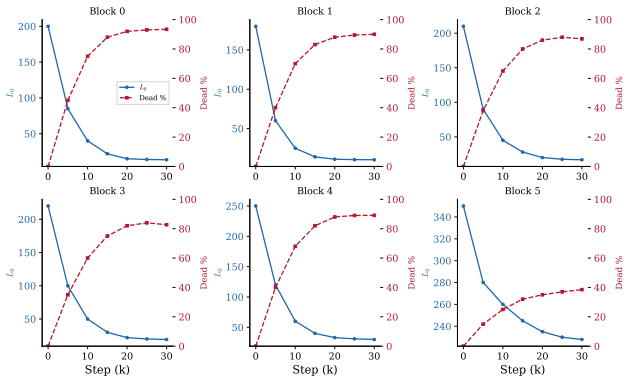

Figure 4.3: Transcoder training curves across all six layers, showing the decrease of L_0 (mean active neurons per token) and the increase of the dead neuron ratio over training steps. Layer 5 shows a sharp increase of L_0, demonstrating the transition from sparse, compositional features to dense representation for predictions.

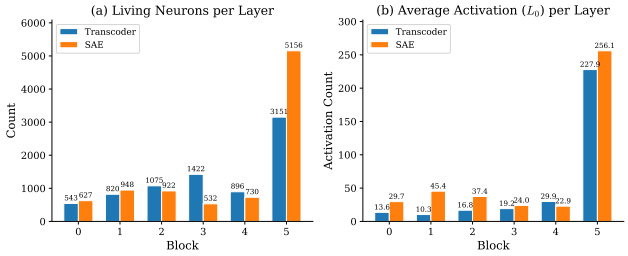

Figure 4.4: Per-layer comparison of living neuron counts and average activation magnitudes for transcoders and SAEs. Both models show that the number of living neurons and the activation density increases largely at Layer 5 than at earlier layers, consistent with the distributed representation analysis for the final layer.

4.5 Decision Tree Interpretability Analysis

For each living neuron with sufficient activation samples (at least 3 positive and 3 negative examples), we train a decision tree to predict its activation from hand-crafted boolean features (see Chapter 3 for details). Neurons with fewer examples are excluded, so the Analyzed counts in Tables 4.2/4.3 may be smaller than the total living neuron counts. The F_1 score on a measures how well the rules of the decision tree can explain each neuron's activation pattern.

4.5.1 Results: Per-Layer Interpretability

Tables 4.2/4.3 and Figures 4.5/4.6 reveal a clear interpretability trend. For transcoders, the fraction with $F_1 > 0.99$ drops monotonically from 58.1% (Layer 0) to 1.8% (Layer 5); for SAEs, from 35.8% to 6.8%. The Unique Rules column shows minimal redundancy at Layer 0 (243 rules for 258 neurons), and deeper layers show a high rule diversity while requiring complex, deep trees.

Table 4.2: Decision tree interpretability statistics for transcoder neurons by layer. For each layer, we report the number of analyzed neurons, counts and percentages exceeding various F_1 thresholds, and the number of unique decision tree rules. The proportion of highly interpretable neurons ($F_1 > 0.99$) decreases monotonically from Layer 0 (58.1%) to Layer 5 (1.8%).

Layer	Analyzed	F1>0.99	F1>0.95	F1>0.90	F1>0.99 (%)	Unique Rules
0	258	150	217	219	58.1	243
1	444	74	286	336	16.7	438
2	647	67	399	437	10.4	617
3	896	44	375	436	4.9	845
4	542	22	193	312	4.1	515
5	2,938	53	710	2,079	1.8	2,907

Table 4.3: Decision tree interpretability statistics for SAE neurons by layer. The same metrics as Table 4.2 are reported. SAE neurons follow the same interpretability trend as transcoder neurons, with the $F_1 > 0.99$ proportion declining from 35.8% at Layer 0 to 6.8% at Layer 5.

Layer	Analyzed	F1>0.99	F1>0.95	F1>0.90	F1>0.99 (%)	Unique Rules
0	617	221	308	310	35.8	312
1	714	201	416	436	28.2	517
2	663	137	355	423	20.7	574
3	351	54	179	223	15.4	310
4	460	47	208	272	10.2	361
5	4,108	280	1,237	1,777	6.8	3,452

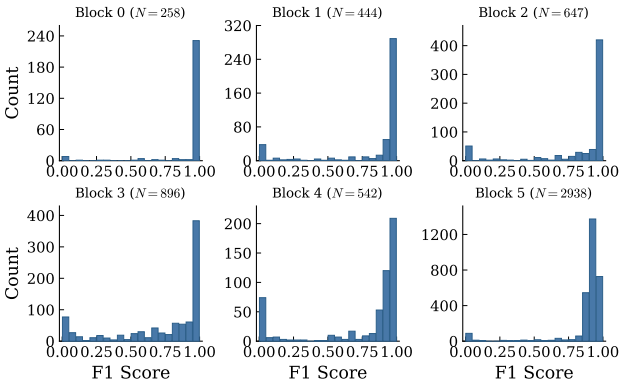

Figure 4.5: Histograms of decision tree F_1 scores for transcoder neurons at each layer. Early layers are have many highly interpretable neurons with high F_1 scores, while deeper layers show lower scores, demonstrating the transition from interpretable features to complex, distributed representations.

