TRACING THE INTERNAL COMPUTATION OF A CHESS PLAYING TRANSFORMER

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern transformer neural networks achieve strong master-level performance in chess, yet a mechanistic understanding of how their internal computations produce move decisions remains limited. Building on advances in mechanistic interpretability, we adapt Low-Rank Sparse Attention (Lorsa) to LC0's policy network and combine it with Transcoders to decompose attention and MLP computations throughout the model into a unified set of sparse features. We also extend circuit tracing to LC0's bilinear policy output head. We introduce a chess-structured taxonomy and an agentic pipeline that proposes feature interpretations and, when possible, translates them into rule-based tests grounded in chess structure. On this foundation, we construct end-to-end attribution graphs that trace attributed pathways from board inputs through sparse features to move logits. Through case studies, we uncover interpretable tactical computations underlying the model's decisions for specific input positions and target moves. Moving beyond position-specific cases, we introduce quantitative metrics showing that computations are largely separated across candidate moves and highly branched within each move, while feature attribution gradually concentrates on the corresponding source and target squares across layers. Our code is available at `https://anonymous.4open.science/r/Leela-SAEs-446A`.

1 INTRODUCTION

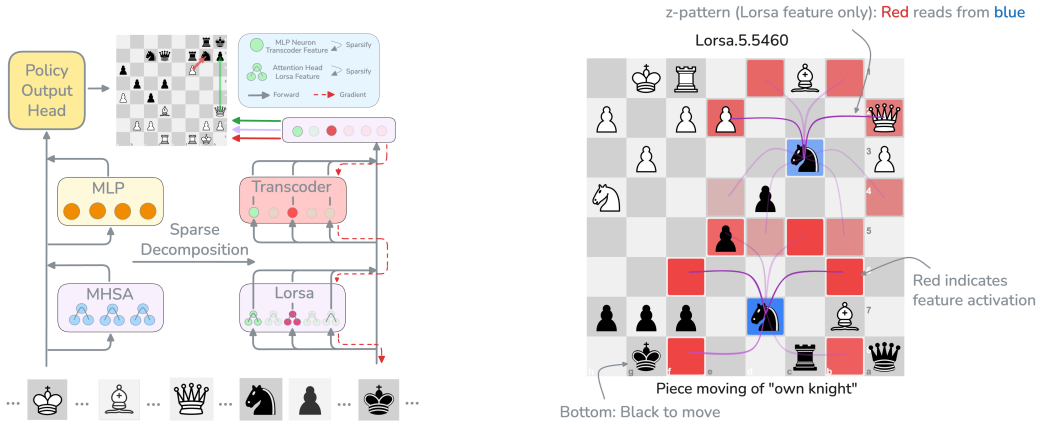
Modern neural networks have demonstrated remarkable abilities in reasoning tasks such as board games Tesauro (1994); Campbell et al. (2002); Silver et al. (2016; 2017), mathematics Seo et al. (2015); Lightman et al. (2024); Romera-Paredes et al. (2024), and coding Li et al. (2022). Building on the MCTS-based self-play training framework Silver et al. (2017), modern Transformer models have achieved remarkable playing strength in chess. Among them, the **Leela Chess Zero (LC0)** series achieves master-level performance even without search Ruoss et al. (2024); Leela Chess Zero Team (2024); Monroe & Team (2024); Monroe et al. (2026); Milosz et al. (2026).

Despite LC0's strong performance, we still lack an end-to-end mechanistic view of how its internal computations transform a board state into move decisions. Prior studies have identified localized chess concepts McGrath et al. (2022) or analyzed only a few attention heads Jenner et al. (2024), but not how the full policy network produces move decisions.

Neural network computations are often distributed across dense, superposed representations, making individual neurons or attention heads difficult to interpret Elhage et al. (2022); Jermyn et al. (2024); Lindsey et al. (2024). Sparse decomposition methods address this problem by decomposing these computations into more interpretable features. In particular, Transcoders extract interpretable features from MLP modules Ge et al. (2024); Dunefsky et al. (2024); Ameisen et al. (2025), while Low-Rank Sparse Attention (Lorsa) extends this decomposition to attention modules He et al. (2026).

Transcoders and Lorsas provide a unified sparse decomposition of LC0 BT4's MLP and attention computations; Lorsa is adapted to bidirectional attention with dynamic Smolgen biases replacing positional encodings Leela Chess Zero Team (2024). We organize proposed feature interpretations with a chess-structured taxonomy and use rule-based tests to validate the subset that can be reliably operationalized. Extending circuit tracing Marks et al. (2025); Ameisen et al. (2025) to LC0's bilinear policy output head, we construct **end-to-end attribution graphs** that reveal interpretable computational pathways from board embeddings to move outputs.

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.



(a) Overview of sparse decomposition and circuit tracing.

(b) Visualization interface.

Figure 1: **Overview and visualization interface.** Left: we use Transcoders and Lorsas to sparsely decompose LC0's MLP and attention modules, constructing interpretable attribution graphs through circuit tracing. Right: red squares indicate spatial feature activations. For Lorsa features, each activated square is associated with a z-pattern highlighting its attentional focus. Purple arrows connect activated squares to target squares with high attribution.

Through case studies, we show that attribution graphs help us understand the model's internal look-ahead, identify multiple strategic cues behind tactical decisions, and diagnose a rare failure in which BT4 favors an overly conservative move over an immediate checkmate.

Moving beyond individual positions, we statistically and causally analyze attribution graphs across many decisions through quantitative metrics. We find that different moves are supported by largely distinct and internally branched attribution graphs. At the same time, feature attribution for a move progressively converges on the source and target squares of the selected move across layers. This spatial convergence is accompanied by a semantic progression from piece-level features to richer tactical and move features.

Overall, our framework provides a feature-level account of LC0's attributed computation, together with a methodology for interpreting sparse features and selectively testing their semantic and causal relevance.

Roadmap. Section 2 introduces sparse decomposition and feature interpretation; Section 3 presents attribution-graph construction; Section 4 analyzes three case studies; Section 5 studies how move computation is organized through quantitative metrics.

Our main contributions are as follows.

- We adapt Transcoders and Lorsas to LC0 BT4 to sparsely decompose its MLP and Smolgen-biased bidirectional attention computations. We further develop an agentic pipeline that organizes proposed interpretations with a chess-structured taxonomy and, when possible, generates rule-based tests for their validation.
- We extend circuit tracing to LC0's bilinear policy output head, constructing attribution graphs and using case studies to interpret the model's internal computations.
- We introduce metrics showing that candidate moves are supported by largely distinct and internally branched feature circuits, while information progressively concentrates on the selected move's source and target squares, accompanied by a semantic evolution from piece-level to tactical and move-decision features.

2 PRELIMINARIES OF SPARSE DECOMPOSITION

2.1 SPARSE DECOMPOSITION

To disentangle computations under superposition Elhage et al. (2022); Jermyn et al. (2024), we employ a *sparse decomposition* approach Ameisen et al. (2025) by applying Transcoders for MLPs and Lorsas He et al. (2026) for attention modules. These replacement layers express the original module computations in terms of sparse, interpretable features that can be analyzed and intervened on individually.

Transcoder. A Transcoder approximates the computation of an MLP layer using a sparse, linear decomposition. Given the MLP input $x \in \mathbb{R}^d$, the Transcoder encoder produces sparse feature activations via a Top-K operation, $z = \text{Top-}K(W_E x + b_E)$, where $z \in \mathbb{R}^m$ contains only the K largest activations, with all other entries set to zero. The MLP output is then reconstructed as a linear combination of decoder vectors, $\hat{y} = W_D z + b_D$. This formulation mimics the MLP's behavior while allowing for individual feature analysis and causal intervention.

Low-rank Sparse Attention (Lorsa). A Lorsa disentangles attention heads in superposition into linear rank-1 *output value (OV)* heads, which can be interpreted as a sparse dictionary of features. Sparsity constraints ensure that only a few heads are activated per token position. Formally, each Lorsa feature takes in $x \in \mathbb{R}^{n \times d}$ and computes attention with a one-dimensional value, i.e.,

$$\begin{aligned} A^h &= \text{softmax}\left(\frac{xW_Q^h (xW_K^h)^\top}{\sqrt{d_h}} + B_{\text{Smolgen}}\right) \in \mathbb{R}^{n \times n}, \\ v^h &= x w_V^h \in \mathbb{R}^{n \times 1} \text{ (scalar value per token)} \end{aligned} \tag{1}$$

where $W_Q^h, W_K^h \in \mathbb{R}^{d \times d_h}$ and $w_V^h \in \mathbb{R}^{d \times 1}$ are parameters of a feature, and B_{Smolgen} is LC0's Smolgen-generated attention bias.

The scalar activation of the Lorsa feature at token position i is computed as

$$z_i^h = A_i^h v^h = \sum_{j=1}^{n} A_{i,j}^h \, (w_V^h)^\top x_j \in \mathbb{R}, \tag{2}$$

which can be decomposed into token-wise contributions, or *z-patterns*, z-pattern$_{i,j} = A_{i,j}^h (w_V^h)^\top x_j$, indicating how strongly token j contributes to the activation at position i. The feature output is then given by $\hat{Y}_h = z^h W_O^h$, with W_O^h playing a role analogous to the decoder weight W_D in Transcoders.

2.2 MEASURING INTERPRETABILITY FOR FEATURES

Interpretability Interface. We use an agentic pipeline to analyze top-activating samples, propose interpretations, and iteratively refine and validate them. For annotation, we predefine a chess-structured taxonomy of ten semantic categories and an `Uninterpretable` category, with few-shot examples for each category. The agent first labels a feature's taxonomy, then proposes an interpretation based on the taxonomy and iteratively refines it through validation. Appendix G details this agentic interpretation-to-validation workflow, while Appendix F.4 describes the taxonomy. Table 1 reports representative validation results used in Section 4.

Notation. We denote a feature instance as **`Tc.`**ℓ**`.f@`**s for Transcoder features and **`Lorsa.`**ℓ**`.f@`**s for Lorsa features. For example, **`Lorsa.7.14502@h7`** refers to Lorsa feature #14502 of layer 7, which activates at **`h7`**. Appendix F describes the taxonomy, notation, and verification primitives used throughout the paper.

Interpretability Assessment Protocols. Three chess players (Elo 2000+) and OpenAI Codex independently rated the activation consistency and complexity of Transcoder features, Lorsa features, and MLP neurons on a 1–5 scale Anthropic Interpretability Team (2024) (Table 2). For blind comparison, Lorsa z-patterns were hidden, though observing the full z-pattern would provide stronger

Table 1: Representative rule-based validation of Transcoder and Lorsa features. Colors indicate semantic categories: Det, Src/Tgt, Val, Cap, Pro, Tac, Spa, and Mov. For each feature shown, the agentic workflow maps its interpretation to an executable rule (Appendix F) and evaluates that rule in an independent random-position scan. Precision is the fraction of activated squares satisfying the rule, while recall is the fraction of rule-positive squares on which the feature activates; the full table appears in Table 6.

Feature ID	Interpretation	Verification Rule	Precision	Recall
Tc.0.12663	[Det]OwnPawn	Act. @ OwnPawn	100%	100%
Tc.14.120	[Src]Source square	Act. @ Top-1 predicted source	88.0%	1.7%
Tc.14.1214	[Tgt]Target square	Act. @ Top-1 predicted target	99.2%	78.7%
Tc.14.8947	[Val]High Value	Act. @ All squares in positions with predicted value > 0.8	89.9%	41.6%
Tc.14.4046	[Val]Low Value	Act. @ All squares in positions with predicted value < -0.8	83.1%	77.7%
Tc.4.5191	[Cap]Opp.Piece Captured by OwnPawn	Act. @ OwnPawn x Opp.Piece	97.0%	86.6%
Lorsa.4.5191	[Pro]Own piece protecting an Opp.Knight check target	Act. @ Opp.Knight check target protected by OwnPiece	67.3%	47.5%
Tc.13.10331	[Tac]Eat Opp.Pawn and check target/capture queen	Act. @ Opp.Pawn, Check Target/Capture Queen	74.0%	7.7%
Lorsa.8.16275	[Tac]Opp.Rook/Queen semi-open file check threat	Act. @ Check target of Opp.Rook/Queen, 2-move reachability, $\leq$2 blockers	98.3%	39.5%
Lorsa.2.3155	[Tac]OwnPawn break opening an Own Rook/Queen threat to Opp.Queen/King	Act. @ OwnPawn-controlled square, open file/rank, OwnRook/Queen threat to Opp.Queen/King	62.4%	18.6%
Lorsa.13.1520	[Tac]Three-step look-ahead alignment	Act. @ First-step endpoint aligned with third-step endpoint	90.7%	45.0%
Lorsa.7.14439	[Spa]Squares with the same color as OwnKing	Act. @ Square with the same color as OwnKing	100%	8.6%
Lorsa.0.7083	[Spa]Opp.Rook rank-wise defensive coverage	Act. @ Squares in rank-wise defensive coverage Opp.Rook	99.3%	23.8%
Lorsa.8.9586	[Mov]OwnBishop movement	Act. @ Bishop-reach squares $\leftarrow$ OwnBishop	98.7%	85.8%

Table 2: Interpretability assessment. We report both human evaluation scores and automatic interpretability assessment results across 100 MLP neurons, Transcoder features and Lorsa features. Human scores are given on a 1–5 scale by three chess-player annotators (Elo 2000+), measuring activation consistency and complexity described in Appendix G.1.

