The Algorithm Is Not the Behavior: Learned Priors Override Look-Ahead in a Chess-Playing Neural Network

Elias Sandmann
Fraunhofer HHI

Sebastian Lapuschkin
Fraunhofer HHI
TU Dublin

Wojciech Samek
Fraunhofer HHI
TU Berlin

firstname.lastname@hhi.fraunhofer.de

Abstract

Recent mechanistic work has uncovered learned algorithms within neural networks, from modular arithmetic to search and planning in game-playing agents. But does algorithmic structure guarantee algorithmic behavior? We investigate this in Leela Chess Zero, the strongest neural chess engine, where prior work identified learned look-ahead. By extending the logit lens to its move-selecting policy network, we discover that correct puzzle solutions—including immediate checkmates—often appear in intermediate layers but are systematically overridden in the final output, a phenomenon we term *forgotten puzzles*. Replicating prior analyses on these positions, we find that look-ahead operates normally—future moves of the correct continuation are represented, causally important, and linearly decodable—ruling out a failure of the algorithm. Instead, late layers increasingly shift toward safe play over aggression. To test whether this shift drives the override, we steer the model against these preferences and recover 71.7% of forgotten puzzles, providing causal evidence that safety priors override algorithmically computed solutions. These findings show that algorithmic structure does not guarantee algorithmic behavior: a model can internally solve a problem and still output the wrong answer.

1 Introduction

When mechanistic analysis identifies a learned algorithm inside a neural network, does that algorithm determine the network's output? Several works have provided evidence for algorithmic computation within neural networks, from modular arithmetic (Nanda et al., 2023) and search in synthetic tasks (Brinkmann et al., 2024) to planning in Sokoban (Bush et al., 2025; Taufeeque et al., 2026) and look-ahead in a chess-playing neural network (Jenner et al., 2024). These works establish that learned algorithms exist and are causally active, focusing on how they function when they succeed. Less is known about how they can fail: can a model internally compute the correct solution through a learned algorithm only to output the wrong answer—and if so, what overrides it?

We investigate this by extending the logit lens (nostalgebraist, 2020)—which projects intermediate representations through a model's output head (Jastrzębski et al., 2018) to reveal what the network would predict at each layer—to the move-selecting policy network of Leela Chess Zero (Leela Chess Zero team), the model in which Jenner et al. (2024) identified learned look-ahead and the strongest neural chess engine available today (TCEC, 2026). Leela's encoder-only transformer architecture provides a distinct interpretability advantage: it treats each chessboard square as a token and uses all tokens simultaneously for prediction, giving the logit lens direct access to every intermediate activation contributing to the model's output. In decoder-only language models, by contrast, the logit lens is limited to decoding from a single token position and sees only part of the computation.

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

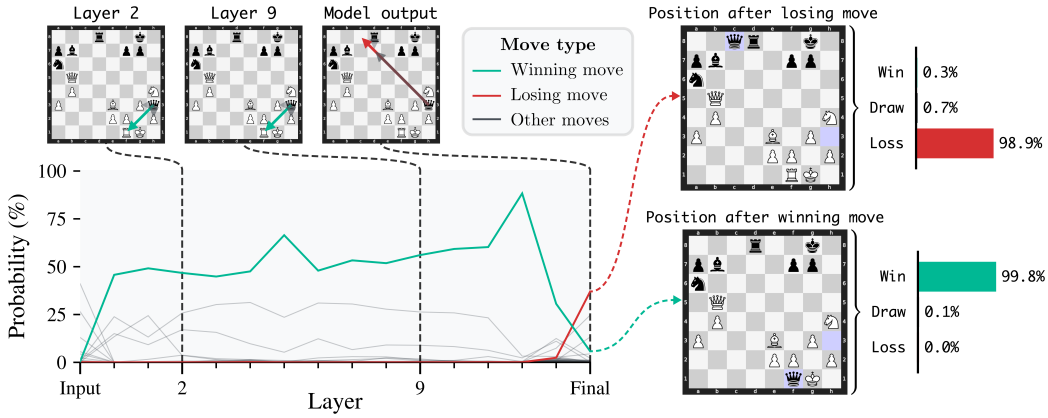


Figure 1: Our extended logit lens maps intermediate activations to policy distributions across transformer layers in Leela Chess Zero, revealing a forgotten mate-in-two puzzle with optimal line of play—the *principal variation*—1. ♕×**f1**+ ♔×**f1** 2. ♖**d1**#, Black to move. **Left:** The **correct** queen sacrifice ♕×**f1**+ maintains high probability through most layers, peaking at layer 12, before dropping sharply in the final layers. The **losing** move ♕**c8**—a quiet queen retreat to the back rank—remains near zero throughout the network and rises only in the final layer to become the top choice. **Right:** The value head, trained to predict game outcomes, correctly assesses both resulting positions, indicating that the policy head's final move selection contradicts the model's own evaluation of the consequences. At the puzzle position itself, the value head is consistent with the final policy and does not see the win. Full probabilities are in Appendix J, with additional examples in Appendix K.

Applying the logit lens reveals interpretable policies at every layer, with move preferences following a structured progression through recognizable chess heuristics of increasing complexity rather than converging smoothly on the final answer (Figure 2). But the layer-wise view also exposes a surprising phenomenon: in chess puzzles with a single clearly winning move, the correct solution often appears in intermediate layers only to be overridden in the final output, including immediate checkmates where even a single step of look-ahead should suffice (Figure 1).

Do these *forgotten puzzles* arise despite look-ahead functioning normally, or does the algorithm simply fail in these positions? By replicating the analyses of Jenner et al. (2024) on forgotten puzzles, we find that look-ahead operates as expected: future moves of the correct continuation are represented, causally important, and linearly decodable. If the algorithm is not at fault, what overrides it? Characterization of these positions reveals that the model overwhelmingly replaces tactical solutions, such as sacrifices, with quiet positional alternatives, suggesting that learned priors for safe play may be responsible. Such priors would reflect sensible generalizations from the training distribution that conflict with the correct move in rare tactical positions. To test for such priors, we measure which human-interpretable chess concepts, such as king safety and threats, each layer prioritizes, and find that late layers increasingly favor safer concepts over more aggressive ones. Gradient-based steering against these preferences recovers 71.7% of forgotten puzzles, providing causal evidence that *safety priors* override algorithmically computed solutions.

Our contributions are: (1) We extend the logit lens to the Post-LN architecture of Leela's policy network, yielding highly interpretable intermediate policies whose playing strength improves consistently with depth. (2) Using this, we identify forgotten puzzles—positions where intermediate layers find the correct solution but the final output does not—and verify that look-ahead functions normally in these positions. (3) We provide causal evidence that learned safety priors override these algorithmically computed solutions, recovering a majority of forgotten puzzles through gradient-based steering against safety concepts. (4) We show that forgotten puzzles are genuine failures of the network rather than of its objective, as the policy's own training target would place almost all of its weight on the solution. Since both the algorithm and the priors emerge from the same likelihood-based training objective, these results raise the question of whether such training can produce models that reliably prefer algorithmically computed solutions over distributional heuristics, even in domains with verifiable ground truth. More broadly, our findings imply that models may possess capabilities their outputs do not reflect, and that mechanistic understanding can identify such failure modes and suggest principled interventions to address them. Our code is available on GitHub.

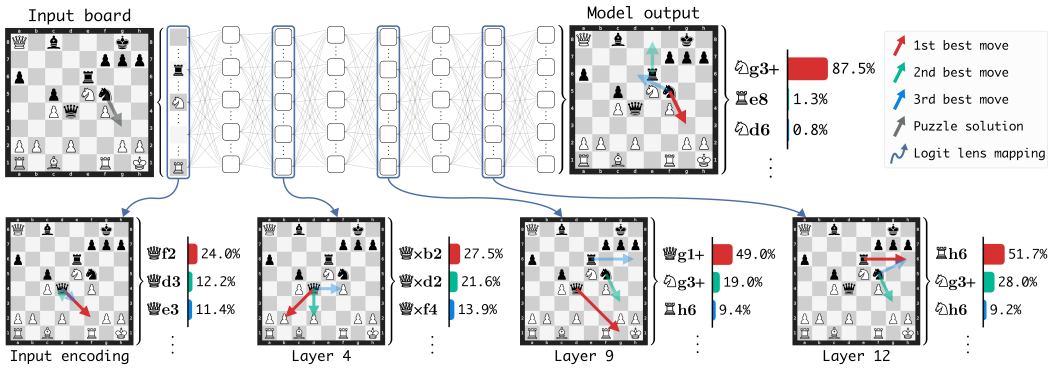


Figure 2: Our extended logit lens reveals progressive policy refinement across layers for a mate-in-two puzzle with principal variation 1. ♘**g3+ h×g3** 2. ♖**h6**#, Black to move. Layer-wise preferences follow a structured progression: from queen moves reflecting piece priors before any inter-square interaction in the input encoding, through greedy captures in early layers, to forcing checks by layer 9 and a positional rook lift at layer 12, before converging on the correct knight sacrifice ♘**g3+** in the final output. While the intermediate preferences shown are all losing, the trajectory is non-arbitrary, with each stage reflecting recognizable chess heuristics of increasing complexity. Full probabilities are provided in Appendix H, with additional examples in Appendix I.

2 Background

2.1 Leela Chess Zero

We analyze the `T82-768x15x24h` transformer model from Leela Chess Zero (Monroe et al., 2026), the network studied by Jenner et al. (2024). This model uses a Post-LN architecture similar to the original transformer (Vaswani et al., 2017) with DeepNorm scaling (Wang et al., 2024), featuring a 15-layer transformer encoder with 768-dimensional embeddings and 24 attention heads per layer. The shared encoder feeds two output heads: a policy head that outputs a distribution over legal moves and a value head that predicts game outcomes. Chess positions are encoded as 8×8 grids where each square corresponds to a token position. Architectural details are provided in Appendix D.

Leela is trained following the AlphaZero paradigm (Silver et al., 2018) and normally functions in tandem with Monte Carlo tree search (MCTS) as a chess engine. Its training games are generated by self-play, in which each move is chosen by MCTS guided by a Leela network: the policy head supplies the prior over which moves to explore, and the value head predicts the game outcome for each position the search reaches. Moves with a higher prior or a better predicted outcome receive more visits, and the engine ultimately plays the move with the most visits at the root. For each position played in these games, the policy head is trained by cross-entropy against the normalized root visit counts of the search, and the value head against the final game outcome. Our main claims concern the policy network, which already demonstrates strong play even without external search (lepned, 2024), and we use search only to examine the network's training target and how its prior affects the engine in deployment. Search details are provided in Appendix D.4.

2.2 Encoder-only Post-LN logit lens

The logit lens (nostalgebraist, 2020) projects intermediate activations after layer ℓ through the final layer normalization and unembedding matrix to obtain layer-wise predictions. This approach works seamlessly for Pre-LN transformers, where layer normalization precedes each sublayer and leaves the residual stream free of normalization operations. Post-LN architectures create a challenge by applying normalization after residual connections instead, transforming the residual stream at each layer and creating non-linear dependencies that prevent straightforward application of the logit lens.

Functionally, the Pre-LN logit lens is equivalent to applying zero ablation to all sublayer outputs beyond a given layer ℓ (Belrose et al., 2023). We extend this principle to Post-LN models by applying the same zero ablation while preserving the subsequent layer normalizations. For consistency we also ablate layer normalization biases for layers beyond ℓ, as justified in Appendix C.

Since Leela is an encoder that uses representations from all tokens simultaneously, we project the intermediate representations of all 64 squares through the policy head to obtain policies at each layer.

2.3 Evaluation setup

To verify that our Post-LN logit lens produces meaningful intermediate predictions, we evaluate the general playing strength and tactical puzzle-solving ability of layer-wise policies following Ruoss et al. (2024). Additionally, we track distributional properties of intermediate policies across layers.

Playing strength assessment We conduct a round-robin tournament between policies derived from representations at the input and all 15 transformer layers. Each pairing plays 200 Encyclopedia of Chess Openings (Matanović, 1978) positions with one game per side using argmax move selection and five games per side using a temperature of $\tau = 1.0$. Elo ratings are computed using BayesElo (Coulom, 2008). We include the Leela policy network from Ruoss et al. (2024) as a fixed anchor using argmax selection to match their evaluation and obtain absolute Elo scores. Additionally, we deploy the layer-wise policies as bots on Lichess (Lichess.org) across multiple time controls, using a temperature of 1.0 for the first five full-moves to introduce opening diversity.

Puzzle-solving performance We evaluate tactical understanding using the *general dataset* of $10{,}000$ Lichess puzzles from Ruoss et al. (2024), each constructed with a single clear winning line—the *principal variation*—while all other moves are significantly inferior, though not necessarily losing. For each layer's policy, we use argmax selection to predict moves and consider a puzzle solved if this reproduces all the player's moves of the principal variation correctly or leads to checkmate.

Intermediate policy dynamics To characterize how layer-wise policies evolve across depth, we track policy entropy and Kendall's τ ranking correlation between intermediate and final move rankings at each layer. Additional metrics are provided in Appendix E.

2.4 Learned look-ahead

Jenner et al. (2024) demonstrated that Leela internally represents future moves of correct puzzle continuations and that these representations are causally important for its output. They analyzed positions where the model correctly solves tactical puzzles with a clear principal variation and presented three lines of evidence, which we briefly summarize below. We replicate their analyses on forgotten puzzles to determine whether this forgetting reflects a failure of the look-ahead mechanism itself or whether look-ahead operates normally but is overridden.

Dataset To study look-ahead, they filter for positions where the model is especially likely to employ look-ahead rather than simple heuristics—tactical puzzles with a unique best line that are difficult to evaluate without considering future moves. They focus on puzzles the model solves correctly; applying their criteria yields $23{,}601$ positions (the *solved dataset*). We apply identical filtering criteria but select for forgotten puzzles, yielding $8{,}648$ positions (the *forgotten dataset*), indicating that forgotten puzzles are far from an edge case. To ensure these represent genuine forgetting rather than absence of look-ahead, we also require the full model to assign $> 50\%$ probability to all the player's subsequent moves in the principal variation, given the preceding correct moves, confirming that the model demonstrably understands the winning continuation. Neither dataset filters for the presence of look-ahead mechanisms; any effects observed are emergent rather than selected for.