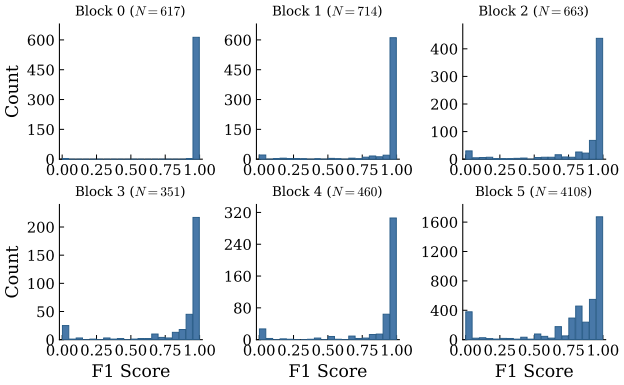

Figure 4.6: Histograms of decision tree F_1 scores for SAE neurons at each layer. The SAE exhibits a qualitatively similar interpretability trend to the transcoder, with a lower starting proportion at Layer 0 (35.8% vs. 58.1% for transcoders) but a higher proportion retained at Layer 5 (6.8% vs. 1.8%).

4.5.2 Early Layers: Simple Board Patterns

Early-layer neurons encode simple board-state predicates. At Layer 0, 58.1% of transcoder neurons achieve $F_1 > 0.99$ with tree depths ≤ 4: sequence membership, position-occupancy conjunctions, recent move identity, and current-player indicators.

Transcoder case study: Layer 0, neuron N2307. $F_1 = 1.000$, depth 1. Rule: "sequence does not contain position 27 AND does not contain position 22" (two of the four opening moves). The neuron is active at steps 0–5 (magnitude 5.6–10.4), then deactivates at step 6 when position 27 is played. A prototypical binary sequence tracker exemplifying Layer 0's role as a fact encoder.

Transcoder case study: Layer 1, neuron N1700. $F_1 = 0.959$. Rule: "piece count ≥ 28 AND legal moves < 3" (endgame detector). Unlike Layer 0's binary patterns, N1700 shows graded activation that increases toward game end (0.9 at step 27, 11.0 at step 32), illustrating Layer 1's integration of multiple facts into higher-order features.

SAE case study: Layer 0, neuron N5341. $F_1 = 1.000$, depth 1. Rule: "sequence does not contain position 8". Active at steps 0–2, off at step 3 when position 8 is played. The simplest negative sequence tracker, confirming SAE features decompose into monosemantic units at early layers.

SAE case study: Layer 1, neuron N1935. $F_1 = 0.994$, depth 2. Rule: "sequence does not contain position 16 AND not contain position 17". Active for 27 consecutive steps, deactivating when position 16 is played. Sequence trackers remain robustly extracted across the first two layers despite Layer 1's lower overall interpretability (28.2% vs. 35.8% at $F_1 > 0.99$).

4.5.3 Middle Layers: Moderate Interpretability

Middle layers show moderate interpretability scores while remaining partially interpretable. Layer 2: 10.4% at $F_1 > 0.99$, 67.5% at $F_1 > 0.90$. Layer 3: 4.9% and 48.7%. These layers are a transitional zone where simple rules are composed into more complex features.

Transcoder case study: Layer 2, neuron N4349. $F_1 = 0.993$, depth $\sim$4. Rule: "all four corners occupied AND left/bottom/right edges full". This is a terminal board-state detector, because it is active only at step 32 with rule indicating that the board is almost filled. This neuron has 29 duplicates, possibly because the transcoder training tends to allocate more capacity to more salient features.

SAE case study: Layer 2, neuron N1743. $F_1 = 1.000$, depth 1. Rule: "sequence does not contain position 16". This neuron is active for 26 steps until position 16 is played, confirming that the residual stream preserves the information throughout the sequence.

SAE case study: Layer 3, neuron N0771. $F_1 = 1.000$, depth 1. Rule: "sequence does not contain position 27". This is a perfect sequence tracker neuron. However, Layer 3 SAE's per-step explanation rate dropped to 29–40% during mid-games, making these simple neurons the most understandable ones among other complex features.

4.5.4 Late Layers: Complex Representations

Late-layer neurons form complex representations that are hard to be modeled by shallow decision trees. Layer 4: 4.1% at $F_1 > 0.99$ (89.1% dead). Layer 5: 1.8% (61.5% dead). In the 70.8% of the 2,938 living Layer 5 neurons with $F_1 > 0.90$, most of them are moderate fits from a deep, complex tree that are not really human-understandable.