Feature Type	Human Assessment		Automatic Assessment		
	Activation Consistency	Complexity	Activation Consistency	Complexity	Activation Consistency in Attribution Graphs
MLP	2.05 ± 1.04	1.45 ± 0.63	1.95 ± 1.24	1.36 ± 0.51	–
Transcoder	$\mathbf{4.15} \pm 1.15$	2.25 ± 0.81	$\mathbf{4.15} \pm 1.17$	2.46 ± 0.69	$\mathbf{4.56} \pm 0.57$
Lorsa	3.73 ± 1.39	$\mathbf{2.54} \pm 1.08$	3.70 ± 1.07	$\mathbf{2.55} \pm 0.82$	4.47 ± 0.51

interpretability. We do not directly compare Lorsa features with MHSA *query-key* patterns Jenner et al. (2024); Monroe & Team (2024), as Lorsa features reflect integrated *query-key-value interactions* rather than *query-key* patterns alone. Appendix F provides the taxonomy and selective rule-based verification details, and Appendix G describes the automatic interpretation protocol.

3 Attribution Graph Construction

To characterize the end-to-end attributed computation of BT4, we extend the circuit-tracing framework Ameisen et al. (2025) by jointly incorporating sparse decompositions of Lorsas and Transcoders. This allows us to construct an **end-to-end attribution graph** that traces attributed pathways through the model, while Lorsa z-patterns provide directional information about cross-token interactions.

In this section, we present the algorithm for constructing the **end-to-end attribution graph**. We also describe the **specific adaptations** required to apply Lorsa and circuit tracing to LC0 architecture.

3.1 Attribution Graph Construction

Following Shu et al. (2026), we construct attribution graphs over the sparse decomposition as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, where

$$\mathcal{V} = \mathcal{V}_{\text{embed}} \cup \mathcal{V}_{\text{transcoder}} \cup \mathcal{V}_{\text{lorsa}} \cup \mathcal{V}_{\text{error}} \cup \mathcal{V}_{\text{logit}}. \tag{3}$$

Here, $\mathcal{V}_{\text{embed}}$ and $\mathcal{V}_{\text{logit}}$ denote input token embeddings and move logits; $\mathcal{V}_{\text{transcoder}}$ and $\mathcal{V}_{\text{lorsa}}$ denote sparse MLP and attention features; and $\mathcal{V}_{\text{error}}$ denotes each module's reconstruction residual (original output minus sparse reconstruction), which preserves computation not captured by sparse features. We treat these nodes as unified read–write units with a decoder w_{dec}, except logits, and an encoder w_{enc}, except embeddings and reconstruction-error nodes.

For each edge $s \to t \in \mathcal{E}$, we define *direct interaction coefficient*

$$\alpha_{s \to t} = P^t_{i,j} w^{\top}_{\text{dec},s} w_{\text{enc},t}. \tag{4}$$

This coefficient represents the local linear effect of the source activation a_s on the pre-activation of target node t. Here, $P^t_{i,j}$ is the attention pattern of target node t from position i to position j, with $P^t = I$ for Transcoder features and logits. Following the gradient-based construction of Ameisen et al. (2025), we compute Eq. 4 by backpropagating from each target node while holding nonlinear feature gates, attention patterns, and LayerNorm denominators fixed.

Starting from the selected move logit, we trace backward under a fixed node budget. Let $\mathbf{M}$ denote the direct-interaction matrix, with $\mathbf{M}_{t,s} = \alpha_{s\to t}$. The direct attribution of s to t is therefore $a_s\alpha_{s\to t}$.

We define the all-path **attribution score** as

$$A_{s\to t} := \mathcal{A}_{t,s} = a_s \sum_{k=1}^{K} \left(\mathbf{M}^k\right)_{t,s}, \tag{5}$$

where K is the maximum path length in the graph. Thus, $A_{s\to t}$ aggregates attribution from s to t over all directed paths. We greedily expand the frontier feature with the largest attribution score to the selected move logit.

We inspect the resulting graphs through an interactive interface that displays each feature's activation pattern and top-activating samples as shown in Appendix H. We merge semantically similar features into **supernodes** to obtain a more concise attribution graph. We evaluate the fidelity at two levels: replacement fidelity and attribution faithfulness Ameisen et al. (2025); Appendix D provides the metrics and quantitative results.

3.2 LC0-Specific Attribution Adaptations

As shown in Figure 5, LC0 differs from standard language transformers in several ways that affect sparse decomposition and attribution tracing. We therefore adapt our Lorsa architecture and attribution-graph construction to two LC0-specific components.

Smolgen-biased bidirectional attention. LC0 applies bidirectional attention over fixed board-square sequences, using Smolgen-generated biases instead of RoPE. Lorsa follows this setup and initializes each Smolgen module from the corresponding MHSA block.

Q/K-separated policy output head. BT4 uses a bilinear policy output head rather than a linear unembedding. For a move $m = (i, j)$, the logit is

$$\ell_m = q_i^\top k_j, \qquad q_i = W_Q^{\mathrm{pol}} r_i, \quad k_j = W_K^{\mathrm{pol}} r_j, \tag{6}$$

where r_i denotes the final residual stream at token position i. We trace the two sides separately: the query-side trace freezes k_j and attributes through q_i, while the key-side trace freezes q_i and attributes through k_j. This yields two all-path attribution matrices, $\mathcal{A}^{(Q)}$ and $\mathcal{A}^{(K)}$, which we merge by aligning identical feature nodes and summing the scores from the two attribution matrices. The merged score is used only for ranking and graph construction, not as an additive logit decomposition; under zero-baseline conservation it sums to $2\ell_m$, while feature-to-feature attributions remain unchanged.

4 Case Studies

In this section, we use case studies to show that attribution graphs provide an interpretable view of BT4's internal computation. Rather than presenting full attribution graphs, we group features into semantically coherent supernodes and show representative supernodes that capture the main computational structure, as shown in Figure 2; full attribution graphs and other cases are provided in Appendix H.2.

Our case studies highlight three aspects of BT4's chess reasoning. In Sec. 4.1, we show that attribution graphs reveal internal look-ahead in a mate-in-two scenario, tracing the representation of a future mating continuation and its contribution to the current knight sacrifice. In Sec. 4.2, we show that BT4 selects a brilliant move by **considering and integrating diverse strategic information** including chess intuition such as opening the d-file for the rook. In Sec. 4.3, we show that comparing attribution graphs of different moves can diagnose tactical errors: even in a mate-in-one position, the model still

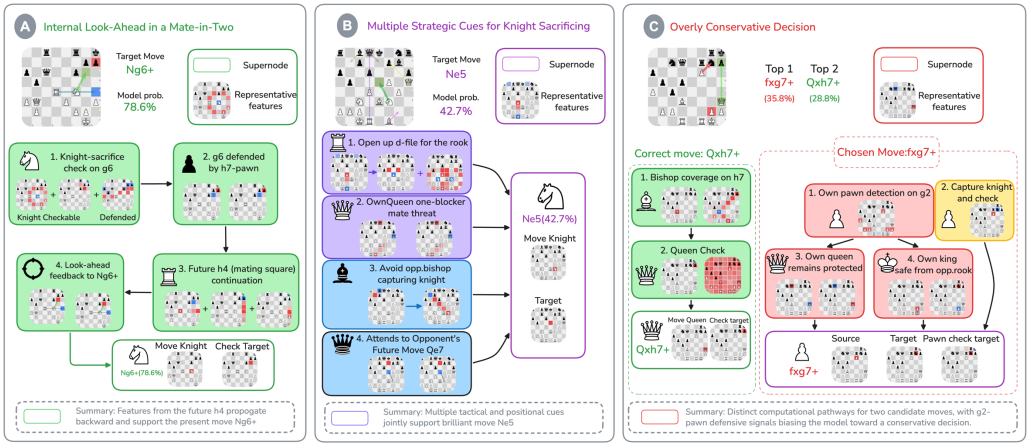


Figure 2: Overview of three case studies and key computational structures. We group the features into supernodes capturing the main computational structure. (A) Internal look-ahead in Section 4.1: the attribution graph links information about the later mating continuation to the target square of the current move. (B) Multiple strategic cues in Section 4.2: multiple strategic cues jointly support BT4's brilliant move, such as opening the d-file for the rook and creating a mating threat with the queen. (C) Conservative decision in Section 4.3: defensive signals associated with the g2 pawn bias the model toward the conservative move. Detailed attribution graphs are provided in Appendix H.2.

devotes substantial computation to piece protection relations and king safety, revealing a possible internal source of its uncertainty and conservative preference.

4.1 Internal Look-Ahead

Previous work Jenner et al. (2024) showed that LC0 encodes look-ahead information in representations of move-relevant squares. Figure 2A shows how the attribution graph traces the propagation of look-ahead information across layers. In early layers, the model represents that moving the knight to **g6** gives a check while **g6** is also defended by the opponent's pawn. In the middle layers, features associated with h4—the rook's mating destination two plies later—emerge in the attribution graph by attending to the rook on d4 and the king on h8. Higher-layer attention then routes attribution from features associated with the later **h4** back toward **g6**, promoting the move **Ng6+**. Together, these features localize a look-ahead pathway through which the future mating continuation supports the current move.

4.2 Multiple Strategic Cues

In the case shown in Figure 2B, the model finds a brilliant knight sacrifice, **Ne5**, which is the only move that opens up the position. We found that the model internally represents and uses multiple cues that jointly support this decision: the move opens the d-file for the rook, creates a mating threat with the queen, and prevents the opponent's bishop from developing and capturing the knight.

4.3 Diagnosing an Overly Conservative Decision

In the position shown in Figure 2C, BT4 selects a conservative and incorrect move **fxg7+** with 35.8% probability, while assigning only 28.75% to the correct mate-in-one move **Qxh7+**.

The attribution graphs reveal a source of this conservatism. A feature detecting the g2 pawn, **Tc.0.12663@g2**, supports **fxg7+** through downstream pathways associated with queen protection and king safety. In particular, the detection of the g2 pawn helps keep the queen protected in the exchange, thereby suppressing queen moves and shielding the king from direct exposure to the opponent rook after the **fxg7+** continuation. Removing the g2 pawn from the input position raises the probability of **Qxh7+** to 91.7% while directly ablating **Tc.0.12663@g2** raises it to 90.6%, indicating that this pawn-related signal has a substantial effect on the model's decision.

5 Organization of Move Computation in Attribution Graphs

Main finding. Attribution graphs reveal two organizational characteristics in the model's move computation. Features used for different moves are largely separated, while individual graphs are branched rather than fully connected. In addition, feature attribution progressively concentrates on the source and target squares of the selected move, accompanied by a semantic progression from piece-level representations to tactical and move-level representations.

5.1 Separation and Branching in Attribution Graphs

We characterize the organization of move computation through separation across candidate moves and branching within each move-specific attribution graph.

Feature overlap. We measure feature reuse between the attribution graphs for the top two candidate moves using post-pruning Jaccard similarity. Two feature nodes are considered the same if their $(\text{feature ID}, \text{board square}, \text{layer})$ tuples are identical. Across 1,000 different-source and 1,000 same-source positions, overlap is generally low, whereas same-source pairs show substantially higher overlap (Table 3(left); see Appendix E for details), consistent with reuse of shared-source representations such as piece-detection features.

Attribution-graph branching factor. We quantify within-graph branching using a normalized *attribution-weighted branching factor*, which measures the fraction of attribution-weighted feature pairs that are mutually unreachable in the directed graph. A single chain and a complete directed graph both have a branching factor of zero, whereas graphs with more distinct branches have a higher branching factor. The metric, rewiring procedure, and layer-wise results are detailed in Appendix E and Figure 10. Across 1,000 attribution graphs, Table 3(right) shows a higher branching factor for the original graphs than for constrained rewired baselines, indicating that the observed branching reflects the specific graph connectivity rather than these structural constraints alone.