Activation patching To measure the causal importance of individual square representations, Jenner et al. (2024) employ activation patching. For each position, a corrupted board state is generated by applying a small modification—such as adding or removing a single piece—selected to substantially affect the model's output while minimally affecting a weaker model, to target mechanisms beyond simple heuristics. Representations from the corrupted forward pass are then patched into the clean forward pass at each layer one square at a time. The resulting change in log-odds of the correct move measures each square's causal importance at that depth. They find that target squares of future moves in the principal variation exhibit outsized causal effects, indicating that look-ahead information is actively used. Although the model does not select the correct move in forgotten puzzles, it still assigns some probability to it. If this residual probability originates from look-ahead, causal interventions targeting look-ahead should reduce it, and log-odds reduction remains a meaningful metric.

Attention head ablation Jenner et al. (2024) identified specific attention heads that transfer information between squares involved in future moves—in particular, heads that attend from the first move's target square to the third move's target square, effectively relaying look-ahead information backward along the principal variation. To test the causal role of this information flow, they zero-ablate the single attention entry corresponding to this query-key pair and find that it has an outsized effect on the log-odds of the correct move.

Probing To test whether information about future moves is explicitly encoded in the model's representations, Jenner et al. (2024) train bilinear probes (Hewitt and Liang, 2019), which score pairs of squares through a bilinear form of their activations, to predict the third move in the principal variation in two steps: first predicting the target square of the third move from activations at the first move's target square, then predicting the source square conditioned on this predicted target. Full methodological details for all look-ahead analyses are provided in Appendix L.

3 Results

3.1 Capability progression and policy dynamics

Tournament strength Table 1 reports Elo ratings across layers. Playing strength increases monotonically with depth but suggests a three-phase progression rather than uniform improvement. Early layers show rapid gains through layer 5, middle layers form a performance plateau through approximately layer 10, and late layers demonstrate sharp strengthening beginning around layer 11. This pattern holds consistently under both deterministic ($\tau = 0$) and stochastic ($\tau = 1$) move selection. Real-world Lichess deployment shows similar trends with late-layer strengthening (Appendix F).

Table 1: Playing strength (Elo rating) across transformer layers and evaluation methods

Evaluation	Input	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	Full	Anchor
Internal Tournament ($\tau = 0$)	351	609	655	730	832	908	993	997	1028	1030	1055	1082	1117	1356	1670	2268	2292
Internal Tournament ($\tau = 1$)	46	387	391	469	585	718	810	799	816	813	851	872	876	1065	1119	1367	–
Lichess Bullet	817	843	907	930	961	998	1071	1085	1087	1096	1094	1124	1159	1399	1622	2306	–
Lichess Blitz	545	631	611	655	740	723	815	815	827	906	952	860	946	1205	1482	2099	–
Lichess Rapid	492	617	569	669	789	791	871	907	954	954	985	1008	1072	1324	1637	2237	–

Puzzle-solving capabilities Figure 3 shows clear improvement in puzzle-solving ability across network layers within each Elo range, and a consistent decrease in performance across increasing difficulty levels for all layers. Dashed lines and shading mark phase boundaries with annotated slope ratios indicating relative improvement rates between phases. While the early-to-middle phase distinction is less pronounced than in tournaments, the final-phase acceleration is clearly visible, particularly for harder puzzles where improvement rates reach 24.8 times the middle phase.

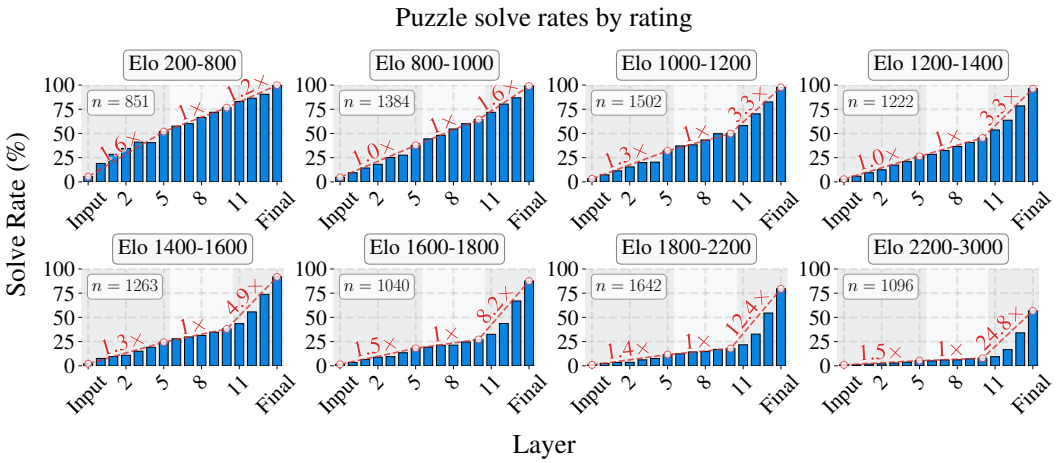


Figure 3: Puzzle-solving performance across layers, stratified by Elo rating. Red dashed lines mark phase boundaries derived from tournament analysis, indicating phase-specific improvement rates.

Distributional analysis Kendall's τ correlation between intermediate and final layer move rankings stays near zero through middle layers and rises sharply only in the final layers (Appendix E). Policy entropy remains approximately constant across layers, indicating that intermediate predictions reflect meaningful preference distributions. Constant entropy with changing rankings implies that the model reorganizes move preferences across layers rather than gradually sharpening them.

Solution discovery and forgetting Figure 4 tracks three metrics across layers: the **current** solve rate at each layer, the **cumulative** solve rate across all layers up to the current one, and the rate of puzzles solved for the **first time** at each layer. The gap between **current** and **cumulative** rates—highlighted by the **hatched** region—shows that solutions are frequently discovered and subsequently discarded. The **discovery** rate follows the three-phase pattern from playing strength, with reduced discoveries during the middle-layer plateau and renewed acceleration in late layers where many puzzles are solved for the first time. The gap between the final **cumulative** and **current** solve rates represents **forgotten** puzzles: solutions discovered at intermediate layers but lost in the output.

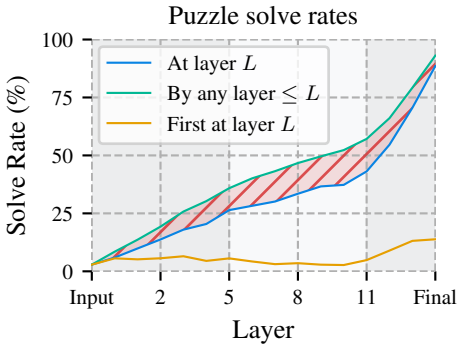


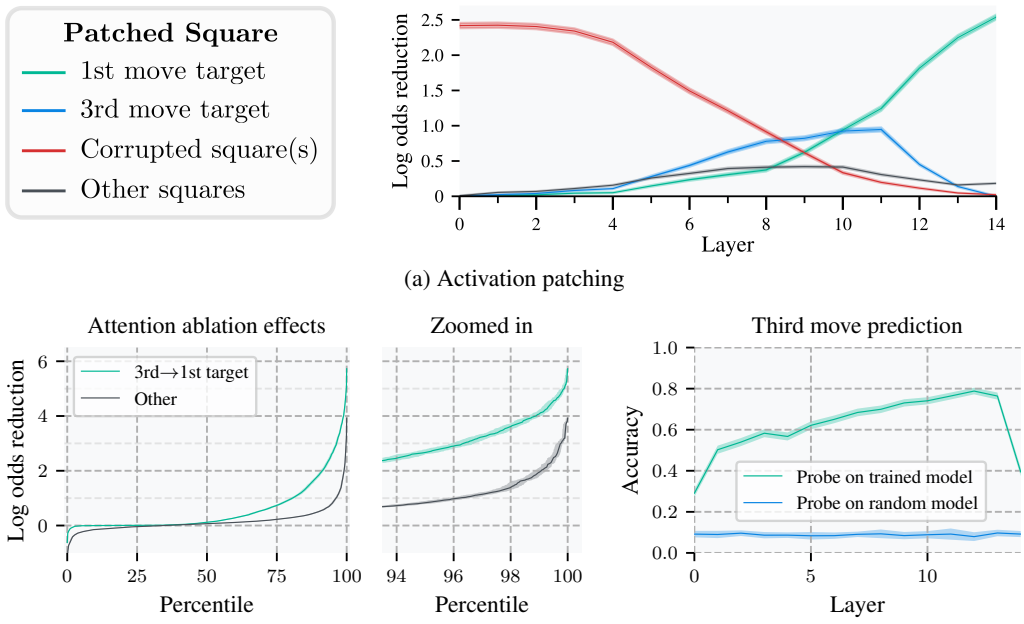
Figure 4: Layer-wise puzzle-solving performance across network depth. The hatched region indicates puzzles solved by at least one previous layer but not by the current layer—forgotten solutions.

3.2 Forgotten puzzles and the limits of learned look-ahead

Having established that our logit lens produces meaningful intermediate policies, we turn to forgotten puzzles—and find a systematic pattern rather than random failures. In the general dataset, 57% of forgotten puzzles involve a sacrifice[1], compared to 26% of solved ones. Within the forgotten dataset, 87.3% of puzzles involve a sacrifice and the correct moves are overwhelmingly forcing (76.9% checks, 77.9% captures), yet the model replaces them with quiet moves in 68.3% of cases. The model's own value head assigns higher win probability to the position after the correct move than to the position after the model's chosen move in 97.0% of forgotten puzzles (Appendix M). Notably, 45.6% are puzzles whose principal variation ends in checkmate, about half of them mate-in-two positions where even a single step of reliable look-ahead should guarantee playing the correct move.

Look-ahead analysis Could these puzzles be solved purely by shallow heuristics or chance in intermediate layers, without any contribution from look-ahead? The localization of the override suggests otherwise: 63.4% of forgotten puzzles remain solved after the final attention sublayer, indicating that the override is concentrated in the final feed-forward layer. To directly test whether look-ahead is present and causally contributes to these solutions, we apply the methodology described in Section 2.4 to the forgotten dataset. Figure 5 shows that all three lines of evidence yield qualitatively similar patterns to those reported by Jenner et al. (2024) on solved puzzles, albeit with somewhat smaller effect sizes for the causal interventions, as forgotten puzzles have lower baseline probability on the correct move and thus less log-odds to lose. Activation patching reveals that the **3rd move target** square exhibits outsized causal effects in middle to late layers, diminishing after layer 11 as look-ahead information is transferred to the **1st move target** square, the same pattern as in solved puzzles. Zero-ablating the single attention entry in L12H12 that relays information from the **3rd** to the **1st move target** square reduces the log-odds of the correct move by more than 1.5 in more than 10% of forgotten puzzles, and in 61.6% of cases this single entry has a larger effect than ablating all remaining 4,095 entries simultaneously. For the example in Figure 1, ablating this single scalar drops the correct move's probability from 5.8% to 1.0%, corresponding to a log-odds reduction of 1.79. Among forgotten puzzles in which patching shows look-ahead to be present and the correct move is on top at the penultimate layer, cutting this entry alone removes it from the top at that layer in 29.9% of cases. Bilinear probes decode the third move with up to 79% accuracy for forgotten puzzles, compared to 90% for solved puzzles, though probes on randomly initialized models also

[1] A puzzle involves a sacrifice if the player's material balance drops below its starting value along the principal variation.



(a) Activation patching

(b) Attention head ablation

(c) Probing

Figure 5: Look-ahead analysis from Jenner et al. (2024) on forgotten puzzles. **(a)** Mean activation patching effects in the residual stream across layers. The **1st move target** and **3rd move target** squares show outsized effects compared to the **other** squares line. The **corrupted** square(s) effect diminishes in late layers. Error bars are two times the standard error of the mean. **(b)** Zero-ablation of the attention entry from the third to the first move target square in L12H12. The **3rd→1st target** line shows ablation of this single entry, while the **other** line shows ablation of all remaining 4,095 entries simultaneously. Lines show the effect at a given percentile of puzzles sorted by effect size. Error bars are 95% CIs. **(c)** Bilinear probe accuracy for predicting the third move across layers on a **trained** and **randomly** initialized model. Error bars combine variance across five probe training runs with finite-sample uncertainty in accuracy estimates.

show lower accuracy on forgotten puzzles. Results for additional look-ahead heads identified by Cruz (2025) and the corresponding findings on solved puzzles are in Appendix L.

Taken together, the evidence demonstrates that the look-ahead mechanism operates normally in forgotten puzzles—future moves are represented, causally important, and linearly decodable. The model computes the correct tactical solution but does not use it. The presence of algorithmic structure does not guarantee algorithmic behavior. If the failure is not in look-ahead, what overrides it?

3.3 Safety priors override tactical solutions

The forgotten puzzle characterization revealed that the model systematically replaces forcing tactical solutions with safe, positional alternatives. To test whether this pattern reflects learned priors for safe play, we analyze how the preferences of layer-wise policies for chess concepts evolve across network depth. Since the model's internal representation of safety is unknown, we rely on handcrafted evaluation terms as human-interpretable proxies, focusing on concepts with a direct relationship to safe play, such as the safety of each king and the threats each side poses.

Layer-wise concept preferences To examine which chess concepts each layer prioritizes, we analyze how layer-wise policies weight moves by their conceptual effects. Following McGrath et al. (2022), who trained linear concept probes on AlphaZero's representations, we use Stockfish 8's (Stockfish developers, 2016) handcrafted evaluation terms as human-interpretable concepts, a choice made for comparability with Sadler and Regan (2019). Each concept maps a position to a scalar value that measures an interpretable property, such as king safety or threats. Rather than probing for concepts, we measure *concept preferences* directly from layer-wise move probabilities.