Transcoder case study: Layer 4, neuron N3534. $F_1 = 0.959$, depth 1. Rule: "position 8 is opponent's piece". This relative board state feature confirm previous linear probing results. It is also a rare depth-1 example at Layer 4, as 74% of Layer 4's interpretable neurons need depth 5–10, and per-step interpretability is only $\sim$25.4%. Most uninterpretable neurons here appears to be computing flip directions and positions.

4.6 Truly Useful Neurons

Table 4.4 reports the minimum number of transcoder neurons needed to maintain top-k accuracy $\geq 99.5\%$, determined by binary search and pruning. The effective width turned out to be way smaller than the dictionary size: Layer 1 needs only 205 of 820 living neurons. At Layer 0, 51.5% of minimum-useful neurons have $F_1 > 0.99$; at Layer 5, only 0.25%. This implies that the Transformer inner representations have redundant capacity for prediction, and also confirms the interpretability trend even among functionally essential neurons.

4.7 Causal Intervention Experiments

We aim to upgrade the correlational evidence from decision trees to causal claims by directly modifying feature activations and checking if the model's output changes as expected (see Chapter 3 for the framework). Our primary metric is $P(\text{only-CF})$, the probability assigned to moves that are legal only under the counterfactual intervention. We also verify the specificity by conducting control interventions on unrelated neurons.

All six experiments (four transcoder, two SAE) are conducted on a single 32-move game: BOS $\to$ 13 $\to$ 7 $\to$ 8 $\to$ 9 $\to$ 28 $\to$ 27 $\to$ 26 $\to$ 33 $\to$ 1 $\to$ 35 $\to$ 22 $\to$ 25 $\to$ 30 $\to$ 19 $\to$ 32 $\to$ 24 $\to$ 3 $\to$ 4 $\to$ 10 $\to$ 31 $\to$ 18 $\to$ 12 $\to$ 29 $\to$ 34 $\to$ 5 $\to$ 23 $\to$ 16 $\to$ 11 $\to$ 17 $\to$ 2 $\to$ 6 $\to$ 0.

Table 4.4: Minimum number of transcoder neurons required per layer to maintain top-k accuracy at or above 99.5% (or the layer's natural accuracy if below this threshold), determined by iterative pruning of the least-active neurons. Note that Min. Useful is computed over all living neurons and may exceed the "Analyzed" count in Table 4.2, because some rarely-activating neurons contribute to accuracy but lack sufficient samples for decision tree training. The F_1 columns report counts only among neurons that were both minimum-useful and DT-analyzed.

Layer	Min. Useful	Verified Acc. (%)	F1>0.99	F1>0.90
0	291	99.39	150	219
1	205	99.50	38	185
2	743	99.48	67	437
3	301	99.50	23	258
4	265	99.50	10	201
5	800	99.58	2	494

4.7.1 Experiment 1: Transcoder Move Detector Swap (Layer 0)

At step 15 (current move = position 32), we swap the activations of N2871 ("move is 32", $F_1 = 1.000$) and N0864 ("move is 18", $F_1 = 1.000$), telling the model the current move is 18 rather than 32.

The result is striking: $P(\text{legal only after } 32 : \{23, 31\})$ drops from 12.7% to 0.2%, while $P(\text{legal only after } 18 : \{17\})$ rises from 0.05% to 14.1% (282×)—the model redistributes predictions as if move 18 had been played.

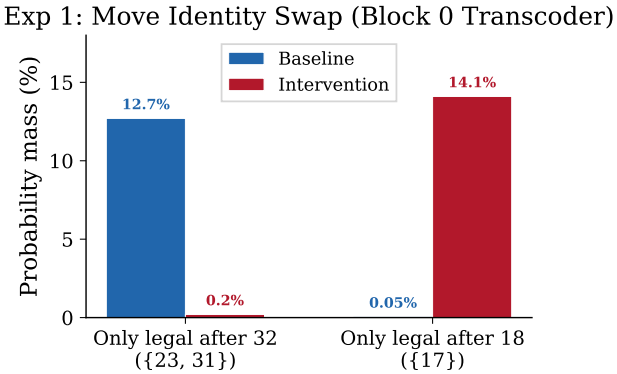

Figure 4.7: Experiment 1: Move-specific neurons activation swap intervention at Layer 0. We swapped the activations of neurons for "move is 32" and "move is 18", and caused the model to give probability mass to legal moves as if position 18 was played, which demonstrates that Layer 0 move-specific neurons are causally used in downstream legal move prediction.

4.7.2 Experiment 2: Board State Intervention (Layer 4)

At step 8, an opponent piece is already at position 8. We zero the 9 Layer 4 neurons whose rules are related to position 8's board state, simulating the counterfactual state that position 8 is either empty or occupied by the current player.

$P(\text{only-CF } \{6\})$ increases from 0.02% to 7.37%, while $P(\text{only-real } \{2, 3\})$ decreases from 25.68% to 21.84%. A control intervention on an unrelated neuron produces no change

(P(only-CF) = 0.02%). However, the same intervention at Layers 2–3 has negligible effect, this is either because later layers have the same abilities as previous layers so that it can recover, or Layers 2–3 have other redundant features that can compensate for the missing ones.