Feature Overlap	
Different source	Same source
$5.76\% \pm 3.84\%$	$\mathbf{17.19\%} \pm 8.59\%$

Branching Factor B	
Original graphs	Rewired graphs
$\mathbf{75.7\%} \pm 7.58\%$	$51.6\% \pm 5.20\%$

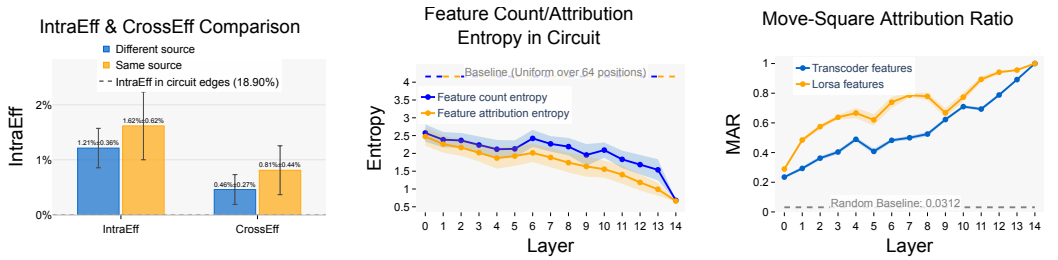
Table 3: Feature overlap between candidate-move attribution graphs (left) and attribution-weighted branching factors for original graphs and degree-preserving rewired baselines (right; 632 nodes per graph on average)

Causal influence. We next ask whether both cross-move separation and within-move branching are reflected in causal influence. Following the idea of zero patching Nanda (2022); Heimersheim & Nanda (2024), we patch each activated feature to zero in the residual stream and measure its effect on downstream features; formal definitions of the feature-to-feature, intra-circuit, and cross-circuit effects are given in Appendix E.

Figure 3a shows that $\mathrm{IntraEff}(\mathcal{V})$ exceeds on average $\mathrm{CrossEff}_{12}$. Features within the same attribution graph therefore exhibit stronger causal interactions than features across attribution graphs of different moves. However, both effects remain substantially smaller than the *feature-to-feature effect* between feature pairs connected by edges within an attribution graph, which has a mean effect of **18.9%**. Moreover, the correlation between edge weights and intervention effects ($r = 0.4390 \pm 0.1269$) further supports that edge weights capture causal influence.

5.2 Layer-wise Convergence Toward Move Representations

Having shown that attribution graphs for different moves use largely distinct feature sets and exhibit substantial internal branching, we next examine how their spatial organization evolves across layers. As illustrated by the attribution-graph cases in Appendix H.2, attribution progressively concentrates



(a) Intra-circuit effect vs. cross-circuit effect.

(b) Feature-count and attribution entropy.

(c) Move Square Attribution Ratio (MAR).

Figure 3: **Features within the same attribution graph exhibit stronger causal interactions, while feature count and attribution progressively concentrate on the selected move's source and target squares.** (a) Mean intra-circuit effects exceed cross-circuit effects between the top two moves. (b) Feature-count entropy (H_{number}) and attribution entropy ($H_{\text{attribution}}$) decrease across layers. (c) MAR increases across layers for significant and activated feature sets.

on the selected move's source and target squares. We quantify this trend using board-square metrics over attribution graphs for the top two moves from 1,000 randomly sampled chess positions.

Feature count and attribution entropy. We first measure how strongly features concentrate at the board-square level. For layer ℓ, let $c_i^{(\ell)}$ be the number of selected features at square i, and let $a_i^{(\ell)} = \sum_{f \in \mathcal{F}_i^{(\ell)}} |A_{f \to t}|$ be their total all-path attribution score to logit node t, defined in Eq. 5. We normalize them over board squares and compute feature count entropy and attribution entropy as

$$p_i^{(\ell)} = \frac{c_i^{(\ell)}}{\sum_j c_j^{(\ell)}}, \qquad q_i^{(\ell)} = \frac{a_i^{(\ell)}}{\sum_j a_j^{(\ell)}},$$
$$H_{\text{number}}^{(\ell)} = -\sum_i p_i^{(\ell)} \log p_i^{(\ell)}, \qquad H_{\text{attribution}}^{(\ell)} = -\sum_i q_i^{(\ell)} \log q_i^{(\ell)}. \tag{7}$$

Figure 3b shows that features become increasingly spatially concentrated across layers, as both feature count entropy and attribution entropy decrease and remain well below the uniform baseline.

Move-square attribution ratio. We next examine whether attribution progressively concentrates on the source and target squares of the selected move. Using the all-path attribution score in Eq. 5, we define the total attribution at square p and the layer-wise move-square attribution ratio (MAR) as

$$E_p^{(\ell)} = \sum_{f \in \mathcal{F}_p^{(\ell)}} |A_{f \to t}|, \qquad \text{MAR}^{(\ell)} = \frac{E_{\text{source}}^{(\ell)} + E_{\text{target}}^{(\ell)}}{\sum_{p=1}^{64} E_p^{(\ell)}}. \tag{8}$$

A higher MAR indicates that a larger fraction of the all-path attribution to the selected-move output is concentrated on its source and target squares.

As shown in Figure 3c, MAR increases across layers and reaches 1 in the final layer, which is structurally enforced by LC0's policy output head. MAR demonstrates progressive spatial information flow to the selected move's source and target squares.

Layer-wise Taxonomy Distribution. We further examine their semantic evolution across layers. We analyze the layer-wise distribution of features in attribution graphs under our taxonomy. Figure 4 shows a clear semantic progression: early layers are dominated by piece-detection features, middle layers contain more tactical features, and later layers contain increasing numbers of source- and target-square features. This distribution is consistent with a progression from piece-level to richer tactical and move-level representations.

Summary. Together, these results reveal how move computation is organized within attribution graphs: different moves are supported by largely distinct feature circuits with substantial internal

branching, while feature attribution progressively concentrates on the move's source and target squares across layers, suggesting a convergence of move-relevant information. Semantically, the representations progress from piece-level information to tactical and move-decision representations.

6 Discussion and Limitations

We use sparse decompositions of both MLP and attention modules to discover interpretable features. These features enable circuit tracing through attribution graphs that capture relational and multi-step structure beyond human concepts.

Beyond the three cases discussed in the main text and detailed in Figures 15, 17 and 18, Figures 19 and 20 provide two additional cases. The first shows BT4 detecting a queen–rook mate-in-one threat and routing the threat signal to a knight defense; the second shows an offensive checking move that also neutralizes the opponent's mating threat.

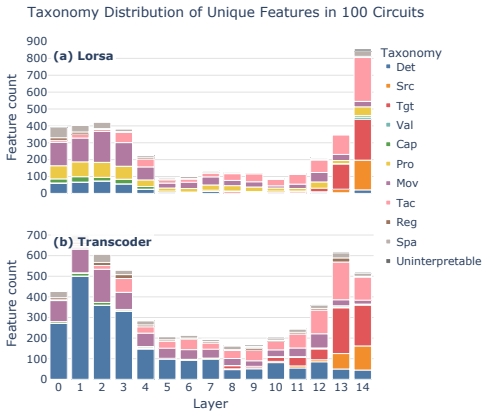


Figure 4: Layer-wise distribution of feature taxonomy within 100 randomly sampled attribution graphs.

6.1 Limitations

We discuss the limitations of our approach. These observations also highlight multiple directions in which future work may further improve upon our current framework.

Bilinear Interaction Approximation. For LC0's bilinear policy head, we trace the query and key side separately. This does not fully preserve multiplicative interactions between query- and key-side features. Future work could apply joint QK tracing to explicitly analyze these bilinear feature interactions Kamath et al. (2025).

Inhibition Circuits. Within the circuit tracing framework, our analysis is restricted to relationships among active features. Consequently, it is difficult to explain how and why some features are suppressed by others, or why certain moves are not selected Shu et al. (2026).

Uninterpretable features. Some features remain difficult to interpret, such as dense features Anthropic Interpretability Team (2024) and register features. Low recall for some tested features also suggests that our interpretations may miss finer-grained semantic conditions encoded by the model. This may stem from intrinsic model opacity, reconstruction error, representational demands imposed by LC0's other output heads during training, or high-dimensional features Engels et al. (2025); the same complexity may also contribute to the model's expressive capacity.

7 Conclusion

Through sparse decomposition, we construct dictionaries of computational features from both the MLP and attention modules of LC0 BT4, with a chess-structured taxonomy and rule-based tests providing semantic grounding for features. We use these features to construct attribution graphs across the model from board inputs to move logits. Case studies of attribution graphs trace a look-ahead pathway, the integration of multiple tactical cues, and diagnose a conservative decision. Quantitative analyses suggest that move-specific computation is externally separated and internally branched. Across layers, attribution becomes increasingly concentrated on the source and target squares of the selected move, accompanied by a progression from piece-level to tactical and move-level representations. Together, these results provide a feature-level account of LC0's computation and a methodology for discovering, interpreting, and validating sparse features, and guidance for studying models with stronger reasoning capabilities.

AI USE STATEMENT

In this work, we used generative AI tools to propose and refine feature-interpretation hypotheses, provide feedback on methodology and experiments, implement executable feature-verification tests, support qualitative feature analysis and result interpretation, and assist with translation. In particular, OpenAI Codex CLI agents form part of the experimental pipeline for feature interpretation, test generation, and blinded assessment, as described in Appendix G. We did not use generative AI to generate synthetic datasets or to clean or reformat the experimental data; proof-related disclosure categories are not applicable to this work. Additionally, generative AI tools assisted with code editing and literature search. The authors inspected AI-generated code and executable tests, checked interpretations against the underlying activation evidence, and manually reviewed the paper's claims, equations, citations, and experimental results. We take responsibility for the final content of this work, including all text, claims, code, and other artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The authors have read and adhere to the ICLR Code of Ethics. This study analyzes a publicly available chess model and board positions derived from the public `lichess_standard_rated` dataset. The analysis uses no player identifiers, attempts no inference about individual players, and involves no human subjects. We are not aware of additional material ethical risks arising from this work.

REPRODUCIBILITY STATEMENT

Appendix B specifies the analyzed model and architecture; Appendix C documents the dataset, training, and graph-construction settings; and Appendix D reports sparse-decomposition and attribution-graph fidelity. Appendix E defines the graph metrics and null baseline, while Appendices F and G provide the taxonomy, verification primitives, and agent prompts used for interpretation and evaluation.

REFERENCES

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL `https://transformer-circuits.pub/2025/attribution-graphs/methods.html`.

Anthropic Interpretability Team. Circuits updates – june 2024. `https://transformer-circuits.pub/2024/june-update/index.html`, 2024.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL `https://transformer-circuits.pub/2023/monosemantic-features/index.html`.

Murray Campbell, A. Joseph Hoane Jr., and Feng-Hsiung Hsu. Deep blue. *Artif. Intell.*, 134(1-2): 57–83, 2002. doi: 10.1016/S0004-3702(01)00129-1. URL `https://doi.org/10.1016/S0004-3702(01)00129-1`.

David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, Satvik Golechha, and Joseph Isaac Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=R73ybUciQF`.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=2dnO3LLiJ1.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable LLM feature circuits. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/2b8f4db0464cc5b6e9d5e6bea4b9f308-Abstract-Conference.html.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/toy_model/index.html.

Joshua Engels, Eric J. Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. Not all language model features are one-dimensionally linear. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=d63a4AM4hb.

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. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=tcsZt9ZNKD.

Xuyang Ge, Fukang Zhu, Wentao Shu, Junxuan Wang, Zhengfu He, and Xipeng Qiu. Automatically identifying local and global circuits with linear computation graphs. *CoRR*, abs/2405.13868, 2024. doi: 10.48550/ARXIV.2405.13868. URL https://doi.org/10.48550/arXiv.2405.13868.

Zhengfu He, Xuyang Ge, Qiong Tang, Tianxiang Sun, Qinyuan Cheng, and Xipeng Qiu. Dictionary learning improves patch-free circuit discovery in mechanistic interpretability: A case study on othello-gpt. *CoRR*, abs/2402.12201, 2024. doi: 10.48550/ARXIV.2402.12201. URL https://doi.org/10.48550/arXiv.2402.12201.

Zhengfu He, Junxuan Wang, Rui Lin, Xuyang Ge, Wentao Shu, Qiong Tang, Junping Zhang, and Xipeng Qiu. Towards understanding the nature of attention with low-rank sparse decomposition. In *The Fourteenth International Conference on Learning Representations, ICLR 2026, Rio de Janeiro, Brazil, April 23–27, 2026*. OpenReview.net, 2026. URL https://openreview.net/forum?id=9A2etpDFIB.

Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching. *CoRR*, abs/2404.15255, 2024. doi: 10.48550/ARXIV.2404.15255. URL https://doi.org/10.48550/arXiv.2404.15255.

Erik Jenner, Shreyas Kapur, Vasil Georgiev, Cameron Allen, Scott Emmons, and Stuart J. Russell. Evidence of learned look-ahead in a chess-playing neural network. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/37d9f19150fce07bced2a81fc87d47a6-Abstract-Conference.html.

Adam Jermyn, Chris Olah, and Tom Conerly. Circuits updates - january 2024. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/jan-update/index.html#attn-superposition.

Harish Kamath, Emmanuel Ameisen, Isaac Kauvar, Rodrigo Luger, Wes Gurnee, Adam Pearce, Sam Zimmerman, Joshua Batson, Thomas Conerly, Chris Olah, and Jack Lindsey. Tracing attention computation through feature interactions. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attention-qk/index.html.

Adam Karvonen, Benjamin Wright, Can Rager, Rico Angell, Jannik Brinkmann, Logan Smith, Claudio Mayrink Verdun, David Bau, and Samuel Marks. Measuring progress in dictionary learning for language model interpretability with board game models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/9736acf007760cc2b47948ae3cf06274-Abstract-Conference.html.