For each move m from position s to resulting position s', and each concept c, we compute $\Delta c_m = c(s') - c(s)$, representing the change in c caused by m. All evaluations are from the perspective of the current player, so positive values indicate an improvement. At each layer ℓ, we compute:

$$\Delta c_\ell = \sum_{m \in \text{legal}(s)} \pi_\ell(m) \cdot \Delta c_m = \mathbb{E}_{\pi_\ell}[\Delta c_m] \tag{1}$$

where $\pi_\ell(m)$ is the move probability assigned by layer ℓ's policy. This represents the expected concept change when sampling moves according to π_ℓ. Full details and plots are provided in Appendix N.

The left panel of Figure 6 shows the evolution of four safety-related concepts across layers. Early and middle layers favor aggressive over defensive concepts, with higher Δc_ℓ for **opponent king vulnerability** and **own threats**, while later layers increasingly prioritize safe play, increasing **own king safety** and reducing **opponent threats**, with all four concepts converging to similar values.

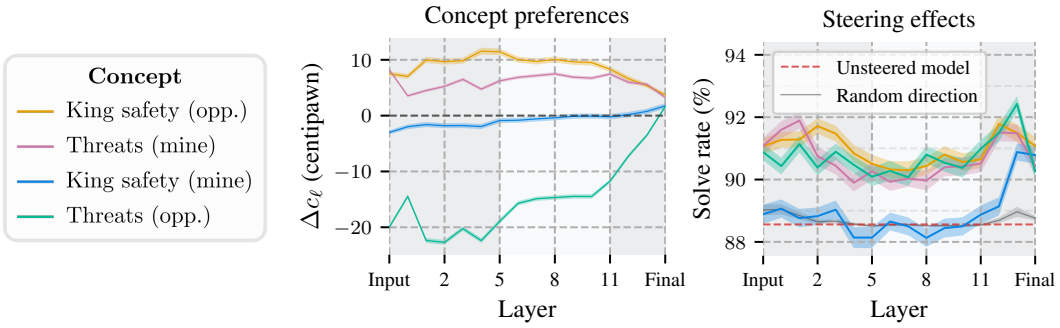


Figure 6: Safety-related concept analysis across layers. **Left:** Mean concept preference Δc_ℓ for four safety-related concepts, measured in centipawns with 95% CI. Late layers shift from aggressive toward conservative concept preferences. **Right:** Held-out solve rate when steering with each concept's gradient, with the strength $\alpha \in \{\pm 0.25, \pm 0.5, \ldots, \pm 1.5\}$ selected per layer on half of the puzzles and evaluated on the other half. Lines show the mean and shaded areas one standard deviation across 200 random splits. Steering against the late-layer safety preferences substantially improves solve rates, peaking at layer 13. The **gray line** shows the mean of 50 random unit vectors under the same protocol, with 95% CI across vectors, while the **dashed line** indicates the unsteered baseline.

Causal validation via concept steering As a proof of concept for the hypothesis that safety priors contribute to the forgotten puzzle phenomenon, we perform activation steering (Dathathri et al., 2020; Subramani et al., 2022) using a first-order approximation in activation space with respect to the concept preference metric. For a given concept c at layer ℓ with steering strength α, we modify the residual stream activations $\mathbf{h}_\ell$ as:

$$\mathbf{h}'_\ell = \mathbf{h}_\ell \pm \alpha \cdot \mathbb{E}[\nabla_{\mathbf{h}_\ell} \Delta c_L] \cdot \frac{\mathbb{E}[\|\mathbf{h}_\ell\|]}{\|\mathbb{E}[\nabla_{\mathbf{h}_\ell} \Delta c_L]\|} \tag{2}$$

where Δc_L is the concept preference of the full model and the gradient is normalized and scaled to the layer's mean activation norm. Gradients are computed on positions sampled from general play, independent of any puzzle dataset. We sweep over steering strengths α and intervention layers ℓ for each concept and measure the solve rate on the general puzzle dataset, capturing both recovery of forgotten puzzles and regressions. Using Monte Carlo cross-validation (Arlot and Celisse, 2010), we select the best strength for each concept and layer on a random half of the puzzles, evaluate it on the other and report the mean and standard deviation over 200 random splits.

The right panel of Figure 6 shows steering effects across layers. They are largest when steering the model to place less emphasis on mitigating **opponent threats** at layer 13, which reaches a solve rate of $92.4 \pm 0.2\%$ compared to the 88.6% **baseline**. With $\alpha = -0.5$, the best strength on all puzzles, this setting solves 519 additional puzzles with 129 regressions. Notably, it recovers 62.4% of the forgotten puzzles in the general dataset while also solving 34.4% of puzzles the model never solved at any layer. Applied to the held-out forgotten dataset, the same configuration recovers 71.7% of its puzzles. The optimal steering direction aligns with the concept preference analysis: for all four

concepts, performance improves when placing less emphasis on safety, specifically reducing the weight on **own king safety** and **opponent threats against us** while increasing emphasis on **threats we pose** and **opponent king vulnerability**. For three of four concepts, steering yields improvements across all layers, substantially exceeding the **random baseline**, with the strongest single effect at layer 13, just before the override.

The observed regressions suggest that not all puzzles benefit from reduced safety preferences. Nevertheless, puzzles are adversarial by construction—the winning move is frequently a sacrifice or forcing sequence that conflicts with conservative heuristics learned from the training distribution, explaining why reducing safety preferences yields a substantial net improvement on puzzles. That a simple gradient-based intervention targeting safety-related concepts suffices to recover a majority of forgotten puzzles supports the hypothesis that learned safety priors are a primary cause of the forgotten puzzle phenomenon. While other concepts also improve performance above the **random baseline**, the most directly safety-related concepts yield the largest and most consistent effects; complete steering analyses for all concepts are provided in Appendix O.

3.4 Search and the training objective

Leela's policy is trained to match the outcome of search rather than to solve puzzles, so forgotten puzzles could simply be the product of a training objective that is misspecified on these positions rather than a genuine failure of the network. To test this, we run the self-play search of the training procedure on each puzzle with the settings used to train this network, and compare the resulting training target with the policy predicted by the network. Additionally, we examine how forgotten puzzles affect the full engine in deployment, with details of both experiments in Appendix P.

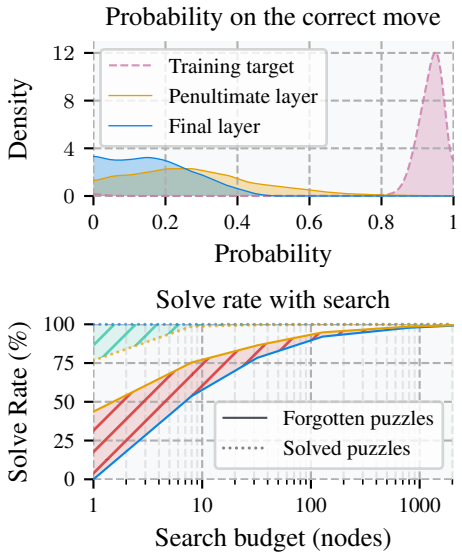


Figure 7: Training target and search. **Top:** Probability assigned to the correct move on forgotten puzzles by the **training target**, the **penultimate layer** and the **final layer**. **Bottom:** Engine solve rate by search budget, with the policy read from the **final layer** or the **penultimate layer**. Shading shows the net effect of the final layer, **worse** on forgotten and **better** on solved puzzles. With a single node, the engine plays the policy's top move.

Training target The top panel of Figure 7 shows that on forgotten puzzles the **training target** places a mean of 93.1% of its visits on the correct move, and its most visited move is the correct one in 99.4% of puzzles. The correct move also has the highest value at the root in 99.5% of puzzles, so more search would concentrate the target further. Under the policy's own training loss, the **penultimate layer** is closer to the target than the **final layer** in 77.5% of forgotten puzzles. These positions most likely never occurred in training, but had they occurred, the training objective would have asked for the solution. The override therefore does not come from the objective itself, but is consistent with a prior that is sensible on the training distribution and carries over to positions where it conflicts with the solution.

Search in deployment We run Leela with default settings at different search budgets and count a puzzle as solved if the engine plays the correct move. The bottom panel of Figure 7 shows that, given enough search, the engine recovers forgotten puzzles, but the **final layer** makes this **more expensive** than a policy read from the **penultimate layer**. On solved puzzles, where the final layer's top move is already correct, it **helps** instead.

4 Related work

Chess models and their interpretability The success of AlphaZero (Silver et al., 2018) and Leela Chess Zero (Leela Chess Zero team) has motivated a growing body of interpretability work on neural

chess engines, including concept probing (McGrath et al., 2022), concept discovery (Schut et al., 2025), and evidence of learned look-ahead (Jenner et al., 2024; Cruz, 2025). Separately, Karvonen (2024) studied world models in language models trained on chess move sequences, and Ruoss et al. (2024) trained a transformer on Stockfish's evaluations to play chess without external search. These models differ in training paradigm but are architecturally similar, suggesting the tension between learned algorithms and distributional priors we identify may extend to them as well.

Intermediate decoding nostalgebraist (2020) introduced the logit lens for decoder-only language models, with subsequent work extending it to encoder-decoder architectures (Langedijk et al., 2024), to anticipating future tokens (Pal et al., 2023) and aiming to improve prediction accuracy (Belrose et al., 2023; Yom Din et al., 2024), though with objectives that penalize the intermediate-to-final disagreements we study. The logit lens has been applied to Leela Zero, a superhuman Go model, albeit limited to divergence metrics (Du, 2023), and used to identify distinct computational phases in language models (Lad et al., 2025), a pattern we also observe in Leela.

Learned algorithms in neural networks Beyond language models, where circuit discovery has become a major research direction, recent work has mechanistically identified learned algorithms in other domains including search in synthetic tasks (Brinkmann et al., 2024) and planning in Sokoban (Bush et al., 2025; Taufeeque et al., 2026). These works demonstrate that identified circuits are causally relevant for the model's output. In contrast, Nikankin et al. (2025) find that neural networks can solve seemingly algorithmic tasks like arithmetic through ensembles of heuristics rather than generalizable algorithms. Our work sits between these findings: algorithmically computed solutions can be present and causally active yet still be overridden in the final output by competing heuristics.

5 Conclusion

We have shown that the presence of learned algorithmic structure in a neural network does not guarantee that the network's output will reflect it. In Leela Chess Zero, look-ahead computes correct tactical solutions that are systematically overridden by safety priors in the final layers—a failure of the network rather than of its training objective. Concept preference analysis shows that late layers shift toward safe play, and gradient-based steering against these priors recovers a majority of forgotten puzzles, providing causal evidence for this override. More broadly, our results suggest that circuit discovery alone is insufficient for predicting model behavior: identified algorithms may not determine the output, and incorrect outputs may not indicate absent capabilities. Such failures can be exposed by mechanistic analysis, which can also point to interventions that correct them.

Limitations The main limitations of our work are: (1) While our results suggest safety priors as a primary cause of forgotten puzzles, we do not provide a precise mechanistic description of how these priors are implemented. (2) We verify the forgotten puzzle phenomenon across multiple Leela networks (Appendix G), but our causal analyses are conducted on a single network. (3) Our concept analysis uses Stockfish's evaluation terms as proxies and is restricted to one-step position evaluations; the model likely encodes features that do not map cleanly onto these concepts.

Impact There has been growing confidence that identifying algorithmic circuits in neural networks yields understanding of model behavior. Our results introduce a caveat: correctly identified algorithms may be overridden by competing priors, meaning that behavioral predictions based on circuit analysis alone may be incomplete. While our analysis focuses on a chess-playing model, the tension between learned algorithms and priors likely exists in any model trained on broad distributions, with or without search. Indeed, related phenomena have been observed in language models, where correct answers decodable at intermediate layers are overridden in late layers by false demonstrations (Halawi et al., 2024) or sycophancy priors (Wang et al., 2026). Learning from search is also not unique to Leela (Appendix A): in reasoning models, the language model acts as a policy guiding a search over the space of possible chains of thought (Chollet, 2024), and reinforcement learning with verifiable rewards distills successful chains back into the model (Lambert et al., 2025; Guo et al., 2025). Beyond interpretability, such overrides connect to broader concerns about hidden capabilities in deployed models—capabilities present in internal computations but not reflected in behavior (Gekhman et al., 2025), which may surface under distribution shift or adversarial inputs (Wei et al., 2023). Whether similar overrides affect algorithmic solutions in domains such as mathematical reasoning remains an open question.

Author contributions

Elias conceived, implemented, and conducted the study, performed all experiments, and wrote the manuscript. Sebastian and Wojciech provided supervision and feedback throughout the project.

Acknowledgments and Disclosure of Funding

We gratefully acknowledge Jenner et al. (2024) for the codebase and inference framework on which this work is built, and Ruoss et al. (2024) for the puzzle evaluation and tournament setup. Without their open-source releases, this research would not have been possible. We also thank the Leela Chess Zero team and the Stockfish developers for making their engines and networks openly available, and Lichess.org for its open puzzle database and bot platform. Our gratitude also goes to Ekkehard Schnoor for pointing us to chess as a research direction, and to Marie and Enno Sandmann for always distracting us at the right moments.

This work was supported by the Federal Ministry of Research, Technology and Space (BMFTR) as grants [BIFOLD (01IS18025A, 01IS180371I), xJuRAG (16IS25015B)]; the European Union's Horizon Europe research and innovation programme (EU Horizon Europe) as grant ACHILLES (101189689); and the German Research Foundation (DFG) as research unit DeSBi [KI-FOR 5363] (459422098). The authors declare no competing interests.

References

Sylvain Arlot and Alain Celisse. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4:40–79, 2010. doi: 10.1214/09-SS054. URL https://doi.org/10.1214/09-SS054.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens, 2023. URL https://arxiv.org/abs/2303.08112.

Jannik Brinkmann, Abhay Sheshadri, Victor Levoso, Paul Swoboda, and Christian Bartelt. A mechanistic analysis of a transformer trained on a symbolic multi-step reasoning task. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4082–4102, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.242. URL https://aclanthology.org/2024.findings-acl.242/.