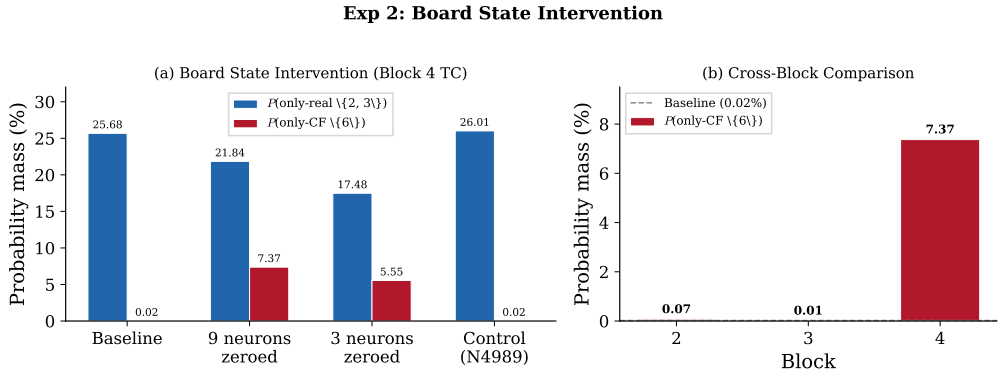

Figure 4.8: Experiment 2: Board state intervention at Layer 4 transcoder. Zeroing the 9 neurons that are related to the opponent's occupancy of position 8 lead to a huge increase in the probability of counterfactual-only legal moves, while a control intervention on an unrelated neuron produces no measurable effect.

4.7.3 Experiment 3: Sequence History Intervention (Layer 0)

Neuron N6389 ($F_1 = 1.000$, depth 1) encodes "sequence does not contain position 19" (active at steps 0-12 before 19 was played).

All-step: zeroing N6389 at every step suppresses $P(19)$ to <0.11% at steps 9–13. *Single-step*: zeroing at step 3 alone halves $P(19)$ from 40.51% to 21.40%. The increasing effect size reflects accumulated evidence as the game progresses.

4.7.4 Experiment 4: Composite Feature Intervention (Layer 1)

Neuron N5080 ($F_1 = 0.991$, depth 8) encodes a composite rule combining sequence history, board state, and flip information. At step 3, zeroing N5080 drops $P(7, \text{illegal})$ from 19.58%

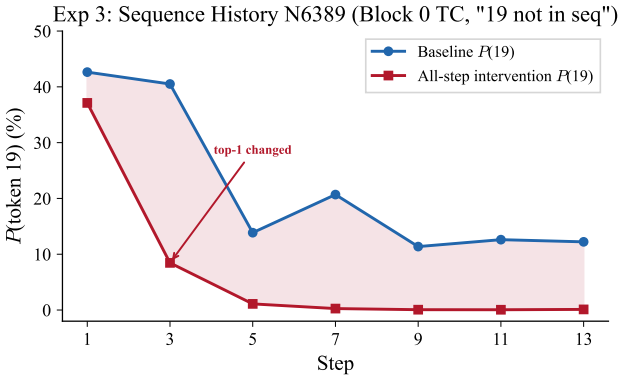

Figure 4.9: Experiment 3: Sequence history intervention at Layer 0 transcoder. Zeroing neuron N6389 ("sequence does not contain position 19") suppresses the model's probability of predicting move 19. The all-step variant (left) shows cumulative suppression strengthening over game steps; the single-step variant (right) demonstrates that a single-layer, single-step intervention reduces $P(19)$ from 40.5% to 21.4%.

to 6.20% and increases $P(\text{legal})$ from 75.1% to 91.4%. This intervention actually improves prediction quality, showing that not all features contribute positively at every step, and not just the features that are related to predictions will be activated.

4.7.5 Experiment 5: SAE Sequence History (Layer 2)

Layer 2 SAE neuron N1743 ($F_1 = 1.000$, depth 1): "sequence does not contain position 16". Zeroing this neuron out decreases $P(16)$ by >90% at all 14 legal steps. The effect is comparable to the transcoder intervention (Experiment 3), confirming that residual-stream sequence information is actively used by the model, not passively stored.