Leela Chess Zero Team. Transformer progress. https://lczero.org/blog/2024/02/transformer-progress/, February 2024.

Yujia Li, David H. Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *CoRR*, abs/2203.07814, 2022. doi: 10.48550/ARXIV.2203.07814. URL https://doi.org/10.48550/arXiv.2203.07814.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Jack Lindsey, Adly Templeton, Jonathan Marcus, Thomas Conerly, Joshua Batson, and Christopher Olah. Sparse crosscoders for cross-layer features and model diffing. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/crosscoders/index.html.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=I4e82CIDxv.

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. doi: 10.1073/pnas.2206625119. URL https://www.pnas.org/doi/10.1073/pnas.2206625119.

Szymon Milosz, Piotr Duch, and Szymon Grabowski. Beyond search-imitation: Prior-directed exploration for searchless chess. *CoRR*, abs/2608.27757, 2026. doi: 10.48550/ARXIV.2608.27757. URL https://doi.org/10.48550/arXiv.2608.27757.

Daniel Monroe and The Leela Chess Zero Team. Mastering chess with a transformer model. *CoRR*, abs/2409.12272, 2024. doi: 10.48550/ARXIV.2409.12272. URL https://doi.org/10.48550/arXiv.2409.12272.

Daniel Monroe, George Eilender, Philip Chalmers, Zhenwei Tang, and Ashton Anderson. Chessformer: A unified architecture for chess modeling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=2ltBRzEHyd.

Neel Nanda. Attribution patching: Activation patching at industrial scale. https://www.neelnanda.io/mechanistic-interpretability/attribution-patching, 2022.

Yoann Poupart. Contrastive sparse autoencoders for interpreting planning of chess-playing agents. *CoRR*, abs/2406.04028, 2024. doi: 10.48550/ARXIV.2406.04028. URL https://doi.org/10.48550/arXiv.2406.04028.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nat.*, 625(7995):468–475, 2024. doi: 10.1038/S41586-023-06924-6. URL https://doi.org/10.1038/s41586-023-06924-6.

Anian Ruoss, Grégoire Delétang, Sourabh Medapati, Jordi Grau-Moya, Kevin Li, Elliot Catt, John Reid, Cannada A. Lewis, Joel Veness, and Tim Genewein. Amortized planning with large-scale transformers: A case study on chess. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/78f0db30c39c850de728c769f42fc903-Abstract-Conference.html.

Lisa Schut, Nenad Tomašev, Thomas McGrath, Demis Hassabis, Ulrich Paquet, and Been Kim. Bridging the human–ai knowledge gap through concept discovery and transfer in AlphaZero. *Proceedings of the National Academy of Sciences*, 122(13):e2406675122, 2025. doi: 10.1073/pnas.2406675122. URL https://www.pnas.org/doi/10.1073/pnas.2406675122.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. Solving geometry problems: Combining text and diagram interpretation. In Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton (eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pp. 1466–1476. The Association for Computational Linguistics, 2015. doi: 10.18653/V1/D15-1171. URL https://doi.org/10.18653/v1/d15-1171.

Wentao Shu, Xuyang Ge, Guancheng Zhou, Junxuan Wang, Rui Lin, Zhaoxuan Song, Jiaxing Wu, Zhengfu He, and Xipeng Qiu. Bridging the attention gap: Complete replacement models for complete circuit tracing. *OpenMOSS Interpretability Research*, 2026. URL https://interp.open-moss.com/posts/complete-replacement.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nat.*, 529(7587):484–489, 2016. doi: 10.1038/NATURE16961. URL https://doi.org/10.1038/nature16961.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nat.*, 550(7676):354–359, 2017. doi: 10.1038/NATURE24270. URL https://doi.org/10.1038/nature24270.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Gerald Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Comput.*, 6(2):215–219, 1994. doi: 10.1162/NECO.1994.6.2.215. URL https://doi.org/10.1162/neco.1994.6.2.215.

Junxuan Wang, Xuyang Ge, Wentao Shu, Zhengfu He, and Xipeng Qiu. Dimensional collapse in transformer attention outputs: A challenge for sparse dictionary learning. In *Forty-third International Conference on Machine Learning, ICML 2026, Seoul, South Korea, July 6–11, 2026*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2026. URL https://openreview.net/forum?id=K9rWNmHhkD.

CONTENTS

A Related Work 16

B LC0 BT4 Details 16

B.1 Transformer Architecture 16

B.2 BT4 Strength Evaluation 16

C Dataset and Training 17

D Sparse Decomposition Training and Fidelity 18

E Graph Organization Metrics 19

F Chess-Structured Feature Taxonomy and Selective Rule-Based Verification 21

F.1 Design Rationale 21

F.2 Chess Primitives 21

F.3 Verification Primitives and Feature-Specific Tests 21

F.4 Feature Taxonomy 21

G Automatic Interpretation Details 23

G.1 Rule-Based Validation of Selected Interpretations 25

H Supplementary Details for the Case Study 25

H.1 Highest-Activation Samples for Features in Case Study 25

H.2 More Cases of Attribution Graphs 25

A Related Work

Chess as a Testbed for Reasoning. Chess is an ideal domain for studying reasoning abilities of a model due to its structured rules and multi-step logic. Computer chess systems have long achieved strong performance, from Deep Blue Campbell et al. (2002) to neural network AlphaZero Silver et al. (2018). Building on the MCTS-based self-play framework of AlphaZero, modern Transformer models—most notably the Leela Chess Zero (LC0) series Leela Chess Zero Team (2024)—have reached master-level performance in chess.

Sparse Decomposition and Replacement Layers. Since internal representations in neural networks are highly compressed in superposition Elhage et al. (2022); Jermyn et al. (2024), sparse decomposition methods like Sparse Autoencoders (SAEs) are employed to learn sparse, more interpretable features from superposition Bricken et al. (2023); Karvonen et al. (2024). Transcoders extend this paradigm by decomposing MLP computations and constructing replacement layers for MLP to extract interpretable computational features rather than just static activations Dunefsky et al. (2024); Ge et al. (2024); Ameisen et al. (2025). Furthermore, Low-Rank Sparse Attention (Lorsa) has been introduced to resolve the challenge of attention superposition He et al. (2026). While these methods have primarily been validated in large language models, we apply them here to construct unified replacement layers for both MLP and attention modules.

Mechanistic Interpretability in Chess. Prior interpretability studies have identified linearly separable concepts aligned with human knowledge in AlphaZero McGrath et al. (2022), as well as machine-specific concepts transferable to humans Schut et al. (2025). Subsequent work used Sparse Autoencoders (SAEs) to recover high-fidelity board-state properties such as piece locations Karvonen et al. (2024). Poupart (2024) further applied contrastive SAEs to LC0, analyzing features associated with optimal and suboptimal trajectories. Cross-layer Transcoders in Chessformer Ameisen et al. (2025); Monroe et al. (2026) revealed tactical concepts in a domain-specific transformer. However, these approaches remain primarily static and local, with only specific attention heads shown to support look-ahead reasoning Jenner et al. (2024). Similar techniques in Othello-GPT attempted patch-free circuit discovery via SAE He et al. (2024). Our work extends these efforts by moving from static representation analysis to uncovering dynamic, end-to-end computations.

B LC0 BT4 Details

B.1 Transformer Architecture

LC0 is a family of transformer-based chess models trained with the Monte Carlo Tree Search (MCTS) self-play reinforcement learning paradigm introduced by AlphaZero Silver et al. (2017).

In this work, we study BT4[1], a transformer chess model that achieves master-level performance using only its policy network Leela Chess Zero Team (2024). Our analysis focuses on this policy network; additional implementation details are shown in Figure 5.

BT4 consists of 15 transformer layers with residual connections and hidden size $d_{\text{model}} = 1024$. Each multi-head self-attention (MHSA) layer contains 32 attention heads. The input sequence has length 64, corresponding to the 64 squares of the 8×8 chessboard.

B.2 BT4 Strength Evaluation

To evaluate the playing strength of BT4 without search, we test it on Lichess puzzles of different tactical depths. Specifically, we construct five subsets corresponding to mate-in-1 through mate-in-5 puzzles. For each subset, we sample 20,000 positions, except for mate-in-5, where only 5,000 positions are used due to the smaller number of available puzzles. The results are shown in Table 4. Using a single forward pass of the policy network without search, BT4 achieved a 5–0–0 record against our human player rated above 2100+ on the Lichess platform.

[1]Specifically, we use `BT4-1024x15x32h-swa-6147500`, a stable release from the LC0 project. The weights are publicly available at `https://lczero.org/play/networks/bestnets/`.

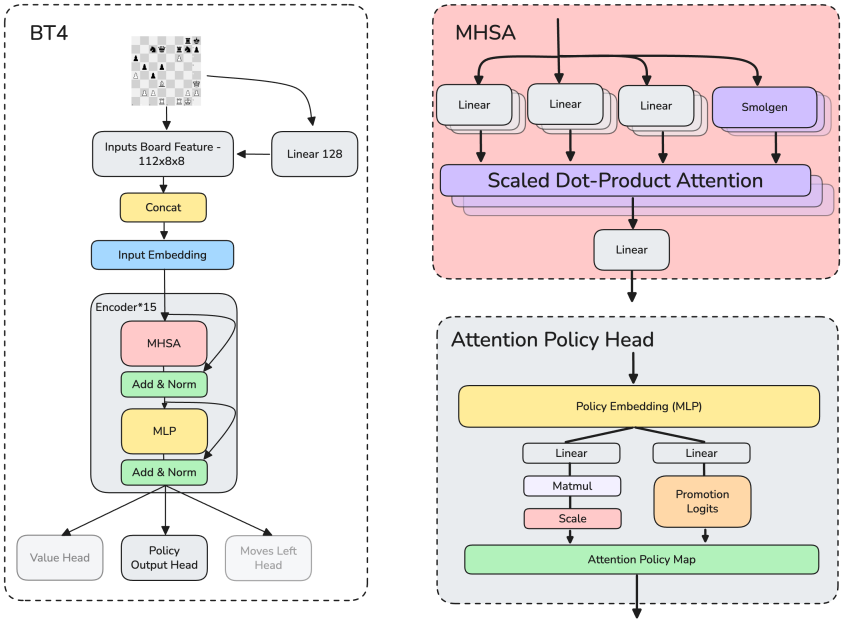


Figure 5: Overview of the BT4 architecture, featuring Smolgen-enhanced MHSA for modeling chessboard positional relations and an attention-based policy output head.

Puzzle set	Mate-in-1	Mate-in-2	Mate-in-3	Mate-in-4	Mate-in-5	All
Accuracy	0.996	0.970	0.890	0.776	0.707	0.932

Table 4: BT4 puzzle accuracy without search on Lichess mate puzzles of varying tactical depth. We sample 20,000 positions for each subset from mate-in-1 to mate-in-4, and 5,000 positions for mate-in-5 due to limited data availability.

C Dataset and Training

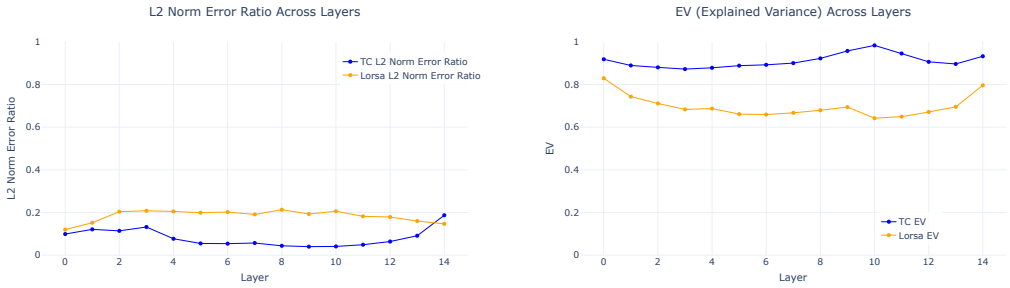
We use chess positions from the `lichess_standard_rated`[2]dataset to generate activations and train both Transcoders and Lorsas.

While each position is represented as a sequence of 64 tokens corresponding to 64 squares, we sample 800M tokens for training and 100M for analysis of the Transcoders and Lorsas. Additional sparse-decomposition evaluations are summarized at the beginning of Appendix D.

To approximate the original MHSA computation, we match the attention architecture described in Appendix B.1, including bidirectional attention and the Smolgen module. We further optimize the initialization scheme by initializing W_O within the output subspace of the original attention heads Wang et al. (2026). The original W_Q, W_K, Smolgen parameters, and attention scaling factors are also used to initialize Lorsas.

Training and graph-construction settings. Transcoders use batches of 32,768 scalar activation vectors, while Lorsas are trained with a learning rate of 2×10^{-4}. For both methods, the auxiliary k-loss coefficient is $1/32 = 0.03125$, with $k_{\text{aux}} = 512$ for Transcoders and $k_{\text{aux}} = 2048$ for Lorsas. For attribution-graph construction, we use a fixed budget of at most 4,096 candidate feature nodes before tracing; this is an upper bound on the candidate pool rather than the final number of nodes in each graph. We set the edge-retention threshold to 0.57, retaining the highest-influence edges whose cumulative contribution reaches 57% of the total edge influence, and iteratively remove feature nodes that lose all incoming or outgoing edges.