Thomas Bush, Stephen Chung, Usman Anwar, Adrià Garriga-Alonso, and David Krueger. Interpreting emergent planning in model-free reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=DzGe40glxs.

François Chollet. OpenAI o3 breakthrough high score on ARC-AGI-Pub. ARC Prize blog, 2024. URL https://arcprize.org/blog/oai-o3-pub-breakthrough. Accessed: 2026-10-27.

Rémi Coulom. Whole-history rating: A Bayesian rating system for players of time-varying strength. In *Computers and Games*, 2008.

Diogo Cruz. Understanding the learned look-ahead behavior of chess neural networks. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=np4Bg2zIxL.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1edEyBKDS.

dje-dev. Discussion on layer normalization in transformer architecture. Message in the Leela Chess Zero Discord server, 2024. URL https://discord.com/channels/425419482568196106/425419999096733706/1317222179901018183. Accessed: 2026-10-27.

Haoxing Du. Inside the mind of a superhuman Go model: How does Leela Zero read ladders?, 2023. URL https://www.lesswrong.com/posts/FF8i6SLfKb4g7C4EL/inside-the-mind-of-a-superhuman-go-model-how-does-leela-zero. Accessed: 2026-10-27.

Zorik Gekhman, Eyal Ben David, Hadas Orgad, Eran Ofek, Yonatan Belinkov, Idan Szpektor, Jonathan Herzig, and Roi Reichart. Inside-out: Hidden factual knowledge in LLMs. In *Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=f7GG1MbsSM.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

Danny Halawi, Jean-Stanislas Denain, and Jacob Steinhardt. Overthinking the truth: Understanding how language models process false demonstrations. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Tigr1kMDZy.

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

Stanisław Jastrzębski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, and Yoshua Bengio. Residual connections encourage iterative inference. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJa9iHgAZ.

Erik Jenner, Shreyas Kapur, Vasil Georgiev, Cameron Allen, Scott Emmons, and Stuart Russell. Evidence of learned look-ahead in a chess-playing neural network. In *Advances in Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=8zg9sO4ttV&noteId=knXvSjTD1G.

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

Vedang Lad, Jin Hwa Lee, Wes Gurnee, and Max Tegmark. Remarkable robustness of LLMs: Stages of inference? In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Wxh5Xz7NpJ.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=i1uGbfHHpH.

Anna Langedijk, Hosein Mohebbi, Gabriele Sarti, Willem Zuidema, and Jaap Jumelet. DecoderLens: Layerwise interpretation of encoder-decoder transformers. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4764–4780, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.296. URL https://aclanthology.org/2024.findings-naacl.296/.

Leela Chess Zero team. Leela Chess Zero. URL https://lczero.org/.

Leela Chess Zero team. CCRL dataset, 2018. URL https://lczero.org/blog/2018/09/a-standard-dataset/. 2.5 million games from CCRL 40/40 and 40/4 tournaments; Accessed: 2026-10-27.

lepned. How well do Lc0 networks compare to the greatest transformer network from DeepMind?, 2024. URL https://lczero.org/blog/2024/02/how-well-do-lc0-networks-compare-to-the-greatest-transformer-network-from-deepmind/. Accessed: 2026-10-27.

Lichess Bot Devs. lichess-bot: A bridge between Lichess API and chess engines. https://github.com/lichess-bot-devs/lichess-bot, 2025. Accessed: 2026-10-27.

Lichess.org. Lichess: Free online chess. URL https://lichess.org. Accessed: 2026-10-28.

Aleksandar Matanović. *Encyclopaedia of Chess Openings*. Batsford Limited, 1978.

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), November 2022. ISSN 1091-6490. doi: 10.1073/pnas.2206625119. URL http://dx.doi.org/10.1073/pnas.2206625119.

Timothee Mickus, Denis Paperno, and Mathieu Constant. How to dissect a Muppet: The structure of transformer embedding spaces. *Transactions of the Association for Computational Linguistics*, 10: 981–996, 2022. doi: 10.1162/tacl_a_00501. URL https://aclanthology.org/2022.tacl-1.57/.

Diganta Misra. Mish: A self regularized non-monotonic activation function. In *British Machine Vision Conference*, 2020. doi: 10.5244/C.34.191. URL https://bmva-archive.org.uk/bmvc/2020/conference/papers/paper_0928.html.

Daniel Monroe. Transformer progress, February 2024. URL https://lczero.org/blog/2024/02/transformer-progress/. Accessed: 2026-10-27.

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

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=9XFSbDPmdW.

Yu Nasu. Efficiently updatable neural-network-based evaluation functions for computer shogi. Appeal document, 28th World Computer Shogi Championship, 2018. URL https://www.apply.computer-shogi.org/wcsc28/appeal/the_end_of_genesis_T.N.K.evolution_turbo_type_D/nnue.pdf. In Japanese. Accessed: 2026-10-27.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. Arithmetic without algorithms: Language models solve math with a bag of heuristics. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=O9YTt26r2P.

nostalgebraist. Interpreting GPT: the logit lens, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens. Accessed: 2026-10-27.

Koyena Pal, Jiuding Sun, Andrew Yuan, Byron Wallace, and David Bau. Future lens: Anticipating subsequent tokens from a single hidden state. In *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pages 548–560. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.conll-1.37. URL http://dx.doi.org/10.18653/v1/2023.conll-1.37.

Anian Ruoss, Grégoire Delétang, Sourabh Medapati, Jordi Grau-Moya, Li Kevin Wenliang, Elliot Catt, John Reid, Cannada A. Lewis, Joel Veness, and Tim Genewein. Amortized planning with large-scale transformers: A case study on chess. In *Advances in Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=XlpipUGygX.

Matthew Sadler and Natasha Regan. *Game Changer: AlphaZero's Groundbreaking Chess Strategies and the Promise of AI*. New in Chess, Alkmaar, Netherlands, 2019. ISBN 978-90-5691-818-7.

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/abs/10.1073/pnas.2406675122.

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.

Stockfish developers. Stockfish. https://github.com/official-stockfish/Stockfish.

Stockfish developers. Stockfish 8. https://stockfishchess.org/blog/2016/stockfish-8/, 2016. Accessed: 2026-10-27.

Stockfish developers. Introducing NNUE evaluation. Stockfish blog, August 2020. URL https://stockfishchess.org/blog/2020/introducing-nnue-evaluation/. Accessed: 2026-10-27.

Nishant Subramani, Nivedita Suresh, and Matthew Peters. Extracting latent steering vectors from pretrained language models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 566–581, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.48. URL https://aclanthology.org/2022.findings-acl.48/.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, second edition, 2018.

Mohammad Taufeeque, Aaron David Tucker, Adam Gleave, and Adrià Garriga-Alonso. Path channels and plan extension kernels: A mechanistic description of planning in a Sokoban RNN. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=aAshH4kQ1v.

TCEC. Top Chess Engine Championship season 29. https://tcec-chess.com/#div=p&game=1&season=29, 2026. Accessed: 2026-10-27.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. DeepNet: Scaling transformers to 1,000 layers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(10):6761–6774, October 2024. ISSN 0162-8828. doi: 10.1109/TPAMI.2024.3386927. URL https://doi.org/10.1109/TPAMI.2024.3386927.

Keyu Wang, Jin Li, Shu Yang, Zhuoran Zhang, and Di Wang. When truth is overridden: Uncovering the internal origins of sycophancy in large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does LLM safety training fail? In *Advances in Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=jA235JGM09.

Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. Jump to conclusions: Short-cutting transformers with linear transformations. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 9615–9625, Torino, Italia, May 2024. ELRA and ICCL. URL https://aclanthology.org/2024.lrec-main.840/.

Appendix Contents

A Limitations and future directions **17**

B Compute resources and licensing **18**

C Post-LN logit lens extension **18**

C.1 Pre- vs Post-LN architectures and DeepNorm 18

C.2 Zero ablation methodology for Post-LN architectures 19

C.3 Transformer encoder decomposition 19

C.4 Implications for the logit lens 20

D Model architecture and search **21**

D.1 Input encoding 21

D.2 Transformer encoder 22

D.3 Output heads 22

D.4 Search 23

E Intermediate policy dynamics **23**

F Complete tables for the tournament and Lichess play **26**

F.1 Internal tournament 26

F.2 Lichess bot performance 27

G Forgotten puzzles across Lc0 networks **27**

H Complete probabilities for the solved example puzzle from Figure 2 **29**

I Correctly solved puzzle case studies **32**

I.1 Knight fork and discovered attack 32

I.2 Early and consistent solution lock-in 35

I.3 Knight sacrifice with rook mate 38

I.4 Queen sacrifice for rook mate 41

I.5 Bishop sacrifice for discovered attack 44

J Complete probabilities for the forgotten example puzzle from Figure 1 **47**

K Forgotten puzzle case studies **51**

K.1 Rook sacrifice for queen capture 51

K.2 Pawn sacrifice for queen capture 55

K.3 Queen sacrifice to back-rank mate 59

K.4 Rook sacrifice for material gain 63

K.5 Narrowly forgotten queen sacrifice 66

L Look-ahead analysis details **70**

M Value head evaluation **72**

N Stockfish concept preference analysis **74**

O Concept steering details **84**

P Search experiments **93**

A Limitations and future directions

Scope of analysis Our causal analyses focus on a single chess model with a Post-LN architecture, though we verify that the forgotten puzzle phenomenon itself replicates across other Leela networks (Appendix G). The findings function as an existence proof: algorithmically computed solutions can be overridden by competing priors in a deployed neural network. However, this single-domain analysis cannot disentangle whether the override reflects domain-specific characteristics of chess, architecture-specific properties of Post-LN transformers, or more general principles of neural network computation. Comparison across architectures and domains, particularly Pre-LN language models, would help isolate these effects. No Pre-LN Leela models exist for direct architectural comparison; the development team has chosen Post-LN over Pre-LN in their training paradigm (dje-dev, 2024).

Models without search Can the override arise in models that are deployed without search? Its ingredients do not require search. Essentially every model trained today minimizes an expected loss over a data distribution with parameters shared across inputs, and so learns priors that are right in expectation. The override arises where an internal computation conflicts with such a prior and the two are reconciled late in the network. Related overrides have been reported in language models that use no search (Halawi et al., 2024; Wang et al., 2026). In Leela, the search that wraps the network at deployment corrects much of the override (Section 3.4); a model deployed without such a search has no correction, so the override reaches its output directly and may be more visible, not less.

Reasoning models and search Reasoning models do use a form of search, at least during training. There, the model explores the space of possible chains of thought (Chollet, 2024), and its training resembles Leela's. In AlphaZero and Leela, each node of the search tree is a position. For every move in a self-play game, MCTS builds a tree from the current position, and the policy at that position is trained toward the visit counts at the root. Reinforcement learning with verifiable rewards (Lambert et al., 2025; Guo et al., 2025) applies a similar idea with tokens as nodes. For each prompt, the model samples several complete chains of thought, each a single path through the tree of possible continuations, and checks the final answer. Tokens in chains that reach a correct answer are made more likely and tokens in chains that fail less likely. In both cases, the outcome of the search is distilled back into the network that guides the search as its prior, so that the network itself becomes more likely to produce what the search found. Training can therefore only learn from what the prior lets the search find. Every continuation lies in the support of a softmax policy, but the prior determines which are reached within a sampling budget in both Leela and reasoning models.

This view also gives one reading of why reinforcement learning on language models took as long as it did to work. If no sampled chain reaches the correct answer, the reward is uniform and there is nothing to upweight. The prior has to succeed occasionally before the process can bootstrap itself, which required pretrained language models strong enough to guide the search through the space of chains of thought. In Leela, the search overcomes the prior on forgotten puzzles, because MCTS backs up value estimates through the tree. Sampled chains have no such correction, so a correct chain is found only as often as the prior allows. A prior like Leela's safety prior would therefore rarely be corrected in a reasoning model, because training rarely sees a reward on the problems it gets wrong.

Generalization and applications While our analysis focuses on chess transformers, our methodology could inform interpretability research in other domains. First, concept preference analysis offers an alternative to representation probing: rather than identifying whether concepts are encoded, it

tracks which concepts each layer prioritizes in driving outputs. Applied to language models, this could trace behavioral tendencies such as truthfulness or specificity across layers. Second, our observation that safety priors may override algorithmic solutions suggests that in tasks with verifiable ground truth, suppressing learned distributional biases—such as final-layer tendencies toward common tokens—might improve performance on tasks requiring precise reasoning.

B Compute resources and licensing

Compute resources All experiments were conducted on a single workstation with 2× NVIDIA RTX 5090 (32 GB) and an AMD Threadripper PRO 7965WX (24 cores). Most individual experiments complete within a few hours on a single GPU. The most compute-intensive components are the internal tournaments (∼1–3 days due to sequential game play), concept delta computation via Stockfish (∼16 hours parallelized across CPU cores), the full steering sweep across all concepts (∼2 GPU-days), and the search experiments (∼half a day). All main paper experiments can be reproduced in approximately 2–3 days on comparable hardware.

Licenses and data availability The Lichess puzzle database is available under a Creative Commons CC0 license (Lichess.org). The CCRL dataset is publicly available without a stated license (Leela Chess Zero team, 2018). The puzzle dataset, Encyclopedia of Chess Openings positions, and tournament code from Ruoss et al. (2024) are released under the Apache 2.0 license, with the Encyclopedia of Chess Openings originally compiled by Matanović (1978). The inference framework from Jenner et al. (2024), Leela Chess Zero (Leela Chess Zero team), and Stockfish 8 and 17.1 (Stockfish developers) are all released under the GPLv3 license. The Lichess bots run on the lichess-bot framework (Lichess Bot Devs, 2025), released under the AGPLv3 license.

C Post-LN logit lens extension

The logit lens technique projects intermediate layer representations through the model's output head to examine what the network would predict at different depths. This approach relies on the assumption that representations across layers exist in a shared basis that allows meaningful projection to output space—an assumption enabled by residual connections that create additive pathways for information flow (Jastrzębski et al., 2018). In transformer architectures, architectural inductive biases further encourage information about specific tokens to remain localized to their corresponding positions throughout the network, creating a privileged basis that facilitates cross-layer interpretation (Jenner et al., 2024).