4.7.6 Experiment 6: SAE Board State Intervention (Layer 4)

We zero out 10 Layer 4 SAE neurons that are related to opponent occupancy at position 20. $P(\text{only-CF } \{2, 10, 23, 32\})$ rises from 19.98% to 34.06% (+14.08pp); a control intervention shows little change (20.37%). A cross-check at position 14 produces a 20.5× increase

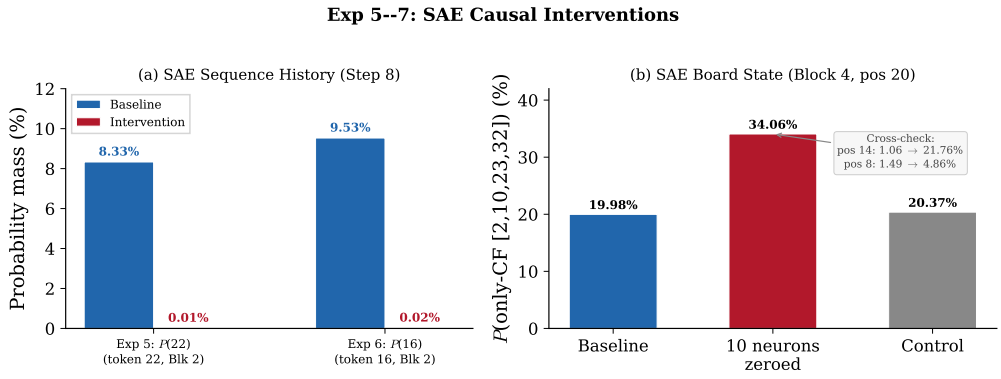

Figure 4.10: Experiments 5 and 6: SAE causal interventions. Left: Sequence history intervention at Layer 2 (neuron N1743, "sequence does not contain position 16"), showing suppression of $P(16)$ by more than 90% across all steps where position 16 is a legal move. Right: Board state intervention at Layer 4, where zeroing 10 opponent-occupancy neurons increases counterfactual-only move probability from 20.0% to 34.1%.

(P(only-CF): 1.06% $\rightarrow$ 21.76%).

This demonstrates that SAE residual-stream features actively influence the model's predictions, extending Nanda et al. [NLW23]'s whole-tile interventions to the level of individual sparse features.

4.7.7 Causal Intervention Summary

Table 4.5 and Figure 4.11 summarize all six experiments. Several patterns emerge: both transcoder and SAE features are causally effective; different layers encode different abstraction levels (Layer 0: move identity; Layer 1: composites; Layer 4: board state); features show cross-layer redundancy, but only specific layers are causally active at a given step; and control interventions consistently show null effects (±0.4pp). This SAE experiment extend prior causal interventions [NLW23] on residual stream representations to individual sparse features, providing finer-grained evidence that residual-stream representations actively drive predictions.

Table 4.5: Summary of all six causal intervention experiments. Each row reports the architecture, target layer, feature type, and key quantitative result. Both transcoder and SAE features across multiple layers and feature types produce large, specific shifts in the model's output distribution upon intervention.

Exp	Architecture	Layer	Feature Type	Key Result
1	Transcoder	0	Move identity	P(only-CF): 0.05%→14.1%
2	Transcoder	4	Board state	P(only-CF): 0.02%→7.37%
3	Transcoder	0	Seq. history	$P(19)$: 40.5%→8.5%
4	Transcoder	1	Composite	P(legal): 75.1%→91.4%
5	SAE	2	Seq. history	$P(16)$: avg $\Delta = -11.3\%$ over 14 steps
6	SAE	4	Board state	P(only-CF): 20.0%→34.1%

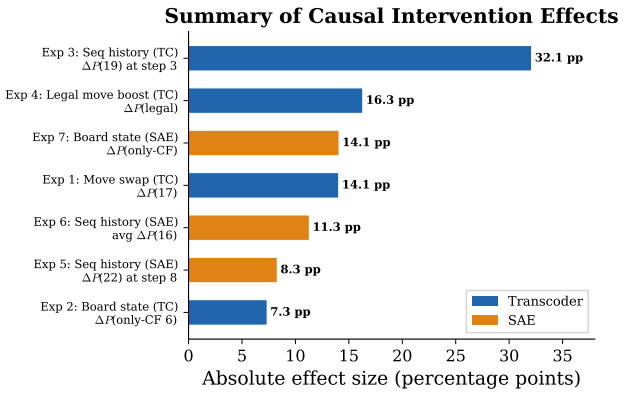

Figure 4.11: Each panel shows the gain in post-intervention probability of the target metric (counterfactual-only move probability or target move probability), together with any control conditions. All interventions produce large, targeted effects.

4.8 Random Forest vs Decision Tree

To understand whether the interpretability quality is limited by classifier capacity or feature set complexity, we compare single decision trees (max depth 10) with random forests (100 trees, max depth 10) across all living neurons using the same features described in Chapter 3.

4.8.1 Neuron Classification

We classify each neuron by generalization F_1: **Simple** (both > 0.9), **Complex but Explainable** (only RF > 0.9), **Feature Insufficient** (neither > 0.9), and **DT Superior** (only DT > 0.9, typically RF overfitting).

4.8.2 Results

Figure 4.12 shows the per-layer classification results. All layers and both architectures mostly contain Simple neurons, confirming that a single decision tree is complex enough for most of the neurons.