[2]The `lichess_standard_rated` dataset is publicly available at `https://database.lichess.org/` and contains over 7.4 billion rated standard chess games played on `lichess.org`.



(a) Layer-wise L2 norm error ratio of sparse decomposition modules.

(b) Layer-wise explained variance (EV) of sparse decomposition modules.

Figure 6: Fidelity of sparse decomposition modules. We report the L2 norm error ratio (left) and the explained variance EV (right) of Transcoder and Lorsa modules across transformer layers on a randomly sampled dataset.

D Sparse Decomposition Training and Fidelity

Validation framework. We evaluate three levels of validity. **Replacement fidelity** measures reconstruction (L2 error and EV) and behavioral preservation (KL divergence, top-1 agreement, and puzzle accuracy). **Semantic validation** reports rule-based precision and recall for operationalizable interpretations (Appendices F and G.2). **Attribution faithfulness** uses zero patching, attribution–effect correlation, edge effects, and graph-level replacement and completeness Ameisen et al. (2025).

Sparse Decomposition Details. In the main experiments, we configure the Transcoder and Lorsa modules with $k = 30$ and an expansion factor of 16. This results in $1024 \times 16 = 16{,}384$ Transcoder and Lorsa features per layer, with 30 activated per square. Lorsa and Transcoder features are both intrinsically 1-dimensional and can be represented uniformly via encoder and decoder vectors, so we match the number of them. To decrease the number of dead features, we apply an auxiliary k loss Gao et al. (2025) for both Transcoders and Lorsas.

Replacement fidelity: L2 norm error ratio and explained variance. In Figure 6, we examine the **L2 Norm Error Ratio** and **Explained Variance (EV)** of the Transcoders and Lorsas trained on a randomly sampled dataset.

$$\text{L2 Norm Error Ratio} = \frac{\mathbb{E}_t\left[\|\hat{\mathbf{x}}_t - \mathbf{x}_t\|_2\right]}{\mathbb{E}_t\left[\|\mathbf{x}_t\|_2\right]}, \tag{9}$$

$$\text{Explained Variance} = 1 - \frac{\mathbb{E}_t\left[\|\hat{\mathbf{x}}_t - \mathbf{x}_t\|_2^2\right]}{\mathbb{E}_t\left[\|\mathbf{x}_t - \bar{\mathbf{x}}\|_2^2\right]}. \tag{10}$$

As shown in Figure 6, these metrics quantify how accurately sparse decomposition modules reconstruct the original module output. We observe that all modules have an L2 norm error ratio below 0.3 and explained variance above 0.6. While Lorsas show lower explained variance in the middle layers—particularly for attention outputs—this is consistent with the increased architectural complexity and nonlinearity of information transfer at intermediate depths.

Replacement fidelity: KL divergence, top-1 agreement, and puzzle accuracy. We replace one module at a time and compare each Transcoder or Lorsa with mean ablation using logit KL divergence, top-1 agreement, and accuracy on 1,000 puzzles (BT4: 0.932). Across almost all layers, sparse replacements yield lower KL divergence and higher agreement and puzzle accuracy than mean ablation (Figure 7).

Attribution faithfulness: zero patching and attribution–effect correlation. We test whether high-attribution features affect the target move logit. For each graph, we zero-patch the 64 features

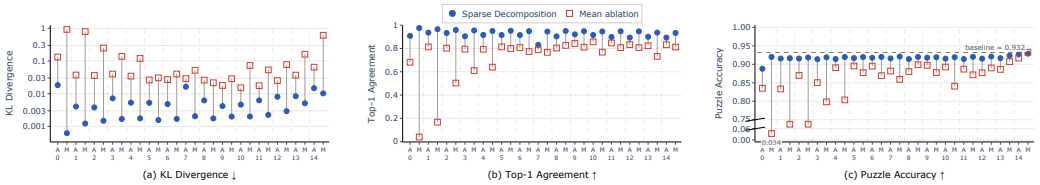
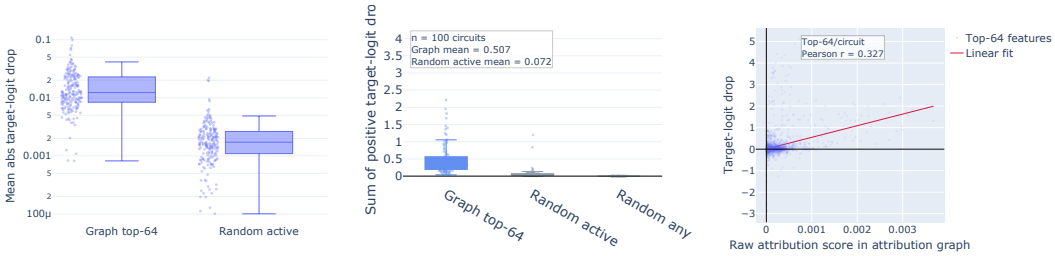


Figure 7: The KL divergence, top-1 agreement, and puzzle accuracy of the sparse decomposition modules across all layers. The KL divergence and top-1 agreement are calculated on random positions, and the puzzle accuracy is calculated on 1,000 puzzles.



(a) Absolute target-logit changes induced by zero patching of top attribution-graph features and randomly selected active features.

(b) Total positive target-logit drops induced by zero patching attribution-graph features and randomly selected active features.

(c) Linear fit between all-path attribution scores and zero patching effects on the target-move logit.

Figure 8: Causal validation of attribution-graph features. (a) Zero patching of the top 64 features selected from an attribution graph produces a larger absolute target-logit change than zero patching of randomly selected active features. (b) Attribution-graph features also produce larger aggregate positive drops in the target-move logit. (c) Among the top 64 attribution-graph features, summed all-path attribution scores are positively correlated with zero-patching effects on the target logit.

with the highest all-path attribution scores (Eq. 5) and compare them with randomly selected active features. The selected features produce larger absolute logit changes and larger summed positive drops (Figure 8a,b). Attribution scores also correlate positively with patching effects (Figure 8c), although the agreement is not exact.

Attribution faithfulness: edge effects and graph completeness. In Section 5, we additionally evaluate causal effects at the edge level. Feature pairs connected by an attribution-graph edge show a larger mean intervention effect than generic within-circuit or cross-circuit pairs, and edge weights correlate with intervention effects.

We also compute graph-level completeness scores following Ameisen et al. (2025). Let $\mathcal{N}$ denote the selected non-error nodes in an end-to-end attribution graph, and let $\mathcal{R}$ denote the error nodes. Let $B_{\text{logit},j}$ be the absolute contribution of node j to the target logit, and let $\widehat{A}_{ji}$ be the normalized indirect attribution from node i to node j. We define the completeness score as

$$S_c = \frac{\sum_{j\in\mathcal{N}} \left(1 - \sum_{i\in\mathcal{R}} \widehat{A}_{ji}\right) B_{\text{logit},j}}{\sum_{i\in\mathcal{N}} B_{\text{logit},i}}. \tag{11}$$

Completeness measures how much indirect influence is carried by features or embeddings rather than error nodes. It increases when more features are retained and when the sparse modules use smaller k (Figure 9).

E Graph Organization Metrics

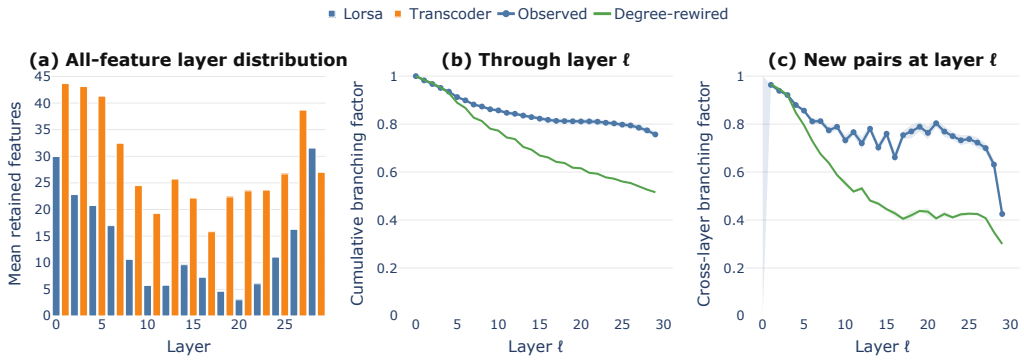


Figure 10: Layer-wise attribution-graph branching. (a) Mean retained Lorsa and Transcoder features per layer. (b) Cumulative branching factor through layer ℓ. (c) Branching factor of feature pairs newly introduced at layer ℓ. Computation remains multi-branched through most of the network, with branches progressively converging near the output.

Feature Overlap For the top two candidate moves, let $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1, \mathcal{A}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2, \mathcal{A}_2)$ denote their attribution graphs. We measure feature reuse using Jaccard overlap over feature nodes. Each node is identified by (feature ID, board square, layer), and two nodes match only if all three components are identical:

$$O_{\mathcal{V}} = \frac{|\mathcal{V}_1 \cap \mathcal{V}_2|}{|\mathcal{V}_1 \cup \mathcal{V}_2|}. \tag{12}$$

In Section 5.1, we measure feature overlap between the attribution graphs of the top two moves across 1,000 positions with different source squares and 1,000 positions with the same source square. Moves sharing a source square also share the same query position in the bilinear policy network. Although overlap is generally low, same-source move pairs exhibit substantially higher overlap, suggesting reuse of shared-source representations such as piece-detection features.

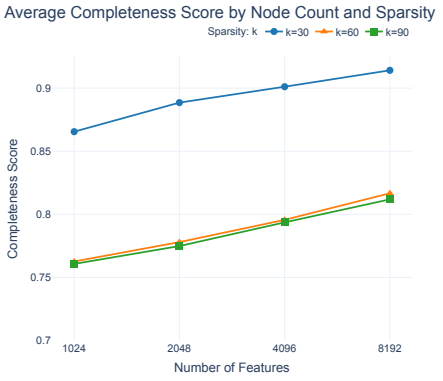


Figure 9: Completeness scores of end-to-end attribution graphs. Higher scores indicate that influential feature nodes receive more indirect influence from features or embeddings rather than error nodes.

Attribution-graph branching factor To measure branching within a graph without over-weighting low-attribution side branches, we weight each feature u by its absolute all-path attribution to the target logit, $w_u = |A_{u \to t}|$, and define

$$B(\mathcal{G}) = \frac{\sum_{u<v} w_u w_v \mathbf{1}[u \not\rightsquigarrow v \wedge v \not\rightsquigarrow u]}{\sum_{u<v} w_u w_v}. \tag{13}$$

This metric measures the attribution-weighted fraction of feature pairs on mutually unordered branches. The **Degree-rewired baseline** randomly rewires edge endpoints while preserving nodes, attribution weights, edge count, DAG ordering, and each node's in- and out-degree, but not layer-pair edge counts. Figure 10 shows that branching remains above this baseline through most layers but decreases near the output, indicating progressive convergence toward the final decision.

Feature-to-feature effect For causal influence, given an upstream feature f_i and a downstream feature f_j, we define the feature-to-feature effect as

$$\mathrm{Eff}(f_i, f_j) = \frac{a_j(h) - a_j(\widetilde{h}^{(i)})}{a_j(h)}, \tag{14}$$

where $a_j(h)$ is the original activation of feature j, and $a_j(\widetilde{h}^{(i)})$ is its activation after patching feature f_i. For an attribution graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, the average intra-circuit and cross-circuit effects are

$$\mathrm{IntraEff}(\mathcal{G}) = \frac{\sum_{f_i, f_j \in \mathcal{V},\, i \neq j} |\mathrm{Eff}(f_i, f_j)|}{|\mathcal{V}|(|\mathcal{V}| - 1)}, \qquad \mathrm{CrossEff}_{12} = \frac{\sum_{f_i \in \mathcal{V}_1,\, f_j \in \mathcal{V}_2} |\mathrm{Eff}(f_i, f_j)|}{|\mathcal{V}_1||\mathcal{V}_2|}. \quad (15)$$

F Chess-Structured Feature Taxonomy and Selective Rule-Based Verification

Table 1 reports some verification results including precision and recall for selected features in the case studies. Below, we describe the descriptive taxonomy used to organize interpretations, the chess primitives available to the verifier, and the statistical tests used when an interpretation can be reliably operationalized. These tests provide quantitative evidence for the tested interpretations, but do not constitute exhaustive validation of the taxonomy or of all automatically generated interpretations.

F.1 Design Rationale

The taxonomy organizes interpretations, whereas verification rules test feature-specific claims. The verifier abstains when no reliable rule can be constructed; high precision with low recall may reflect feature splitting Chanin et al. (2025).

F.2 Chess Primitives

Let $\mathcal{B}$ denote the set of chess positions and $\mathcal{S}$ the set of 64 board squares. For each position $b \in \mathcal{B}$, the model produces a value prediction $V_\theta(b)$ and policy logits $\ell_\theta(b, m)$ over legal moves $m \in \mathcal{M}_b$. We write $m^*_\theta(b)$ and $\mathrm{TopK}_\theta(b; k)$ for the model's top move and top-k moves, respectively.