In Pre-LN transformers, this involves taking intermediate representations, applying the final layer normalization, and projecting through the unembedding matrix. This approach works because layer normalization precedes each sublayer in Pre-LN models, leaving the residual stream unchanged and maintaining representational consistency across layers.

Post-LN architectures complicate this process by applying layer normalization after each sublayer's output is added to the residual stream. Unlike Pre-LN models where only a final normalization is needed, Post-LN models have sequential normalization operations that directly transform the residual stream at each layer. These intermediate normalizations create dependencies between layers that prevent simply taking an intermediate representation and applying only the final layer normalization and output projection, as the intermediate representation has not undergone the normalization transformations it would experience in a complete forward pass.

Our goal is to develop an extension that maps intermediate layer representations to the representational basis expected by the policy head, accounting for the normalization transformations unique to Post-LN architectures.

C.1 Pre- vs Post-LN architectures and DeepNorm

The key difference between Pre-LN and Post-LN architectures lies in when layer normalization is applied relative to the residual connections. This placement affects how representations evolve through the network and impacts the applicability of interpretability techniques.

Pre-LN In Pre-LN transformers, layer normalization is applied before each sublayer (attention and feed-forward), with the computation following the pattern:

$$\mathbf{h}'_\ell = \mathbf{h}_{\ell-1} + \mathrm{MHA}_\ell(\mathrm{LayerNorm}(\mathbf{h}_{\ell-1})) \tag{3}$$
$$\mathbf{h}_\ell = \mathbf{h}'_\ell + \mathrm{FFN}_\ell(\mathrm{LayerNorm}(\mathbf{h}'_\ell)) \tag{4}$$

where $\mathbf{h}_\ell$ denotes the hidden representation at layer ℓ, MHA_ℓ is the multi-head attention operation, and FFN_ℓ is the feed-forward network at layer ℓ. This design ensures that the residual stream itself is never directly transformed by normalization operations, allowing intermediate representations to be projected through the final layer normalization and output head in a straightforward manner.

Post-LN Post-LN models apply layer normalization after adding sublayer outputs to the residual stream:

$$\mathbf{h}'_\ell = \mathrm{LayerNorm}(\mathbf{h}_{\ell-1} + \mathrm{MHA}_\ell(\mathbf{h}_{\ell-1})) \tag{5}$$
$$\mathbf{h}_\ell = \mathrm{LayerNorm}(\mathbf{h}'_\ell + \mathrm{FFN}_\ell(\mathbf{h}'_\ell)) \tag{6}$$

Each layer normalization operation directly transforms the accumulated representation, creating a sequence of transformations that intermediate representations must undergo to reach the final output space.

DeepNorm Leela employs DeepNorm (Wang et al., 2024) scaling to stabilize Post-LN training, modifying the computation to include residual scaling factors:

$$\mathbf{h}'_\ell = \mathrm{LayerNorm}(\alpha \cdot \mathbf{h}_{\ell-1} + \mathrm{MHA}_\ell(\mathbf{h}_{\ell-1})) \tag{7}$$
$$\mathbf{h}_\ell = \mathrm{LayerNorm}(\alpha \cdot \mathbf{h}'_\ell + \mathrm{FFN}_\ell(\mathbf{h}'_\ell)) \tag{8}$$

where $\alpha = (2N)^{1/4} \approx 2.34$ for $N = 15$ layers. This scaling, combined with specialized weight initialization, enables stable training while preserving Post-LN's representational advantages.

C.2 Zero ablation methodology for Post-LN architectures

The standard logit lens can be understood as performing zero ablation—setting all sublayer outputs to zero for layers beyond ℓ, then applying the final layer normalization and projection to output space. We extend this principle to Post-LN models by applying the same zero ablation of sublayer outputs while preserving the normalization operations and α that would transform these representations.

Specifically, for examining layer k, we set $\mathrm{MHA}_\ell(\cdot) = 0$ and $\mathrm{FFN}_\ell(\cdot) = 0$ for all $\ell > k$ during a forward pass, while preserving the α and layer normalization scaling parameters (γ). We set layer normalization biases (β) to zero for layers beyond k to maintain better consistency with the standard logit lens paradigm. The rationale for these choices will be explained in Appendix C.4, following the decomposition analysis.

This approach ensures that representations from layers 1 through k undergo the same sequence of transformations they would experience in the complete network, while removing contributions from later layers. Normalization statistics (μ and σ) are recomputed during this modified forward pass to reflect the actual distribution of the truncated representations, rather than using statistics computed on the full model output. This follows the same principle as Pre-LN logit lens implementations, which directly apply the final layer normalization to the intermediate activations being analyzed.

C.3 Transformer encoder decomposition

To provide intuition for why our proposed logit lens extension makes principled choices for Post-LN architectures, we present a decomposition of the transformer encoder output and reinterpret the logit lens in terms of this decomposition. Following Mickus et al. (2022), the final representation of the encoder at layer L and token position t can be decomposed into:

$$\mathbf{h}_{L,t} = \mathbf{i}_{L,t} + \mathbf{z}_{L,t}^{\mathrm{MHA}} + \mathbf{z}_{L,t}^{\mathrm{FFN}} + \mathbf{b}_{L,t} - \mathbf{m}_{L,t} \tag{9}$$

$$\mathbf{i}_{L,t} = \alpha^{2L} \cdot \frac{\bigodot_{\ell=1}^{L} \gamma_{\ell}^{\mathrm{MHA}} \odot \gamma_{\ell}^{\mathrm{FFN}}}{\prod_{\ell=1}^{L} \sigma_{\ell,t}^{\mathrm{MHA}} \sigma_{\ell,t}^{\mathrm{FFN}}} \odot \mathbf{h}_{0,t} \tag{10}$$

$$\mathbf{z}_{L,t}^{\mathrm{MHA}} = \sum_{\ell=1}^{L} \alpha^{2L-2\ell+1} \cdot \frac{\bigodot_{\ell'=\ell}^{L} \gamma_{\ell'}^{\mathrm{MHA}} \odot \gamma_{\ell'}^{\mathrm{FFN}}}{\prod_{\ell'=\ell}^{L} \sigma_{\ell',t}^{\mathrm{MHA}} \sigma_{\ell',t}^{\mathrm{FFN}}} \odot \tilde{\mathbf{z}}_{\ell,t}^{\mathrm{MHA}} \tag{11}$$

$$\mathbf{z}_{L,t}^{\mathrm{FFN}} = \sum_{\ell=1}^{L} \alpha^{2L-2\ell} \cdot \frac{\gamma_{\ell}^{\mathrm{FFN}} \bigodot_{\ell'=\ell+1}^{L} \gamma_{\ell'}^{\mathrm{MHA}} \odot \gamma_{\ell'}^{\mathrm{FFN}}}{\sigma_{\ell,t}^{\mathrm{FFN}} \prod_{\ell'=\ell+1}^{L} \sigma_{\ell',t}^{\mathrm{MHA}} \sigma_{\ell',t}^{\mathrm{FFN}}} \odot \tilde{\mathbf{z}}_{\ell,t}^{\mathrm{FFN}} \tag{12}$$

$$\begin{aligned}
\mathbf{b}_{L,t} = &\sum_{\ell=1}^{L} \alpha^{2L-2\ell+1} \cdot \frac{\bigodot_{\ell'=\ell}^{L} \gamma_{\ell'}^{\mathrm{MHA}} \odot \gamma_{\ell'}^{\mathrm{FFN}}}{\prod_{\ell'=\ell}^{L} \sigma_{\ell',t}^{\mathrm{MHA}} \sigma_{\ell',t}^{\mathrm{FFN}}} \odot b_{\ell}^{\mathrm{MHA},V} W_{\ell}^{\mathrm{MHA},O} \\
&+ \sum_{\ell=1}^{L} \alpha^{2L-2\ell+1} \cdot \frac{\bigodot_{\ell'=\ell}^{L} \gamma_{\ell'}^{\mathrm{MHA}} \odot \gamma_{\ell'}^{\mathrm{FFN}}}{\prod_{\ell'=\ell}^{L} \sigma_{\ell',t}^{\mathrm{MHA}} \sigma_{\ell',t}^{\mathrm{FFN}}} \odot b_{\ell}^{\mathrm{MHA},O} \\
&+ \sum_{\ell=1}^{L} \alpha^{2L-2\ell} \cdot \frac{\gamma_{\ell}^{\mathrm{FFN}} \bigodot_{\ell'=\ell+1}^{L} \gamma_{\ell'}^{\mathrm{MHA}} \odot \gamma_{\ell'}^{\mathrm{FFN}}}{\sigma_{\ell,t}^{\mathrm{FFN}} \prod_{\ell'=\ell+1}^{L} \sigma_{\ell',t}^{\mathrm{MHA}} \sigma_{\ell',t}^{\mathrm{FFN}}} \odot b_{\ell}^{\mathrm{FFN},O} \\
&+ \sum_{\ell=1}^{L} \alpha^{2L-2\ell+1} \cdot \frac{\gamma_{\ell}^{\mathrm{FFN}} \bigodot_{\ell'=\ell+1}^{L} \gamma_{\ell'}^{\mathrm{MHA}} \odot \gamma_{\ell'}^{\mathrm{FFN}}}{\sigma_{\ell,t}^{\mathrm{FFN}} \prod_{\ell'=\ell+1}^{L} \sigma_{\ell',t}^{\mathrm{MHA}} \sigma_{\ell',t}^{\mathrm{FFN}}} \odot \beta_{\ell}^{\mathrm{MHA}} \\
&+ \sum_{\ell=1}^{L} \alpha^{2L-2\ell} \cdot \frac{\bigodot_{\ell'=\ell+1}^{L} \gamma_{\ell'}^{\mathrm{MHA}} \odot \gamma_{\ell'}^{\mathrm{FFN}}}{\prod_{\ell'=\ell+1}^{L} \sigma_{\ell',t}^{\mathrm{MHA}} \sigma_{\ell',t}^{\mathrm{FFN}}} \odot \beta_{\ell}^{\mathrm{FFN}}
\end{aligned} \tag{13}$$

$$\begin{aligned}
\mathbf{m}_{L,t} = &\sum_{\ell=1}^{L} \alpha^{2L-2\ell+1} \cdot \frac{\bigodot_{\ell'=\ell}^{L} \gamma_{\ell'}^{\mathrm{MHA}} \odot \gamma_{\ell'}^{\mathrm{FFN}}}{\prod_{\ell'=\ell}^{L} \sigma_{\ell',t}^{\mathrm{MHA}} \sigma_{\ell',t}^{\mathrm{FFN}}} \odot \mu_{\ell,t}^{\mathrm{MHA}} \mathbf{1} \\
&+ \sum_{\ell=1}^{L} \alpha^{2L-2\ell} \cdot \frac{\gamma_{\ell}^{\mathrm{FFN}} \bigodot_{\ell'=\ell+1}^{L} \gamma_{\ell'}^{\mathrm{MHA}} \odot \gamma_{\ell'}^{\mathrm{FFN}}}{\sigma_{\ell,t}^{\mathrm{FFN}} \prod_{\ell'=\ell+1}^{L} \sigma_{\ell',t}^{\mathrm{MHA}} \sigma_{\ell',t}^{\mathrm{FFN}}} \odot \mu_{\ell,t}^{\mathrm{FFN}} \mathbf{1}
\end{aligned} \tag{14}$$

where $\mathbf{h}_{0,t}$ is the initial input embedding, $\tilde{\mathbf{z}}_{\ell,t}^{\mathrm{MHA}}$ and $\tilde{\mathbf{z}}_{\ell,t}^{\mathrm{FFN}}$ are the raw unbiased outputs from multi-head attention and feed-forward networks at layer ℓ, and $\alpha = (2N)^{1/4}$ is the DeepNorm scaling factor. The bias terms include $b_{\ell}^{\mathrm{MHA},V}$ (value projection bias), $W_{\ell}^{\mathrm{MHA},O}$ (output projection matrix), $b_{\ell}^{\mathrm{MHA},O}$ (attention output bias), and $b_{\ell}^{\mathrm{FFN},O}$ (feed-forward output bias). Layer normalization parameters are γ_{ℓ} (learned scales), $\sigma_{\ell,t}$ (computed standard deviations), $\mu_{\ell,t}$ (computed means), and β_{ℓ} (learned biases), with superscripts indicating MHA or FFN sublayers. The notation $\bigodot$ denotes element-wise multiplication.

The decomposition separates the final representation into the transformed input embedding ($\mathbf{i}_{L,t}$), accumulated sublayer contributions ($\mathbf{z}_{L,t}^{\mathrm{MHA}}$, $\mathbf{z}_{L,t}^{\mathrm{FFN}}$), bias terms ($\mathbf{b}_{L,t}$), and mean centering effects ($\mathbf{m}_{L,t}$). Each component is scaled by normalization parameters from subsequent layers.

C.4 Implications for the logit lens

The decomposition framework provides direct justification for our zero ablation approach. The Post-LN logit lens can be understood as truncating the summations in $\mathbf{z}_{L,t}^{\mathrm{MHA}}$, $\mathbf{z}_{L,t}^{\mathrm{FFN}}$, and $\mathbf{b}_{L,t}$ to include only layers $\ell \leq k$, while recomputing the normalization statistics (μ and σ) based on the modified representations.

Preservation of scaling parameters The decomposition reveals why preserving the γ and α scaling factors is essential: all terms in the truncated representation—including the input embedding $\mathbf{i}_{L,t}$ and retained sublayer contributions—are transformed by these parameters from subsequent layers. Removing these transformations would fundamentally alter how the truncated representation maps

to the output space of the encoder, although under our zero-ablation configuration the α factor is absorbed by layer normalization and has no numerical effect.