4.8.3 Key Findings

Four findings stand out. First, Simple neurons dominate (TC 70.9%, SAE 70.3%), validating decision trees as the primary tool. Second, SAEs produce cleaner features at Layers 0–4 (85.4% Simple vs. TC 68.5%), likely because the residual stream is more linearly structured than MLP computations. Third, the proportion of *Complex but Explainable* neurons is small (SAE 7.7%, TC 13.3%), so ensemble power is rarely the limiting factor. Fourth, the true bottleneck is feature engineering: Feature Insufficient accounts for 13.1% (TC) and 20.0% (SAE) of neurons, encoding concepts not captured by our $\sim$700 hand-crafted boolean features.

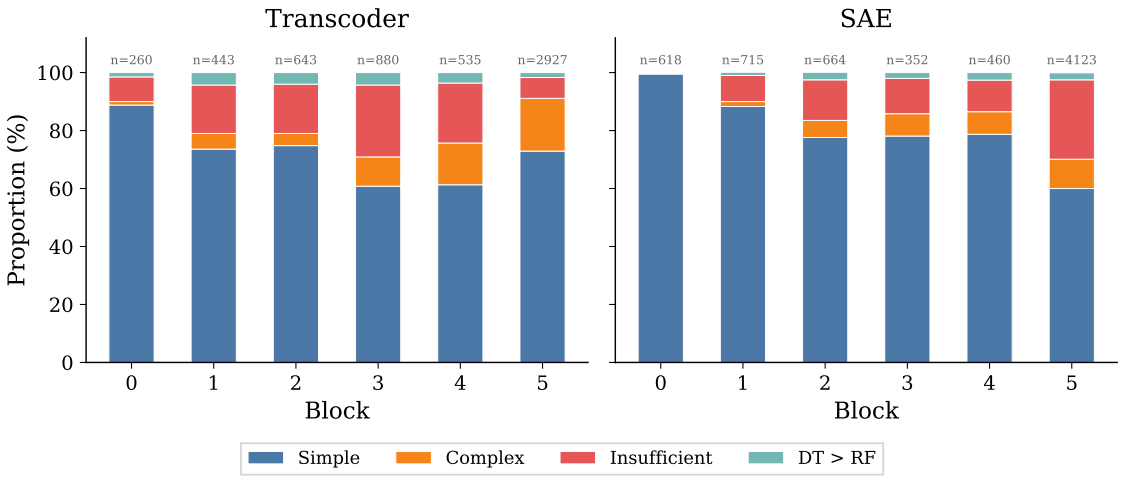

Figure 4.12: Per-layer distribution of the four neuron categories (Simple, Complex but Explainable, Feature Insufficient, DT Superior) for transcoders and SAEs. The Feature Insufficient proportion increases with depth, indicating that the interpretability bottleneck at deeper layers lies in the feature vocabulary rather than classifier capacity.

4.8.4 Deep Dive Verification

Table 4.6 showcases five manually inspected Layer 4 neurons. Complex neurons (N8007, N6482) require multi-feature interactions that ensembles can resolve. The Feature Insufficient neuron (N6674) sees only marginal RF improvement (+4.5pp), confirming inadequate features. DT Superior neurons (N4431, N7706) show RF overfitting on sparse positives.

4.8.5 Conclusion: Justifying Decision Trees

In summary, we justify decision trees as our primary tool for our analysis. First, random forests provide negligible aggregate improvement (Simple >70% for both architectures), and their rules are not human-understandable. The most common failure mode is Feature Insufficient (13–20%), not classifier capacity, indicating that designing more expressive features is the most productive direction for improvement.

Table 4.6: Manual verification of five transcoder neurons from Layer 4, one from each classification category. For each neuron, we report the decision tree and random forest F_1 scores, the number of positive activation examples, the assigned category, and qualitative notes. These cases confirm that the four-way classification captures genuine differences in neuron complexity and feature expressibility.

Neuron	DT F1	RF F1	Positive	Category	Notes
N8007	0.889	1.000	50	Complex	Requires conjunction of 4+ features
N6482	0.759	0.933	158	Complex	Complex spatial pattern
N6674	0.818	0.863	1000	Insufficient	RF only marginal improvement
N4431	1.000	0.929	30	DT Superior	RF overfits on sparse data
N7706	0.923	0.833	71	DT Superior	Simple rule

CHAPTER 5

Discussion

5.1 The Interpretability Trend

The interpretability trend mentioned in Chapter 4 shows different layers model different types of features.

Layer 0 answers "Where are the pieces?", which are local, binary facts that can be captured by depth-1 or depth-2 trees. Layers 1–2 shift to "What patterns are present?", which are edge occupancy, endgame indicators, multi-position relationships.

Layers 3–5 address "What moves are legal?", which requires multi-directional flanking checks that are hard to be decomposed along single feature axes. This is reflected in L_0 increasing largely from $\sim$14 (Layer 0) to 238 (Layer 5, TC) and in the shift to maximum-depth trees with moderate F_1.

This well aligns with Yuan and Søgaard [YS25] and jylin04 [jyl24]'s "bag of heuristics" assumption, that shallow layers have local heuristics that are overwhelmingly interpretable, while deeper layers use multi-step reasoning that resist decomposition.