For a position b, let $\mathcal{P}_b$ be the set of pieces on the board. Each piece $p \in \mathcal{P}_b$ has a type $\mathrm{Type}(p) \in \{K, Q, R, B, N, P\}$, side $\mathrm{Side}(p) \in \{\mathrm{Own}, \mathrm{Opp}\}$, and location $\mathrm{Loc}_b(p) \in \mathcal{S}$. Here Own denotes the current player's pieces and Opp denotes the opponent's pieces.

We use standard chessboard predicates such as $\mathrm{SameFile}$, $\mathrm{SameRank}$, $\mathrm{SameDiag}$, $\mathrm{SameColor}$, and $\mathrm{Adjacent}$ to describe geometric relations between squares. For sliding pieces, $\mathrm{ClearPath}_b(s_1, s_2)$ denotes that no occupied square lies between s_1 and s_2 along a file, rank, or diagonal.

We write $\mathrm{Reach}_b(p, s)$ to indicate that piece p can reach square s according to its chess piece movement rules. Similarly, $\mathrm{Attack}_b(p, s)$ denotes that p attacks s, and $\mathrm{Reach}^{\leq n}_b(p, s)$ denotes reachability within at most n abstract movement steps.

Finally, we use derived predicates such as $\mathrm{Defended}_b(s, \sigma)$ for squares defended by side σ, $\mathrm{Capture}_b(p, q)$ for an attack from p onto an opposing piece q, and ray-based relations such as rook-like, bishop-like, and queen-like reachability. These primitives provide building blocks for the feature-specific tests reported in this paper; they do not define the taxonomy categories themselves.

F.3 Verification Primitives and Feature-Specific Tests

For each selected feature, the verifier translates the proposed interpretation into an executable chess test and compares its predictions with the observed activations using precision and recall. For Lorsa features, the test may also use z-pattern evidence. The verifier abstains when the interpretation cannot be tested reliably.

F.4 Feature Taxonomy

We organize features into a chess-structured taxonomy to capture the interpretable information learned by the sparse decomposition modules and their evolution across layers. We use ten fine-grained categories: **detection** ([Det]), **source square** ([Src]), **target square** ([Tgt]), **value** ([Val]), **capture** ([Cap]), **protection** ([Pro]), **movement geometry** ([Mov]), **spatial relations** ([Spa]), **higher-level tactical motifs** ([Tac]), and **register-like features** ([Reg]). The taxonomy provides a compact basis for annotation and layer-wise analysis. Its categories are descriptive and do not prescribe a unique verification rule.

Detection Features [Det]. Detection features encode piece identity, activating on squares occupied by pieces of a recurring side and type. For Lorsa features, a detection interpretation describes the activation location unless the observed z-patterns also support a directional relation.

Source and Target Square Features [Src] / [Tgt]. Source and target features indicate that the activated square corresponds to the source or target square of high-probability policy moves across activating examples. They typically appear in later layers and are associated with the construction of move outputs. The labels summarize repeated policy-move evidence rather than requiring exact agreement with a single move-ranking rule on every position.

Value Features [Val]. Value features exhibit activation patterns associated with the model's predicted win–draw–loss or scalar value output. Their interpretation is primarily position-level and need not correspond to a fixed local square relation.

Capture Features [Cap]. Capture features encode recurring attacker–target relations between opposing pieces. Depending on the feature, the activation can occur on either participant or on a related destination square; for Lorsa features, z-patterns can provide additional evidence about the direction of information flow.

Protection Features [Pro]. Protection features encode recurring same-side defense relations, such as one piece protecting another or controlling a relevant square. For directional Lorsa features, z-patterns may distinguish the protecting piece from the protected piece or square.

Piece Movement Features [Mov]. Piece movement features encode recurring geometry induced by chess movement rules, including legal destinations, attack fields, and one- or multi-step reachability. The relevant piece, destination, blocker constraints, and directional ordering are specified by the individual interpretation rather than by the category label alone.

Spatial Relation Features [Spa]. Spatial features encode board-level geometric relations, such as same file, same rank, same diagonal, same color, adjacency, or local neighborhoods. Such relations can be defined relative to an anchor piece or square, or between the activation and high-z source squares of a Lorsa feature.

Complex Tactic Features [Tac]. Complex tactic features encode higher-order chess relations composed of multiple lower-level relations. This category covers candidate interpretations involving exchange states, multi-move reachability, blocker constraints, pins, skewers, and mating threats. Because these hypotheses are often context-dependent, we report a rule-based score only when the proposed relation can be implemented without introducing unsupported assumptions.

Register Features [Reg]. There exists a class of features that often activate strongly on empty squares near the board edge. They usually do not correspond to specific local chess concepts; we hypothesize that they encode global information, analogous to register features in vision models Darcet et al. (2024). For Lorsa, register-like behavior may also involve stable source–target routing. These features are tracked during interpretation and layer-wise analysis but are not treated as evidence for a specific chess concept without an additional, independently testable hypothesis.

G AUTOMATIC INTERPRETATION DETAILS

G.1 AGENTIC INTERPRETATION-TO-VALIDATION WORKFLOW

We use an agentic pipeline with a deterministic rule engine:

1. **Interpretation.** The agent uses the taxonomy and few-shot examples to infer a chess concept from all available activation, policy, value, z-pattern, and circuit evidence.
2. **Test construction.** It converts the interpretation into executable chess tests and retains the candidates that best match the board-level evidence.
3. **Refinement.** Precision, recall, and false-positive and false-negative cases are returned to the agent to revise the interpretation and test over several turns. We consider a feature highly interpretable only when both precision and recall exceed 0.60.
4. **Final validation.** The final interpretations are evaluated on independently sampled chess positions. Aggregate results are reported in Table 5, and layer-wise results are shown in Figure 11. Some features have lower recall because their interpretations remain too coarse; for example, a feature may encode a pawn threatening the king, while the proposed interpretation identifies only the pawn.

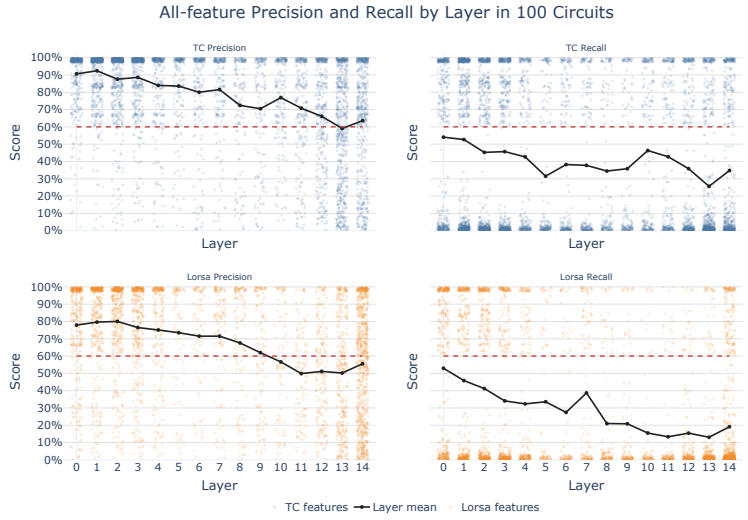


Figure 11: Precision and recall by layer for Transcoder (TC) and Lorsa features across 100 circuits. Points denote individual features, black curves show layer means, and the dashed line marks the 0.60 validation threshold.

Prompt

Inspect all available top-activation boards and express ownership relative to the side to move. Record activated squares, their occupants, policy moves, and relevant movement, attack, defense, check, and value evidence. For Lorsa, inspect the strongest source–target z-patterns. Use circuit inputs and outputs only as supporting context, and require the proposed relation to recur across activation boards.
Compare simple, falsifiable hypotheses spanning detection, policy source or target, movement, tactics, spatial structure, and value. Unless exact occupancy explains at least 90% of activated squares, record supporting and contradicting examples for several candidates. Return a concise taxonomy label, interpretation, confidence, and evidence-grounded rationale, ensuring that the executable test matches the stated semantics. Use top samples only for development: inspect false positives and false negatives, revise the hypothesis over multiple turns, freeze the final test before the independent scan, and abstain unless both precision and recall exceed the validation threshold.

Table 5: Rule quality over three agent refinement turns. "Both" denotes features with precision and recall above 0.60.

Feature type	Agent turn	Precision > 0.60		Recall > 0.60		Both > 0.60	Total
		Count	Share	Count	Share		
TC	1	3,129	60.44%	144	2.78%	–	5,177
	2	4,240	81.90%	1,176	22.71%	–	
	3	4,063	78.48%	2,133	41.20%	2,261 (43.67%)	
Lorsa	1	1,515	40.78%	99	2.66%	–	3,715
	2	2,545	68.51%	596	16.04%	–	
	3	2,479	66.73%	1,106	29.77%	1,158 (31.17%)	

Automatic Blinded Assessment Score. Following the automatic blinded assessment protocol Anthropic Interpretability Team (2024), we use the OpenAI Codex CLI to evaluate Transcoder features, Lorsa features, and MLP neurons in terms of activation consistency and complexity, as shown in Table 2. An excerpt from our evaluation prompt is shown below.

Prompt

You are given exported top-activation evidence for a chess model component, which may be a Transcoder feature, a Lorsa feature, or an MLP neuron. Inspect each top-activation sample directly rather than relying only on the proposed interpretation or aggregate statistics. For each board, identify the activated squares and their occupants, convert pieces to Own/Opponent using the side to move, and check whether the same chess relation is present across samples. Use movement reachability, pawn advances or controls, captures, protection, lines, tactical coordination, value signals, and stable spatial structure as evidence. Your task is to infer the simplest high-fidelity chess pattern represented by the component and assign two blinded assessment scores: **activation consistency** and **complexity**.

Activation consistency. Rate how consistently the activating examples match a single interpretable pattern:

- **5**: Clear pattern with no deviating examples.
- **4**: Clear pattern with one or two deviating examples.
- **3**: Clear overall pattern, but quite a few examples do not fit that pattern.
- **2**: Broad consistent theme, but lacking clear structure.
- **1**: No discernible pattern.

Complexity. Rate the semantic complexity of the inferred chess concept:

- **5**: Rich feature firing on diverse contexts with an interesting unifying theme, e.g., 'check-forced queen loss".
- **4**: Feature relating to high-level semantic structure, e.g., 'capture opponent's queen with check" or 'multi-piece attacking squares".
- **3**: Moderate-complexity concept, such as a move pattern or square coverage, e.g., 'own queen coverage".
- **2**: Single piece or token feature in a specific position, e.g., 'the front pawn".
- **1**: Single token feature, e.g., 'own queen".

Return a concise JSON-style assessment containing: the inferred explanation, the activation consistency score, the complexity score, the number of usable samples, the number of clear fits, partial fits, and deviations, and a short evidence summary naming representative squares, pieces, and any Lorsa high-z relations used in the judgment.

Table 6: Representative rule-based evaluations selected for semantic diversity. Colors indicate Det, Src/Tgt, Val, Cap, Pro, Tac, Spa, and Mov. **Bold IDs** denote features used in the main text.