Treatment of bias terms The bias terms β from layer normalization appear in $\mathbf{b}_{L,t}$ alongside attention and feed-forward biases, indicating they are structurally equivalent. For consistency with the zero ablation principle of removing sublayer contributions beyond layer k, we ablate all bias terms including layer normalization biases. Empirically, preserving versus ablating these biases produces qualitatively similar results; for example, preserving the layer normalization biases changes the solve rate by at most 2.7 percentage points at any layer and leaves the forgotten set nearly unchanged. We provide code to examine all configurations. Our approach uses the policy head unchanged, which contains its own bias terms that may contribute to biased outputs. Moreover, the decomposition only captures bias terms that can be linearly separated from the representation—biases that interact with activation functions cannot be decomposed into constant additive components.

Representational alignment versus learned priors The treatment of bias terms reflects broader questions about their computational role. Bias terms may serve as representational alignment mechanisms that bridge coordinate system differences between layers, or they may encode learned priors about the task domain. The Tuned Lens (Belrose et al., 2023) learns affine transformations with bias terms to achieve representational alignment between intermediate and final layers in Pre-LN architectures, motivated by biased outputs observed with the standard logit lens. In practice, bias terms likely serve both functions simultaneously, and disentangling them remains an open problem.

Learned decoders The Tuned Lens and linear shortcuts (Yom Din et al., 2024) are trained to reproduce the model's final layer, so their objective penalizes exactly the disagreement with the final output that defines forgotten puzzles. Our zero-ablation readout has no such target and adds no trained parameters.

D Model architecture and search

We analyze the `T82-768x15x24h` transformer model from Leela Chess Zero, the network studied by Jenner et al. (2024), using their inference framework for our experiments. This model has 15 transformer layers, 768-dimensional representations, 24 attention heads, and approximately 95 million parameters, and follows the transformer architecture developed by the Leela Chess Zero team (Monroe et al., 2026). It processes chess positions through three stages: input encoding transforms board states into token representations (with each of the 64 squares treated as a discrete token), a 15-layer transformer encoder processes these representations, and task-specific output heads generate move probabilities, position evaluations, and game length predictions. We use the original model that incorporates historical board information for all experiments unless stated otherwise, as this represents the model deployed in practice rather than the variant fine-tuned for interpretability research by Jenner et al. (2024). Besides the current position, the model's input encodes the seven preceding positions of the game. Since all puzzles in the Lichess database are taken from real games, this history is available. To obtain it, we download each puzzle's source game from Lichess (Lichess.org) and replay it up to the puzzle position. Positions sampled from CCRL (Leela Chess Zero team, 2018) likewise come with their full games. The only inputs without a history of their own are the corrupted positions used for activation patching (Appendix L). While minor differences exist on individual positions, the fine-tuned version is generally trained to behave equivalently, and both exhibit the forgotten puzzle phenomenon (Appendix G).

D.1 Input encoding

Leela's input encoding transforms chess positions through binary feature planes, chess-specific positional encodings, and learned projections into the model's embedding space.

Board representation The board state is encoded using 112 binary feature planes of size 8×8, with the first 12 planes representing current piece positions for each piece type and color, and one plane marking whether the position is a repetition. Historical context is incorporated through the 7 previous board positions ($91 = 7 \times 13$ planes) following the same procedure, plus auxiliary planes encoding castling rights (4 planes), side to move (1 plane), fifty-move clock (1 plane holding the

number of half-moves since the last capture or pawn move), and two constant planes (0s and 1s) that are architectural remnants from CNNs without functional meaning.

Positional encoding Leela employs chess-specific positional encodings that capture movement relationships between squares in a 64×64 matrix. Each square receives a 64-dimensional vector where position (i, j) is set to 1 if any piece could legally move from square i to square j in one move regardless of current board state, 0 otherwise, and -1 for diagonal entries $(i = j)$ to distinguish self-reference.

Input preparation The 112 feature planes are reshaped from $112 \times 8 \times 8$ to 64×112, where each of the 64 entries corresponds to a board square with the 112-dimensional feature vector for that square concatenated with its corresponding 64-dimensional positional encoding vector, forming a 64×176 tensor. This combined representation undergoes a linear transformation with Mish activation (Misra, 2020), followed by elementwise scaling and shifting operations, producing the final 64×768 input for the transformer layers. This enriched preprocessing was motivated by observations that early attention layers contributed minimally to performance (Monroe, 2024). While no information mixing between tokens occurs here, this involves substantially more processing than typical language models (which simply use embedding matrices plus positional encodings), potentially explaining why our logit lens mappings yield meaningful results even before the first transformer layer.

D.2 Transformer encoder

The transformer encoder consists of 15 identical layers with a model dimensionality of 768, processing 64 tokens (one per board square) through multi-head attention with Smolgen enhancement, feed-forward networks, and Post-LN normalization with DeepNorm scaling. Unlike autoregressive models, all tokens can attend to each other bidirectionally.

Attention with Smolgen Each layer uses 24 attention heads with 32-dimensional head size, applying scaled dot-product attention to the 64 board square tokens. The Smolgen module enhances standard self-attention by enabling attention scores to depend not only on individual square contents but also on the global board position. It compresses all square representations into a global vector, processes this through MLPs, then generates supplementary attention logits of shape $24 \times 64 \times 64$ that are added to the standard query-key dot products before softmax normalization.

Feed-forward Each FFN layer expands from 768 to 1024 dimensions with squared ReLU activation. Unlike other domains that benefit from $4\times$ expansion ratios, chess models show little improvement from larger feed-forward networks (Monroe, 2024).

Post-LN with DeepNorm The model employs a Post-LN architecture, the original design used in early transformers (Vaswani et al., 2017), applying layer normalization after each sublayer (attention and feed-forward) within the residual stream rather than before sublayers as in modern Pre-LN variants, following empirical comparisons that demonstrated superior performance (dje-dev, 2024). To mitigate vanishing gradients, the model uses DeepNorm scaling (Wang et al., 2024), which applies a constant upscaling factor to residual connections and uses specialized weight initialization. See Appendix C.1 for complete equations.

D.3 Output heads

The transformer encoder's final 64×768 representations are processed by three specialized heads: the policy head generates move probability distributions, the value head predicts win/draw/loss outcomes, and the moves-left head estimates remaining game length.

Policy head The policy head first processes each square's 768-dimensional representation through a shared MLP with Mish activation. These processed representations are then transformed via two separate linear projections: one creating "source" representations for squares where moves originate, and another creating "target" representations for destination squares. The source and target representations are matrix-multiplied along the 768-dimensional axis, producing a 64×64 matrix where each entry contains a scalar logit representing the likelihood of a move from the

corresponding source square to target square. Promotion moves require additional processing through two specialized branches that handle move selection and promotion-type preferences separately. The final output combines standard move logits with promotion logits, which are then filtered to extract only legal moves for the current position before applying softmax normalization.

Value head The value head processes each square's representation through an MLP with Mish activation, reducing them to 32 dimensions per square. These 64 representations are flattened into a single 2048-dimensional vector to enable global position assessment. A second MLP layer compresses this to 128 dimensions, followed by a linear projection to 3 logits, which are then converted to win, draw, and loss outcome probabilities via softmax normalization. Unlike AlphaZero's single scalar output, this three-way classification provides more nuanced position evaluation.

Moves-left head The moves-left head predicts the number of moves remaining until game termination. Each square's representation is processed through an MLP, then flattened into a global vector for position-level assessment. Two additional MLP layers progressively reduce the dimensionality to produce a final scalar output estimating remaining game length. The output layer uses Mish activation because ReLU would provide zero gradients when pre-activation values are negative during training, even though the target output is always positive.

D.4 Search

Leela's search (Leela Chess Zero team) is a variant of Monte Carlo tree search that follows AlphaZero (Silver et al., 2018). For every legal move a in a position s of the game tree, it stores the prior $P(s, a)$ given by the policy head, the number of visits $N(s, a)$ and the mean value $Q(s, a)$ of all evaluations below the move. Each iteration starts at the root and, at every node, follows the move that maximizes

$$Q(s,a) + c\,P(s,a)\,\frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}, \tag{15}$$

where c grows slowly with the number of visits. The search thus favors moves with a high value and, among moves with few visits, those with a high prior. Moves that have not been visited yet enter this rule with a first-play-urgency value in place of Q. Unlike AlphaZero, Leela can also add a small, optional term from its moves-left head, which predicts the remaining length of the game, so that shorter wins are preferred.

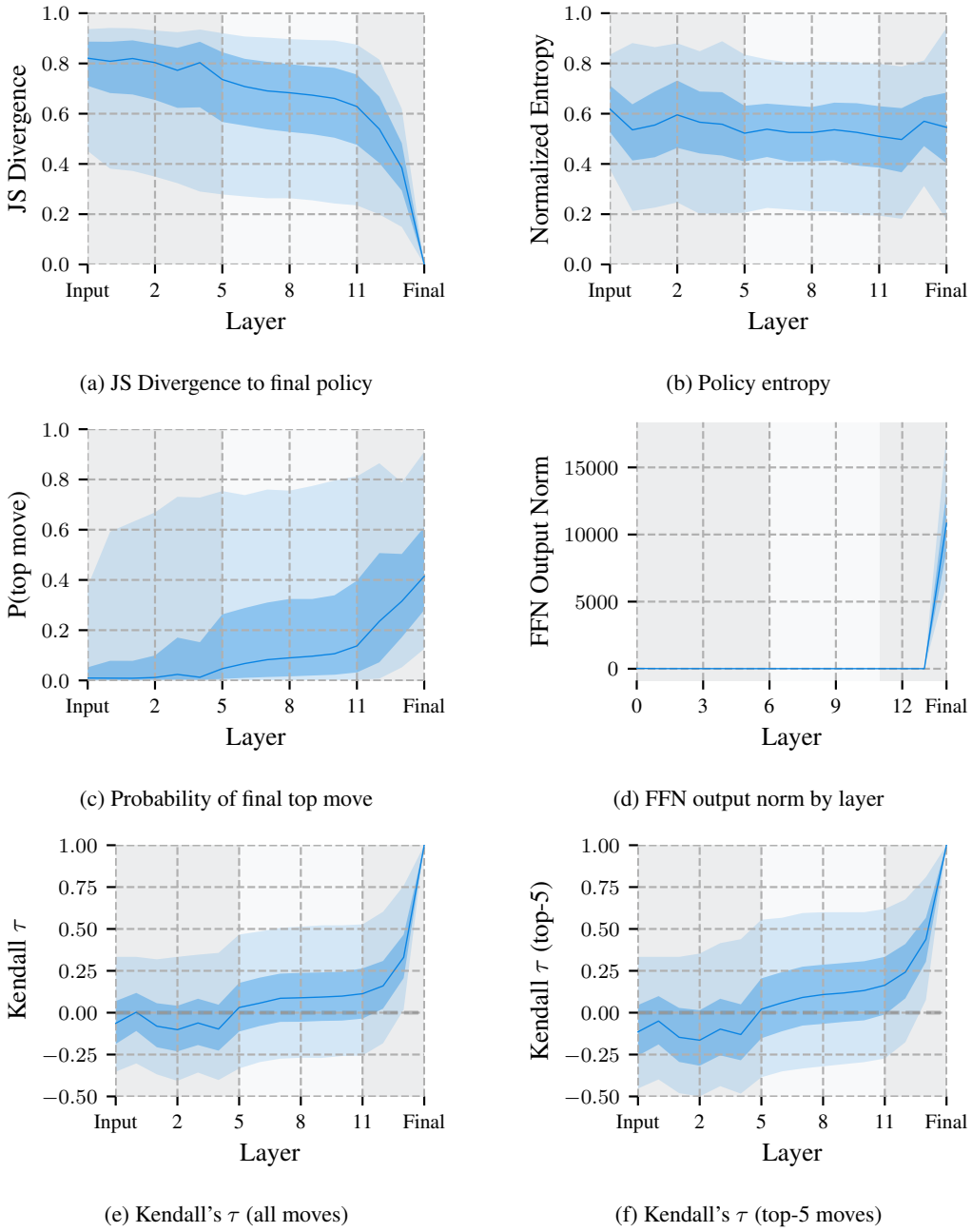
When an iteration reaches a position that has not been evaluated, the network evaluates it once. The policy head gives the priors of the new position's moves, with an optional temperature applied to its logits, and the value head gives the probabilities of a win, a draw and a loss. The win probability minus the loss probability is passed back along the path and averaged into Q at every node, with its sign flipped at each ply. Checkmates, stalemates and draws by rule are scored exactly, without the network. All of this holds in both self-play training and deployment. Our analyses of the policy network use the plain softmax of its logits, without the search temperature.

In self-play training games, Dirichlet noise is added to the priors at the root to encourage exploration, and the search stops once further visits barely change the distribution of visits at the root, measured by the gain in KL divergence per visit. These normalized visit counts are the training target of the policy head, and the move played in the game is sampled from them with a temperature that decreases over the first moves of the game to introduce opening diversity.

In deployment, no noise is added, the search runs for a given budget of nodes or time, and smart pruning stops it early once the most visited move can no longer be overtaken. The engine then plays the most visited move.

E Intermediate policy dynamics

To provide a deeper, quantitative view of the model's inference process, we computed several policy metrics across the network's depth. The following analyses were conducted on a sample of 10,000 positions from the CCRL dataset (Leela Chess Zero team, 2018). Figure 8 presents five metrics that characterize the policy's evolution. We plot the median, the 25th–75th percentile range, and the 5th–95th percentile range and indicate the proposed phases as shaded background.



(a) JS Divergence to final policy

(b) Policy entropy

(c) Probability of final top move

(d) FFN output norm by layer

(e) Kendall's τ (all moves)

(f) Kendall's τ (top-5 moves)

Figure 8: Policy dynamics across network depth on 10,000 CCRL positions. Lines show medians; inner and outer shaded regions represent the 25th–75th and 5th–95th percentile ranges. Background shading indicates the three computational phases.

Jensen-Shannon divergence The Jensen-Shannon divergence shows a median that remains high until the final layers, with the 5th percentile being much lower. This shows that a small subset of positions converge early to the final policy, while the vast majority have different move preferences than the final output throughout most layers. The phase structure is again apparent, with higher variability and erratic changes in early layers, a more stable plateau in the middle phase, and a sharp decline in the final phase. In this metric, the transition to the final phase appears to occur slightly later—around layer 12 rather than 11—though consistent with the overall three-phase progression described in the main text.