5.2 Causal Validation: From Correlation to Causation

The causal interventions (Section 4.7) elevate our analysis from correlation to causality. By extending Nanda et al. [NLW23]'s linear-probe interventions with finer-grained sparse dictionary features, we can manipulate specific computational units with high granularity

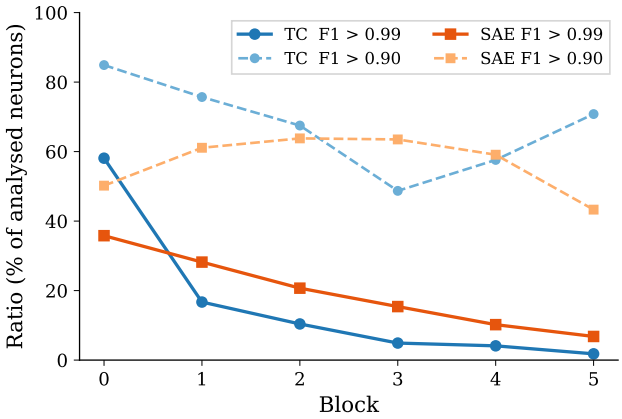

Figure 5.1: The interpretability trend across network depth for transcoders and SAEs. The proportion of features with decision tree $F_1 > 0.99$ declines rapidly from Layer 0 to Layer 5, mirroring a qualitative shift from locally interpretable features in shallow layers to distributed, multi-directional representations in deeper layers that are hard to be decomposed.

rather than coarse whole-tile representations on the whole residual stream.

These experiments confirm that we found causal features. With intervention, we can achieve hundreds of times probability increases, while control groups show negligible effects.

5.3 Implications for Mechanistic Interpretability

5.3.1 Sparse Transcoders as an Analysis Tool

In shallow layers, the transcoder effectively decomposes MLP computations, with near-perfect reconstruction (top-k accuracy $\geq$99.29%) confirming that the decomposition is faithful. In deeper layer the sparse model still replicates model behavior, but they do not reveal how they compose into higher-level algorithms.

Comparing the two architectures, SAEs consistently yield higher interpretability rates than transcoders at Layers 0–4 (85.4% Simple vs. 68.5%). This is expected: the SAE decomposes the residual stream, which accumulates all information from preceding layers,

whereas the transcoder decomposes only the MLP's incremental contribution at each layer (if the MLPs do not wipe out information from the residual stream). A richer, more complete representation naturally gives cleaner and more interpretable sparse features.

5.3.2 The Role of the Analysis Method

The RF vs. DT diagnostic (Section 4.8) shows that the interpretability bottleneck is not classifier capacity. Random forests barely improved single tree results (Simple >70% for both architectures, Complex but Explainable only 7.7–13.3%).

The dominant failure mode is Feature Insufficient (13.1–20.0%), directing future effort toward richer feature vocabularies rather than more expressive classifiers.

Circuit tracing [ALP25, Ant25] could reveal compositional structure across layers, but it requires interpretable features at every depth. With under 2% interpretable at Layer 5, complete tracing is currently impossible.

Our evaluation also differs from large-model approaches that rely on activation explanations scored by human raters or LLMs, with no guarantee of correctness or concept quality [GTT24, TCM24], because the Othello domain provides exact, verifiable predicates. We can report quantitative F_1 scores, which is a much harder task to achieve in natural language settings.

5.4 Limitations and Future Work

Domain specificity. The high decision tree F_1 scores reflect Othello's constrained, rule-based space. Applying the same pipeline to other domains with clear ground truth, such as small language models trained on modular arithmetic or formal grammars, would test how far these findings extend. Domains that lack explicit, limited rules may require fundamentally different interpretability approaches.

Board size. Our experiments use a 6×6 board, which is simpler than the standard 8×8 game. Scaling up to the standard 8×8 game or even larger boards would test whether the interpretability trend remains consistent with game complexity and whether sparse decomposition remains effective at larger scale.

Feature engineering bottleneck. The Feature Insufficient category (13–20%) quantifies the gap left by our hand-designed feature set. Enriching the vocabulary with more complex features could possibly resolve some of these neurons. But it is harder to design features that are both complex and interpretable, especially for the later layers.

Analysis method expressiveness. Decision trees use axis-aligned splits and cannot capture non-axis-aligned concepts. The RF diagnostic partially addresses this but does not yield human-understandable rules.

Fixed dictionary size. We use $D = 8{,}192$ for all layers. However, living neuron counts range from 532 (Layer 3 SAE) to 5,156 (Layer 5 SAE), and duplicate features are still present. We could explore a more computationally efficient setup in future work.

Static, per-neuron analysis. We analyze single-layer neurons without cross-layer circuit tracing [ALP25, Ant25], as circuit tracing methods require interpretable features at every depth.

Causal validation coverage. All our intervention experiments were conducted on very specific features on specific tokens in specific sequences. Because each sparse feature corresponds to a different rule and only activates on a small percentage of game sequences, it is hard to perform large-scale interventions across many board positions as Nanda et al. [NLW23] did with linear probe directions. Extending causal validation to a larger and more diverse set of games remains important future work.