Feature ID	Interpretation	Verification Rule	Precision	Recall
Tc.0.5593	[Det]Opp.Knight	Act. @ Opp.Knight	100%	100%
Tc.0.12663	[Det]OwnPawn	Act. @ OwnPawn	100%	100%
Lorsa.0.13740	[Det]Opp.Queen ← OwnQueen	Act. @ Opp.Queen ← OwnQueen	98.6%	99.6%
Tc.2.12759	[Det]Opp.Knight	Act. @ Opp.Knight	100%	100%
Tc.13.15426	[Det]OwnKnight	Act. @ OwnKnight	100%	99.7%
Lorsa.1.8534	[Det]Opp.Bishop	Act. @ Opp.Bishop	100%	99.7%
Tc.7.3804	[Det]Opp.King	Act. @ Opp.King	100%	99.6%
Tc.12.15674	[Det]OwnQueen	Act. @ OwnQueen	100%	96.6%
Tc.14.13630	[Src]OwnPawn policy source	Act. @ Top-policy source occupied by OwnPawn	43.1%	93.1%
Tc.14.1034	[Src]OwnKnight policy source	Act. @ Top-policy source occupied by OwnKnight	55.4%	56.5%
Tc.13.9961	[Src]Policy source square	Act. @ Top-policy source	92.2%	34.8%
Tc.14.14940	[Tgt]OwnKnight policy target	Act. @ Legal OwnKnight destination selected by the policy	62.6%	62.0%
Tc.14.7938	[Tgt]OwnBishop policy target	Act. @ Top-policy target reached by OwnBishop	50.4%	66.9%
Tc.13.3327	[Tgt]OwnRook capture target	Act. @ Opp.Piece targeted by the top OwnRook move	57.6%	55.4%
Tc.14.1214	[Tgt]Policy target square	Act. @ Top-policy target	95.9%	37.0%
Tc.14.4046	[Val]Low Value	Act. @ All squares in positions with predicted value < -0.8	83.1%	77.7%
Tc.13.6138	[Val]High win probability	Act. @ Positions with side-to-move win probability > 0.6	64.4%	78.5%
Tc.4.5191	[Cap]Opp.Piece Captured by OwnPawn	Act. @ OwnPawn x Opp.Piece	97.0%	86.6%
Tc.0.8584	[Cap]Opp.Piece attacked by OwnPawn	Act. @ Opp.Piece attacked by OwnPawn	100%	95.8%
Tc.3.14478	[Cap]Opp.Piece attacked by OwnPawn	Act. @ Opp.Piece attacked by OwnPawn	100%	79.7%
Tc.1.13650	[Cap]Opp.Knight attacked by OwnBishop	Act. @ Opp.Knight attacked by OwnBishop	98.7%	95.5%
Lorsa.1.5572	[Pro]Opp.Pawn defensive coverage	Act. @ Square attacked/defended by Opp.Pawn	100%	93.4%
Lorsa.3.3742	[Pro]Opp.Knight protected by Opp.Piece	Act. @ Opp.Knight protected by Opp.Piece	86.1%	99.5%
Lorsa.7.10921	[Pro]OwnKnight protected by OwnPiece	Act. @ OwnKnight protected by OwnPiece	85.5%	91.8%
Tc.0.3972	[Pro]Opp.Bishop protected by Opp.Pawn	Act. @ Opp.Bishop defended by Opp.Pawn	100%	88.2%
Lorsa.3.1648	[Pro]Contested, defended OwnPawn	Act. @ OwnPawn defended by OwnPiece and attacked by Opp.Pawn	87.7%	86.6%
Tc.0.3225	[Pro]OwnPawn supported on an OwnRook file	Act. @ OwnPawn aligned with OwnRook on the same file	84.8%	94.3%
Tc.1.8537	[Pro]Opp.Knight coordinated with Opp.Queen	Act. @ Opp.Knight on the blocked diagonal of Opp.Queen	100%	88.7%
Tc.0.13607	[Pro]Opp.Knight protected by Opp.Bishop	Act. @ Opp.Knight defended by Opp.Bishop	100%	85.0%
Tc.0.4336	[Tac]OwnKnight checking destination	Act. @ Legal OwnKnight destination that gives check	94.4%	77.9%
Tc.1.2362	[Tac]Opp.Rook–pawn file coordination	Act. @ Opp.Rook coordinated with a nearby same-file Opp.Pawn	83.1%	66.0%
Tc.2.15875	[Tac]Local Opp.Rook–pawn coordination	Act. @ Opp.Rook adjacent to a coordinated Opp.Pawn	61.3%	63.6%
Tc.13.7425	[Tac]OwnQueen/Rook checking target	Act. @ Checking target of OwnQueen or OwnRook	64.5%	75.2%
Tc.3.13116	[Tac]Check-forced rook/queen loss	Act. @ OwnRook/Queen exposed by a checking skewer	71.5%	11.6%
Tc.1.6695	[Tac]Opp.Queen mate-in-one threat	Act. @ Mate-in-one target of Opp.Queen	67.0%	10.0%
Lorsa.1.15014	[Tac]Queen-check target near Opp.King	Act. @ OwnQueen checking target adjacent to Opp.King	83.5%	33.1%
Lorsa.8.16275	[Tac]Semi-open-file heavy-piece check threat	Act. @ Opp.Rook/Queen checking target within two moves and ≤ 2 blockers	98.3%	39.5%
Lorsa.2.3155	[Tac]Pawn break opening a heavy-piece threat	Act. @ OwnPawn break opening OwnRook/Queen attack on Opp.Queen/King	62.4%	18.6%
Lorsa.13.1520	[Tac]Three-step look-ahead alignment	Act. @ First-step endpoint aligned with third-step endpoint	90.7%	45.0%
Lorsa.0.9370	[Mov]Opp.Pawn forward movement	Act. @ One square ahead of Opp.Pawn	100%	97.0%
Lorsa.2.7624	[Mov]OwnBishop legal destinations	Act. @ Legal destination of OwnBishop	78.2%	99.4%
Lorsa.7.13586	[Mov]Opp.Knight movement	Act. @ Knight-reach squares ← Opp.Knight	99.9%	81.6%
Lorsa.7.12765	[Mov]OwnKnight legal destinations	Act. @ Legal destination of OwnKnight	81.5%	82.5%
Lorsa.2.14055	[Mov]Opp.Bishop legal destinations	Act. @ Legal destination of Opp.Bishop	81.0%	88.5%
Tc.0.10555	[Mov]OwnKing movement	Act. @ Legal destination of OwnKing	99.9%	80.0%
Tc.0.845	[Mov]OwnRook back-rank destination on an OwnQueen diagonal	Act. @ Legal OwnRook destination on Own back rank and OwnQueen diagonal	90.0%	93.5%
Lorsa.8.10098	[Mov]Opp.Queen blocked diagonal	Act. @ Blocked diagonal ray from Opp.Queen	99.2%	77.8%
Lorsa.7.14502	[Mov]OwnBishop movement	Act. @ Bishop-reach squares ← OwnBishop	83.8%	67.5%
Lorsa.8.9586	[Mov]OwnBishop movement	Act. @ Bishop-reach squares ← OwnBishop	98.4%	76.8%

G.2 Rule-Based Validation of Selected Interpretations

We found that features in the sparse decomposition modules exhibit a wide variety of semantic types, with characteristic distribution patterns across layers. Leveraging the structured nature of chess, we use a descriptive taxonomy to organize proposed feature interpretations. Based on human prior knowledge, we use ten fine-grained categories: [Det], [Src], [Tgt], [Val], [Cap], [Pro], [Mov], [Spa], [Tac], and [Reg]. Appendix F describes these categories and the chess predicates available to the verifier. When a proposed interpretation can be reliably operationalized, the agentic workflow stores an executable rule and evaluates it with an independent random-position scan; otherwise, it abstains. Table 1 reports representative interpretations that passed this workflow, together with their validation precision and recall.

H Supplementary Details for the Case Study

H.1 Highest-Activation Samples for Features in Case Study

In this subsection, we present two board position samples with the highest activations in the dataset for representative features shown in the case studies, Figure 12 for **Internal Look-Ahead**, Figure 13 for **Multiple Strategic Cues**, and Figure 14 for **Overly Conservative Decision**.

H.2 More Cases of Attribution Graphs

These additional cases illustrate how attribution graphs expose both successful multi-step reasoning and failure modes in BT4's policy network.

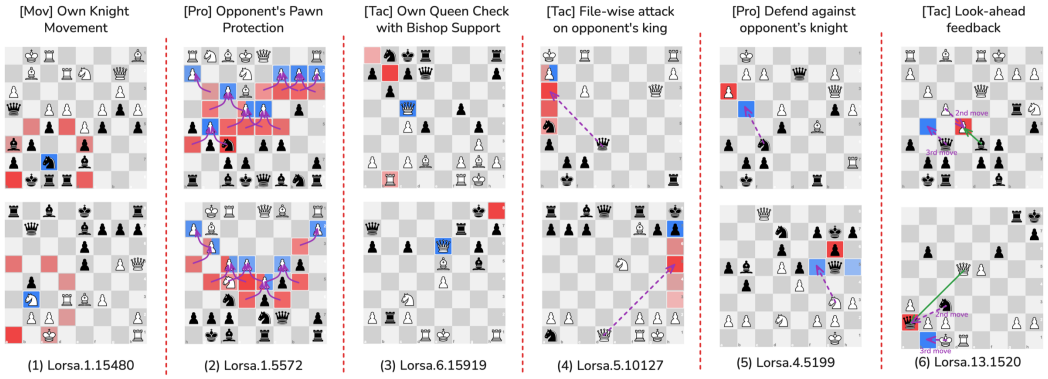


Figure 12: Samples with the highest activations of representative features in the internal look-ahead case in Section 4.1.

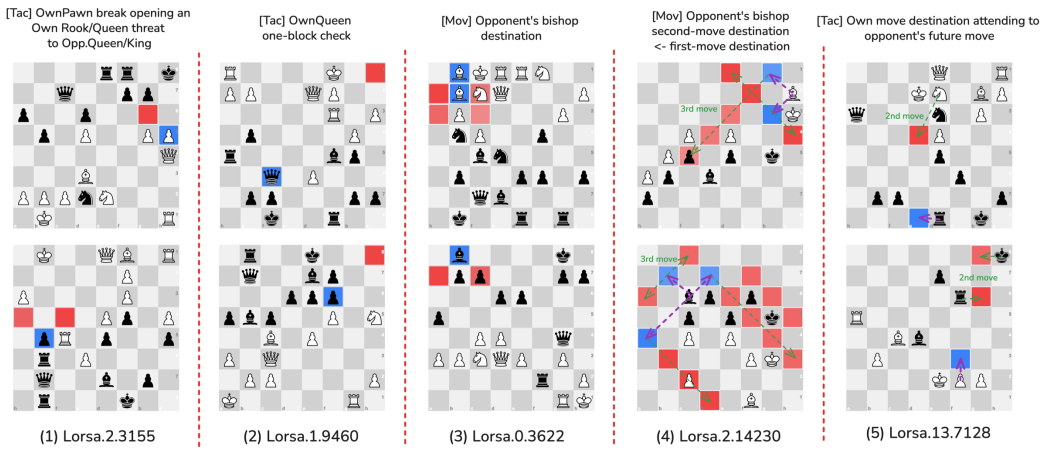


Figure 13: Samples with the highest activations of features mentioned in Section 4.2.

Internal look-ahead Figure 15 presents the end-to-end attribution graph for the mate-in-two case discussed in Section 4. The graph highlights the main look-ahead pathway: features encoding the opponent's **h7** pawn coverage on the **g6** target square co-occur with rook-attack features at **h4**, which correspond to the subsequent checkmate threat. This future-threat information is then routed back to the **g6** target representation through features such as **Lorsa.14.1078**. An enlarged version of the graph is shown in Figure 16. Our attribution analysis therefore complements prior evidence of learned look-ahead Jenner et al. (2024) with a feature-level view of the corresponding attribution pathway.

Multiple Strategic Cues Figure 17 presents the end-to-end attribution graph for this sacrifice case. The graph highlights a multi-factor pathway supporting **Ne5**: the move is connected to opening the d-file for the rook on **d1**, enabling future mating pressure via **Qf7+**, and preparing the subsequent development of **Bg2**. The graph also contains features corresponding to relevant defensive and opponent-response considerations, including the weakened diagonal pressure from **Bb7** and the plausible reply **Qe7**.

Overly Conservative Decisions Figure 18 provides the full attribution-graph view of the conservative failure case introduced in Section 4.3. Although **Qxh7+** is the correct mating move, BT4 assigns higher probability to **fxg7+**, a safer-looking continuation. The attribution graphs show that this preference is partly driven by the **g2** pawn feature **Tc.0.12663@g2**, which supports **fxg7+** by representing the pawn as protected and the king-side structure as stable. The same defensive signal

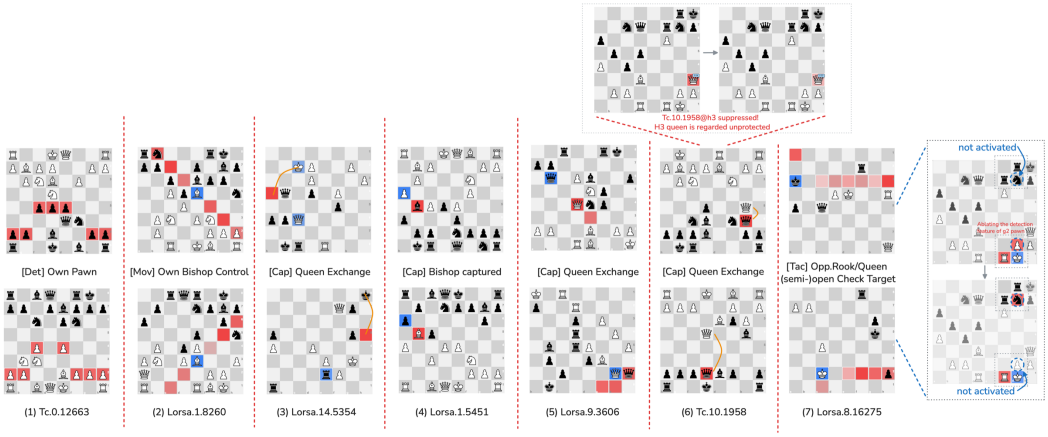


Figure 14: Samples with the highest activations of features mentioned in Section 4.3.