Entropy The policy entropy shows a relatively stable median with modest variation across layers, indicating that overall certainty changes only slightly with depth. The 95th and 5th percentiles are very high and very low respectively, indicating that for any layer there are positions where the layer is either very certain or very uncertain about what moves are good. The phase structure is again apparent, with more erratic fluctuations in the early layers, a stable middle phase, and increased variability toward the end.

Probability of final top move The median probability assigned to the final top move is near zero for the first layers, then grows approximately linearly with a steeper increase in the final layers. The outliers are very high (between 0.6 and 0.8), indicating that all layers correctly predict the top move for some positions with high probability, having already converged. The phase structure is again visible, with slightly more stable behavior in the middle layers and a pronounced acceleration in the final phase, which here again appears to begin around layer 12.

FFN output norms Following Lad et al. (2025), we examine the L2 norm of each transformer FFN sublayer's output during a standard forward pass to probe for their proposed residual sharpening mechanism. While the relatively constant policy entropy suggests no sharpening at the distributional level, the FFN output norms exhibit an extreme last-layer increase, suggesting a feature consolidation process analogous to that observed in language models. This is consistent with the final feed-forward network playing a dominant role in reshaping the policy, as observed in the forgotten puzzle override.

Ranking correlation (Kendall's τ) To analyze the stability of move preferences, we compute Kendall's τ rank correlation between intermediate and final policies. We calculate this metric in two ways: first using all legal moves, and second, to mitigate noise from low-probability moves that are never seriously considered, only over moves that appear in the top five at any layer. Both approaches yield nearly identical results, with the correlation being negative in early layers and remaining low until around layer 12, where it increases sharply. The 5th and 95th percentiles in the top-5 moves analysis show that there are positions where intermediate layer move rankings are either very similar or very dissimilar to the final ranking until late in the network. The phase structure is again apparent, with high variability in early layers, relatively stable correlations across the middle phase up to layer 11, and a sharp rise beginning around layer 12, consistent with the transition to the late phase.

F Complete tables for the tournament and Lichess play

F.1 Internal tournament

We evaluate playing strength through internal round-robin tournaments processed with BayesElo (Coulom, 2008) using the default confidence parameter of 0.5. Each model pairing played 200 distinct openings from the Encyclopedia of Chess Openings (Matanović, 1978), with one game per side at deterministic temperature $\tau = 0$ and five games per side at stochastic temperature $\tau = 1.0$. Games are adjudicated by Stockfish at 0.01 seconds per move, which ends a game once its evaluation exceeds 1,300 centipawns in either direction. Tables 2 and 3 report the resulting Elo ratings, computed from all match outcomes. Each table lists the model's Elo estimate with asymmetric confidence bounds, total score percentage (wins plus half draws), average opponent rating, and draw rate. The Lc0 anchor is evaluated under argmax in both regimes following Ruoss et al. (2024); in the $\tau = 1$ tournament it therefore wins all but 16 of its 32,000 games against the stochastic field, so the absolute Elo scale is only loosely determined, while relative ordering and gaps between the layers remain interpretable. The results show consistent improvement in playing strength with network depth, with clear early- and late-layer acceleration in both deterministic and stochastic settings, while middle layers exhibit a prolonged performance plateau.

Table 2: Playing strength evaluation through internal tournament ($\tau = 0$)

Rank	Model	Elo	+	−	Games	Score (%)	Avg. Oppo.	Draws (%)
1	Lc0 Policy Net (Anchor)	2292	25	25	6400	97	1043	2
2	Full Model	2268	25	25	6400	96	1044	3
3	Logit Lens Layer 13	1670	22	21	6400	85	1081	2
4	Logit Lens Layer 12	1356	13	12	6400	74	1101	5
5	Logit Lens Layer 11	1117	10	10	6400	58	1116	7
6	Logit Lens Layer 10	1082	10	10	6400	55	1118	11
7	Logit Lens Layer 9	1055	10	10	6400	53	1120	12
8	Logit Lens Layer 8	1030	10	10	6400	51	1121	11
9	Logit Lens Layer 7	1028	10	10	6400	51	1122	12
10	Logit Lens Layer 6	997	10	10	6400	48	1124	12
11	Logit Lens Layer 5	993	9	9	6400	48	1124	17
12	Logit Lens Layer 4	908	9	9	6400	40	1129	18
13	Logit Lens Layer 3	832	9	9	6400	32	1134	16
14	Logit Lens Layer 2	730	10	10	6400	23	1140	16
15	Logit Lens Layer 1	655	11	10	6400	17	1145	17
16	Logit Lens Layer 0	609	11	12	6400	16	1148	8
17	Logit Lens Input	351	17	17	6400	4	1164	4

Table 3: Playing strength evaluation through internal tournament ($\tau = 1$)

Rank	Model	Elo	+	−	Games	Score (%)	Avg. Oppo.	Draws (%)
1	Lc0 Policy Net (Anchor)	2292	–	–	32000	100	749	0
2	Full Model	1367	8	8	32000	88	807	1
3	Logit Lens Layer 13	1119	6	6	32000	76	822	2
4	Logit Lens Layer 12	1065	5	5	32000	73	826	3
5	Logit Lens Layer 11	876	5	5	32000	58	837	5
6	Logit Lens Layer 10	872	5	5	32000	57	838	5
7	Logit Lens Layer 9	851	5	5	32000	55	839	5
8	Logit Lens Layer 7	816	5	5	32000	52	841	6
9	Logit Lens Layer 8	813	5	5	32000	52	841	6
10	Logit Lens Layer 5	810	5	5	32000	52	842	6
11	Logit Lens Layer 6	799	5	5	32000	51	842	5
12	Logit Lens Layer 4	718	5	5	32000	43	847	7
13	Logit Lens Layer 3	585	5	5	32000	31	856	7
14	Logit Lens Layer 2	469	6	6	32000	22	863	7
15	Logit Lens Layer 1	391	6	6	32000	17	868	5
16	Logit Lens Layer 0	387	6	6	32000	17	868	4
17	Logit Lens Input	46	9	10	32000	4	889	0

F.2 Lichess bot performance

To validate our internal tournament findings in a real-world setting, we deployed layer-wise policies as bots on Lichess (Lichess.org) using the bot API framework (Lichess Bot Devs, 2025). The bots sample from the policy for the first five moves to introduce opening diversity and then play the top move. They participated in Bullet (1+0, 2+1), Blitz (3+0, 3+2, 5+0, 5+3), and Rapid (10+0, 10+5, 15+10) time controls. Ratings and game counts were recorded on 28 October 2026. At the time of writing, the bots are still playing on Lichess, so the linked profiles may show more recent numbers.

Several experimental constraints shaped the evaluation: (1) bots played exclusively against other bots per Lichess policy, (2) the limited pool of weak bots meant repeated matchups for early layers, and (3) deterministic play after the opening often led to repetitive games. Despite these limitations, the results corroborate our internal tournament findings, showing progressive improvement across layers with the most significant transitions occurring at similar points in the network depth. Table 4 presents the playing strength and performance statistics for our layer-wise Lichess bots, with bot names linked to their respective Lichess profiles.

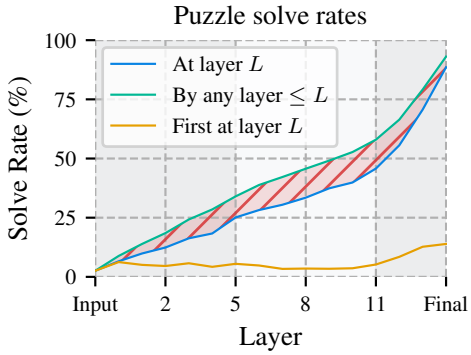
Table 4: Performance of layer-wise policies on Lichess across different time controls

Bot	**Rating**			**Total**	**Performance**		
	Bullet	**Blitz**	**Rapid**	**Games**	**W**	**D**	**L**
LLLBot-In	817 ± 45	545 ± 45	492 ± 45	2190	19	528	1643
LLLBot-0	843 ± 45	631 ± 45	617 ± 45	2023	30	478	1515
LLLBot-1	907 ± 45	611 ± 45	569 ± 45	1929	29	456	1444
LLLBot-2	930 ± 45	655 ± 45	669 ± 45	1820	43	478	1299
LLLBot-3	961 ± 45	740 ± 45	789 ± 45	1799	43	454	1302
LLLBot-4	998 ± 45	723 ± 45	791 ± 45	1689	51	494	1144
LLLBot-5	1071 ± 45	815 ± 45	871 ± 45	1634	105	388	1141
LLLBot-6	1085 ± 45	815 ± 45	907 ± 45	1529	131	326	1072
LLLBot-7	1087 ± 45	827 ± 45	954 ± 45	2135	194	537	1404
LLLBot-8	1096 ± 45	906 ± 45	954 ± 45	2004	194	502	1308
LLLBot-9	1094 ± 45	952 ± 45	985 ± 45	1849	208	446	1195
LLLBot-10	1124 ± 45	860 ± 45	1008 ± 45	1805	210	408	1187
LLLBot-11	1159 ± 45	946 ± 45	1072 ± 45	1652	262	297	1093
LLLBot-12	1399 ± 45	1205 ± 45	1324 ± 45	1529	468	282	779
LLLBot-13	1622 ± 45	1482 ± 45	1637 ± 45	1321	659	162	500
LLLBot-Full	2306 ± 45	2099 ± 45	2237 ± 45	959	521	116	322

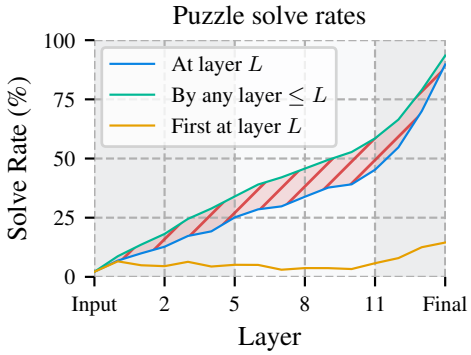
G Forgotten puzzles across Lc0 networks

To verify that the forgotten puzzle phenomenon is not specific to the T82-768x15x24h network analyzed in the main paper, we apply the same logit lens to four additional Lc0 checkpoints from the publicly available Lc0 network repository, spanning three width classes (256, 512, 768), two depths (10 and 15 encoder blocks), and multiple training generations.

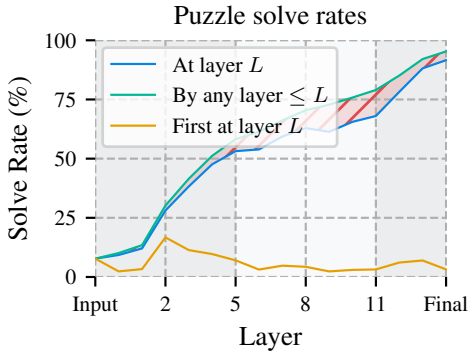
Figure 9 reports puzzle solve rates across layers for each network, evaluated on the general dataset consisting of 10,000 puzzles. The **forgotten** puzzle band—the gap between **solved by any layer** $\leq L$ and **solved at** L—is visible in every network, indicating that solution forgetting is a general property rather than a single-checkpoint artifact.



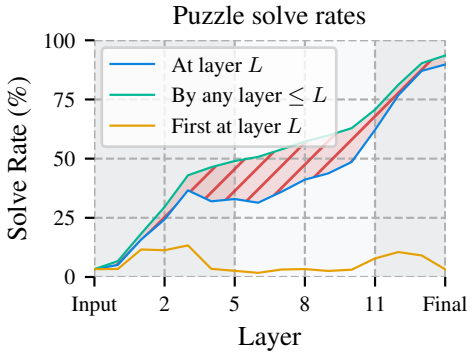
(a) T82 (768x15x24h, the model analyzed in this paper)



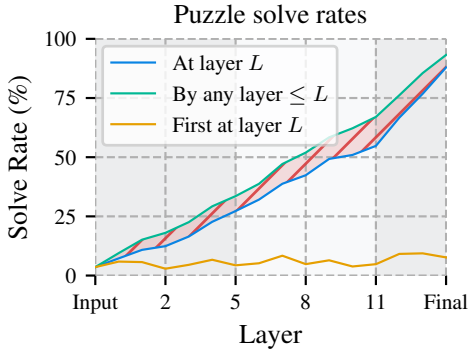
(b) T82 no-history fine-tune (768x15x24h, from Jenner et al., 2024)



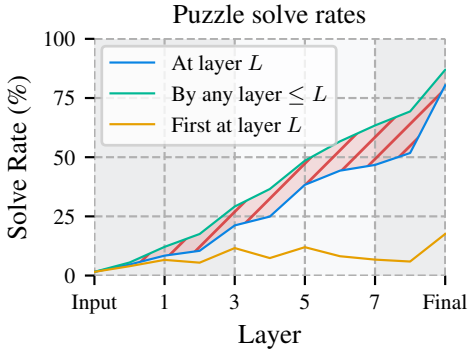
(c) BT3 (768x15x24h)



(d) T3 (512x15x16h)



(e) T1 (512x15x8h)



(f) T1 (256x10, only 10 encoder blocks)

Figure 9: Layer-wise puzzle solve rates for six Lc0 networks of varying width, depth, and training generation. The hatched **forgotten puzzle** band between **solved by any layer** $\leq L$ and **solved at** L is visible in every model.