Optimizer configuration. Setting $\beta_1 = 0$ was critical for transcoder training but was arrived at through non-systematic experimentation. A systematic ablation of β_1 and its interaction with learning rate, dictionary size, and sparsity schedules could yield further improvements.

CHAPTER 6

Conclusion

We conclude our four findings.

First, JumpReLU transcoders and SAEs achieve near-perfect sparse decomposition (top-k accuracy $\geq$99.29% vs. the original 99.62%), with $\sim$25 of 8,192 features active per token. Because reconstruction is so close to perfect, we can confidently attribute any remaining uninterpretable features to representation properties rather than training artifacts, eliminating the ambiguity that is hard to quantify in large-model settings.

Second, we conduct six causal intervention experiments to validate that individual sparse dictionary features are causally used by the model. Extending Nanda et al. [NLW23]'s relative board state interventions via linear probes, our experiments used finer-grained sparse features and confirmed that MLPs and residual-stream representations do actively drive predictions.

Third, the proportion of features with $F_1 > 0.99$ declines from 58.1% (Layer 0) to 1.8% (Layer 5) for transcoders and from 35.8% to 6.8% for SAEs, demonstrating the transition from simple, interpretable local patterns to complex, distributed representations that are hard for rule-based decomposition.

Fourth, comparing random forests against single decision trees, we find that the interpretability bottleneck is feature engineering, not classifier capacity: over 70% of neurons are "Simple" ($F_1 > 0.9$ with a single tree), only 7.7–13.3% are "Complex but Explainable", and 13–20% are "Feature Insufficient". This directs future effort toward better feature vocabularies rather than more powerful classifiers.

REFERENCES

[ALP25] Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, Joshua Batson, et al. "Circuit Tracing: Revealing Computational Graphs in Language Models." *Transformer Circuits Thread*, 2025.

[Ant25] Anthropic. "Tracing Attention Computation Through Feature Interactions." *Transformer Circuits Thread*, 2025.

[BTB23] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. "Towards monosemanticity: Decomposing language models with dictionary learning." *Transformer Circuits Thread*, 2023.

[CER24] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. "Sparse autoencoders find highly interpretable features in language models." *International Conference on Learning Representations*, 2024.

[DCN24] Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. "Transcoders find interpretable LLM feature circuits." *arXiv preprint arXiv:2406.11944*, 2024.

[DHI25] Jason Du, Kelly Hong, Alishba Imran, Erfan Jahanparast, Mehdi Khfifi, and Kaichun Qiao. "How GPT learns layer by layer." *arXiv preprint arXiv:2501.07108*, 2025.

[EHO22] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. "Toy models of superposition." *arXiv preprint arXiv:2209.10652*, 2022.

[GTT24] Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. "Scaling and evaluating sparse autoencoders." *arXiv preprint arXiv:2406.04093*, 2024.

[jyl24] jylin04. "OthelloGPT learned a bag of heuristics." LessWrong / AI Alignment Forum, 2024.

[LHB23] 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." *International Conference on Learning Representations*, 2023.

[MKT22] Thomas McGrath, Andrei Kapishnikov, Nenad Tomašev, Adam Pearce, Martin Wattenberg, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik.

"Acquisition of Chess Knowledge in AlphaZero." *Proceedings of the National Academy of Sciences*, **119**(47):e2206625119, 2022.

[NLW23] Neel Nanda, Andrew Lee, and Martin Wattenberg. "Actually, Othello-GPT has a linear emergent world representation." *arXiv preprint arXiv:2310.07582*, 2023.

[QWZ25] Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. "Gated Attention for Large Language Models: Non-linearity, Sparsity, and Attention-Sink-Free." *Advances in Neural Information Processing Systems*, 2025.

[RCS24] Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Anca Dragan, and Rohin Shah. "Jumping ahead: Improving reconstruction fidelity with JumpReLU sparse autoencoders." *arXiv preprint arXiv:2407.14435*, 2024.

[SAL24] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. "RoFormer: Enhanced transformer with rotary position embedding." *Neurocomputing*, **568**:127063, 2024.

[SWM25] Aditya Singh, Zihang Wen, Srujananjali Medicherla, Adam Karvonen, and Can Rager. "Automatically Finding Rule-Based Neurons in OthelloGPT." *arXiv preprint arXiv:2511.00059*, 2025.

[TCM24] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, Alex Tamkin, Esin Durmus, Tristan Hume, Francesco Mosconi, C. Daniel Freeman, Theodore R Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. "Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet." *Transformer Circuits Thread*, 2024.

[TWL22] Shubham Toshniwal, Sam Wiseman, Karen Livescu, and Kevin Gimpel. "Chess as a Testbed for Language Model State Tracking." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 11385–11393, 2022.

[YS25] Yifei Yuan and Anders Søgaard. "Revisiting the Othello World Model Hypothesis." *arXiv preprint arXiv:2503.04421*, 2025.