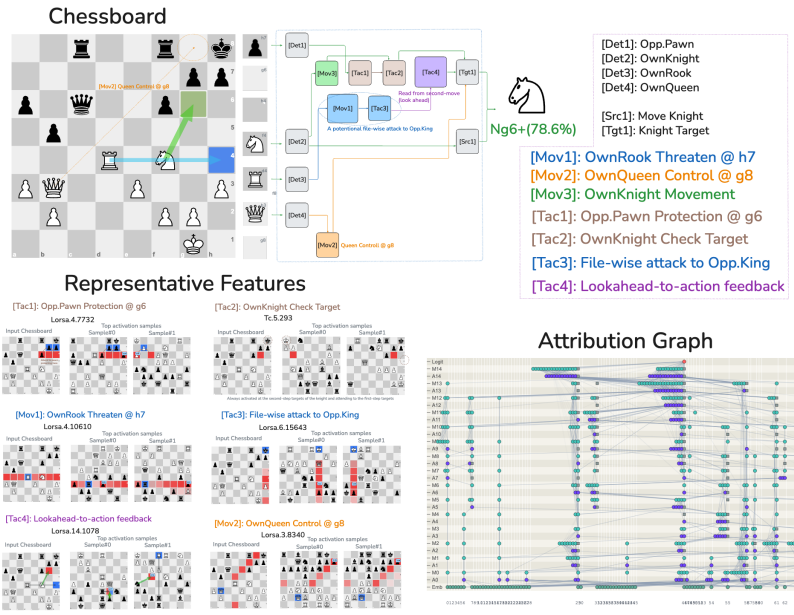


Figure 15: The end-to-end attribution graph for look-ahead reasoning in a mate-in-two scenario. The graph highlights a pathway that connects the immediate sacrifice **Ng6+** with the anticipated checkmate threat from the rook on **h4**. Highly interpretable correlated features are visualized below the graph.

suppresses **Qxh7+** attribution graph, explaining why ablating the **g2** pawn feature or removing the pawn from the board substantially increases the probability of the mating move.

Blocking a Mate-in-One Threat Figure 19 presents the end-to-end attribution graph for a defensive case in which the model blocks an immediate mating threat on **h7**. The pathway first identifies the opponent's coordinated queen–rook pressure on **h7**, then routes this threat signal through a knight-movement feature that supports defending the square. In particular, **Lorsa.13.7717** carries the information that the knight can defend **h7**, while **Lorsa.14.3993** propagates this defended-square information back to the target-square representation of the selected action. This graph illustrates a *lookahead-to-action* pathway in which an anticipated opponent threat is converted into a concrete defensive move.

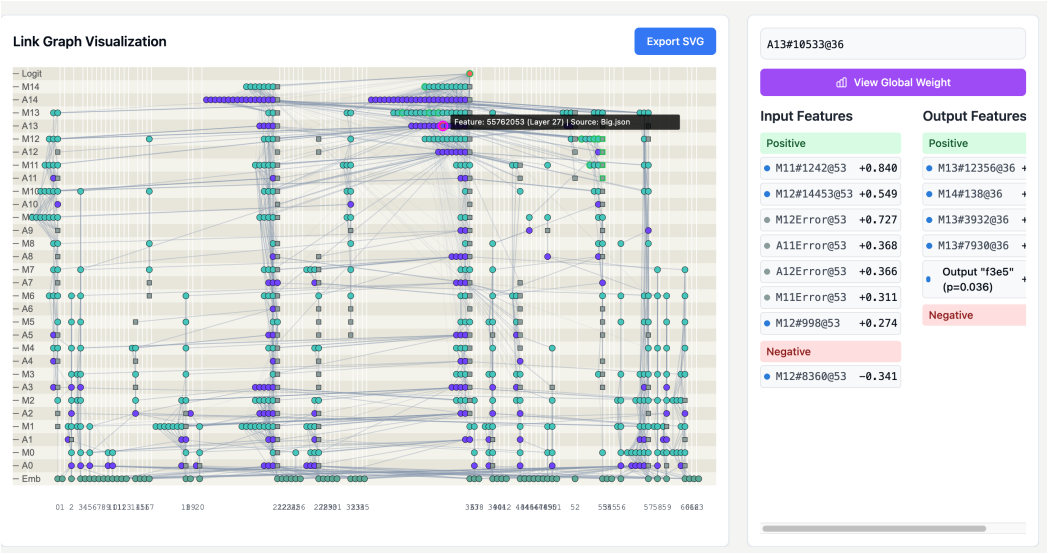


Figure 16: Enlarged view of the attribution graph for the internal look-ahead case.

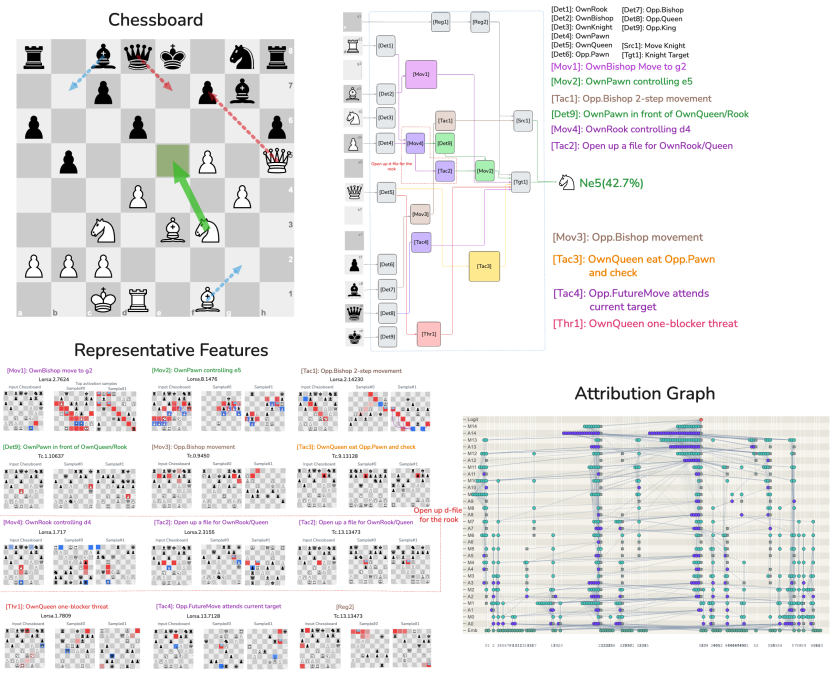


Figure 17: The end-to-end attribution graph for the **Ne5** sacrifice case. The graph summarizes the main pathway supporting BT4's choice, including features related to opening the d-file for the rook on **d1**, creating future mating pressure via **Qf7+**, enabling the follow-up development of **Bg2**, and accounting for plausible opponent-response considerations such as **Qe7**.

Offense with Concurrent Defensive Awareness. In this case shown in Figure 20, the model selects the correct policy **Qc4+**. Analysis of the corresponding attribution graph reveals that, while pursuing an immediate attacking continuation, the model simultaneously accounts for a potential mate-in-one threat from the opponent's queen. The square **c4** not only enables a diagonal attack on the opposing king but also effectively neutralizes this imminent defensive risk. We visualize the local attribution graph supporting this decision, highlighting key features associated with opponent mating threats and diagonal attack patterns, as summarized by the relevant supernodes.

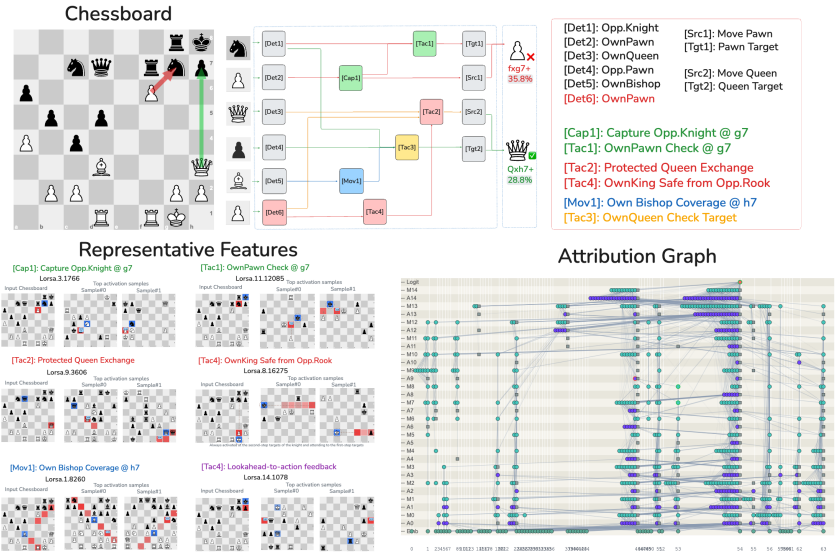


Figure 18: The end-to-end attribution graphs for the overly conservative decision discussed in Section 4.3. The figure compares the feature evidence supporting the suboptimal but conservative move **fxg7+** with the evidence supporting the correct mating move **Qxh7+**. The graph highlights how defensive signals associated with the **g2** pawn bias BT4 toward preserving king safety rather than selecting the immediate checkmate.

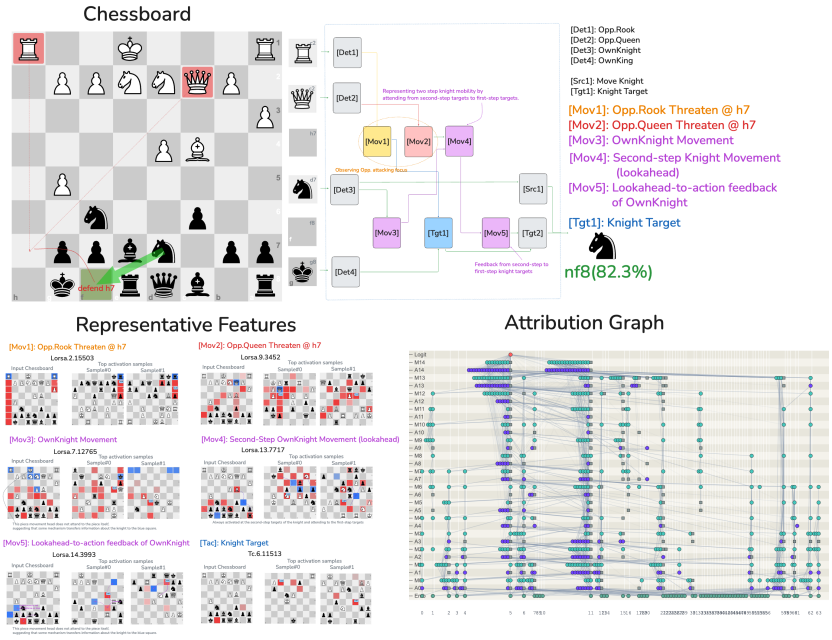


Figure 19: The end-to-end attribution graph for blocking a mate-in-one threat. The graph shows how the model detects the opponent's queen–rook coordination on **h7** and routes this threat information to a knight move that defends the square. Key features involved in this defensive pathway are visualized.

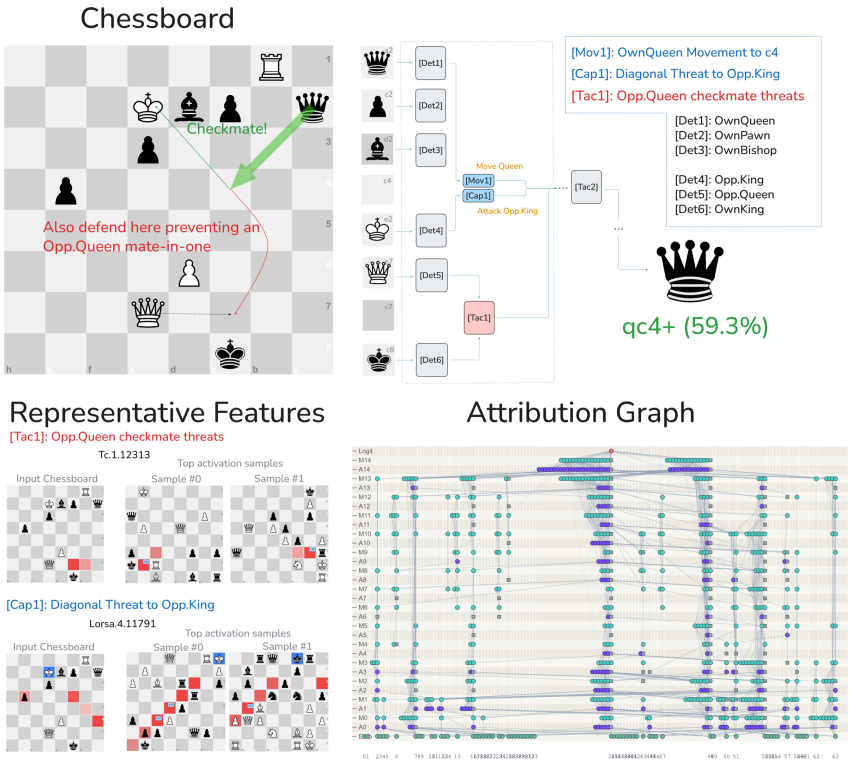


Figure 20: Attribution graph for a case where the model combines offensive and defensive reasoning. The figure shows the input chessboard position, the attribution graph supporting the selected policy **Qc4+**, and the associated supernode interpretations. Two key features are visualized: **Tc.1.12313** represents the awareness of the opponent queen's checkmate threats, and **Lorsa.4.11791** represents diagonal threats against the opposing king.