H Complete probabilities for the solved example puzzle from Figure 2

This puzzle (Figure 10) features a knight sacrifice leading to mate (PV: 1. . . . ♘**g3+** 2. **h×g3** ♖**h6**#). The input layer exhibits piece-specific bias toward queen moves, with multiple captures dominating early probabilities. The winning move ♘**g3+** first becomes the top choice at layer 6 (26.7%) but is then briefly overtaken by ♕**g1+**, which dominates layers 9–10. This temporary preference for ♕**g1+** likely reflects a learned heuristic that queen checks are tactically important, even though this particular check ultimately loses the queen without compensation. After layer 10, ♘**g3+** regains its position as the preferred move, with the exception of layer 12 where a rook lift ♖**h6** (51.7%) briefly becomes the top candidate, potentially pinning the h-pawn and threatening the king along the h-file. The winning move's probability follows a non-monotonic trajectory, fluctuating throughout the layers before ultimately surging to 49.2% at layer 13 and 87.5% in the final output. This contrasts with some of the other cases in Appendix I, where the correct move is either identified immediately after early layers or remains unconsidered until the very late layers. Full probabilities are in Tables 5 and 6.

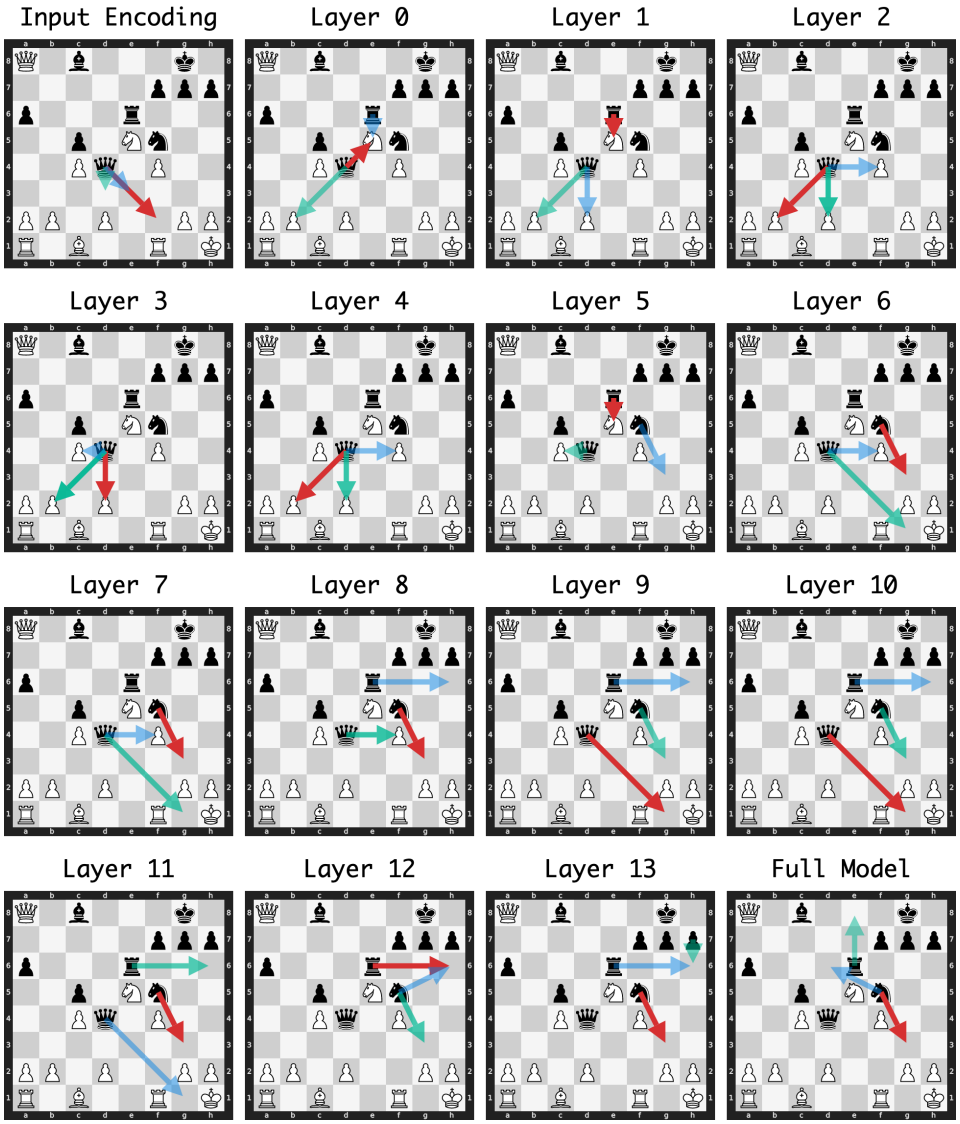


Figure 10: Layer-wise policy evolution for puzzle 62FeU from Figure 2.

Table 5: Move probabilities by layer for puzzle 62FeU (Part 1: Input to Layer 6)

Move	Input	Layer 0	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
♘g3+	6.54%	0.09%	0.14%	1.79%	7.59%	0.41%	15.23%	26.71%
♖h6	0.01%	0.03%	0.04%	0.13%	0.12%	0.15%	0.02%	2.56%
♕g1+	9.16%	1.11%	4.36%	7.52%	6.52%	0.43%	1.30%	22.74%
♕×b2	2.80%	17.91%	19.80%	24.22%	24.28%	27.55%	13.09%	0.96%
♕×d2	6.94%	15.90%	19.44%	21.33%	26.00%	21.59%	9.50%	0.66%
♖×e5	0.07%	16.82%	21.35%	12.49%	5.78%	9.91%	24.47%	10.04%
♕f2	24.04%	0.17%	0.33%	0.12%	0.14%	0.26%	0.09%	0.05%
♕×e5	0.03%	23.35%	18.84%	1.76%	0.63%	0.52%	0.72%	0.26%
♕×f4	0.49%	6.44%	6.63%	13.56%	11.31%	13.92%	13.76%	20.34%
♕×c4	0.80%	9.96%	6.49%	9.72%	11.84%	11.95%	16.31%	10.29%
h6	0.01%	1.04%	0.02%	0.01%	0.00%	0.02%	0.01%	0.00%
♕d3	12.24%	0.07%	0.12%	0.04%	0.09%	0.04%	0.08%	0.08%
♕e3	11.37%	0.05%	0.07%	0.58%	0.16%	0.31%	0.26%	0.56%
♕c3	10.54%	0.04%	0.09%	2.37%	1.38%	1.16%	0.97%	0.70%
♘e3	10.26%	0.09%	0.18%	1.02%	0.40%	0.44%	0.21%	0.07%
♘h6	0.13%	0.24%	0.03%	0.08%	0.01%	0.07%	0.01%	1.01%
♖c6	0.12%	1.51%	0.77%	0.66%	0.67%	4.08%	1.47%	0.87%
h5	0.05%	2.57%	0.11%	0.04%	0.02%	0.03%	0.03%	0.02%
♔h8	0.15%	0.05%	0.02%	0.22%	0.81%	2.20%	0.34%	0.24%
♘d6	0.08%	0.01%	0.02%	0.07%	0.14%	0.08%	0.09%	0.10%
♕e4	1.62%	0.09%	0.13%	0.06%	0.06%	0.07%	0.10%	0.13%
♖b6	0.07%	0.01%	0.01%	0.08%	0.32%	1.36%	0.30%	0.18%
♖e8	0.06%	0.60%	0.22%	0.17%	0.10%	0.06%	0.02%	0.03%
f6	0.07%	0.06%	0.00%	0.01%	0.01%	0.02%	0.05%	0.05%
g6	0.07%	0.11%	0.01%	0.07%	0.05%	0.04%	0.03%	0.02%
a5	0.04%	0.91%	0.13%	0.17%	0.06%	0.07%	0.09%	0.02%
♘e7	0.25%	0.01%	0.01%	0.05%	0.06%	0.10%	0.12%	0.09%
♕d8	0.05%	0.13%	0.04%	0.76%	0.24%	0.22%	0.12%	0.06%
♖e7	0.32%	0.01%	0.05%	0.05%	0.12%	0.68%	0.17%	0.09%
♔f8	0.06%	0.04%	0.02%	0.04%	0.25%	0.68%	0.17%	0.22%
♘h4	0.65%	0.04%	0.01%	0.04%	0.05%	0.03%	0.03%	0.05%
g5	0.54%	0.38%	0.23%	0.07%	0.04%	0.03%	0.01%	0.01%
♕d5	0.02%	0.05%	0.06%	0.31%	0.22%	0.14%	0.52%	0.38%
♕d7	0.04%	0.04%	0.02%	0.09%	0.22%	0.41%	0.11%	0.16%
♖g6	0.07%	0.01%	0.02%	0.09%	0.16%	0.41%	0.04%	0.05%
♕d6	0.02%	0.02%	0.03%	0.08%	0.06%	0.19%	0.07%	0.07%
♖d6	0.10%	0.01%	0.03%	0.06%	0.06%	0.25%	0.07%	0.06%
♖f6	0.10%	0.01%	0.12%	0.07%	0.03%	0.11%	0.03%	0.03%

Table 6: Move probabilities by layer for puzzle 62FeU (Part 2: Layer 7 to Final)

Move	Layer 7	Layer 8	Layer 9	Layer 10	Layer 11	Layer 12	Layer 13	Final
♘g3+	27.02%	20.06%	19.00%	25.38%	39.13%	28.03%	49.15%	87.55%
♖h6	8.66%	15.63%	9.35%	10.04%	28.26%	51.73%	9.61%	0.24%
♕g1+	18.86%	15.30%	49.02%	47.22%	22.42%	2.68%	2.08%	0.28%
♕×b2	0.27%	2.92%	0.42%	0.35%	0.20%	0.10%	0.86%	0.25%
♕×d2	0.19%	1.72%	0.34%	0.27%	0.13%	0.08%	0.63%	0.24%
♖×e5	9.74%	6.51%	5.83%	3.66%	2.71%	1.44%	0.71%	0.37%
♕f2	0.07%	0.21%	0.16%	0.55%	0.08%	0.04%	0.34%	0.27%
♕×e5	0.17%	0.25%	0.37%	0.13%	0.09%	0.06%	0.67%	0.37%
♕×f4	13.50%	18.19%	5.50%	6.82%	0.50%	0.17%	0.55%	0.27%
♕×c4	12.17%	13.07%	4.70%	1.31%	0.37%	0.20%	0.75%	0.29%
h6	0.02%	0.14%	0.12%	0.61%	1.38%	3.87%	14.66%	0.25%
♕d3	0.07%	0.15%	0.22%	0.16%	0.33%	0.15%	0.59%	0.26%
♕e3	0.37%	0.15%	0.06%	0.03%	0.08%	0.04%	0.37%	0.25%
♕c3	1.42%	0.38%	0.22%	0.08%	0.12%	0.03%	0.50%	0.22%
♘e3	0.04%	0.04%	0.03%	0.02%	0.03%	0.03%	0.24%	0.21%
♘h6	4.29%	3.37%	2.22%	1.66%	2.45%	9.23%	6.39%	0.23%
♖c6	0.58%	0.22%	0.24%	0.06%	0.06%	0.05%	0.21%	0.34%
h5	0.20%	0.07%	0.19%	0.32%	0.24%	0.14%	0.88%	0.25%
♔h8	0.33%	0.25%	0.48%	0.32%	0.34%	0.48%	0.27%	0.27%
♘d6	0.17%	0.12%	0.34%	0.15%	0.09%	0.12%	1.97%	0.82%
♕e4	0.11%	0.07%	0.07%	0.03%	0.04%	0.02%	0.31%	0.26%
♖b6	0.12%	0.05%	0.05%	0.02%	0.03%	0.05%	0.28%	0.28%
♖e8	0.10%	0.13%	0.12%	0.07%	0.06%	0.20%	0.70%	1.35%
f6	0.33%	0.13%	0.11%	0.14%	0.15%	0.27%	1.28%	0.29%
g6	0.04%	0.03%	0.05%	0.08%	0.07%	0.12%	1.05%	0.31%
a5	0.04%	0.01%	0.02%	0.01%	0.01%	0.02%	0.18%	0.23%
♘e7	0.07%	0.05%	0.04%	0.02%	0.02%	0.01%	0.84%	0.39%
♕d8	0.08%	0.08%	0.07%	0.05%	0.06%	0.05%	0.48%	0.81%
♖e7	0.08%	0.04%	0.03%	0.01%	0.02%	0.03%	0.33%	0.26%
♔f8	0.21%	0.22%	0.21%	0.10%	0.05%	0.08%	0.22%	0.35%
♘h4	0.05%	0.04%	0.03%	0.04%	0.05%	0.07%	0.39%	0.20%
g5	0.02%	0.02%	0.05%	0.04%	0.03%	0.04%	0.61%	0.29%
♕d5	0.35%	0.13%	0.14%	0.05%	0.07%	0.05%	0.41%	0.27%
♕d7	0.04%	0.09%	0.10%	0.07%	0.08%	0.06%	0.22%	0.35%
♖g6	0.05%	0.04%	0.05%	0.05%	0.10%	0.09%	0.23%	0.28%
♕d6	0.07%	0.04%	0.03%	0.03%	0.05%	0.06%	0.41%	0.28%
♖d6	0.06%	0.04%	0.02%	0.01%	0.04%	0.04%	0.34%	0.29%
♖f6	0.05%	0.05%	0.03%	0.02%	0.04%	0.06%	0.29%	0.28%

I Correctly solved puzzle case studies

I.1 Knight fork and discovered attack

This puzzle (Figure 11) features a tactical sequence beginning with a knight check that sets up a discovered attack (PV: 1. ♘**d6+** **e×d6** 2. ♗**×c6+**). After the knight is captured, the bishop takes the queen's only defender with check. Black must answer the check instead of saving the queen, and White captures the undefended queen on the next move. The input layer exhibits piece-specific bias toward knight moves, which are abandoned in early layers as the model shifts focus to queen captures in layer 1. The queen trade ♕**×a5**, with the queen being protected by a knight, dominates through layers 4–13 with probabilities around 50%. Apart from 11.55% at layer 2, the winning move ♘**d6+** maintains low probability until layer 12 (1.65%), then suddenly jumps to 24.37% at layer 13 before becoming the decisive top choice in the final output. This sudden increase at layer 13—the same layer Cruz (2025) identified as containing one of the look-ahead heads—may indicate the model requires multi-move analysis rather than simple tactical heuristics for this position. Full probabilities are in Tables 7 and 8.

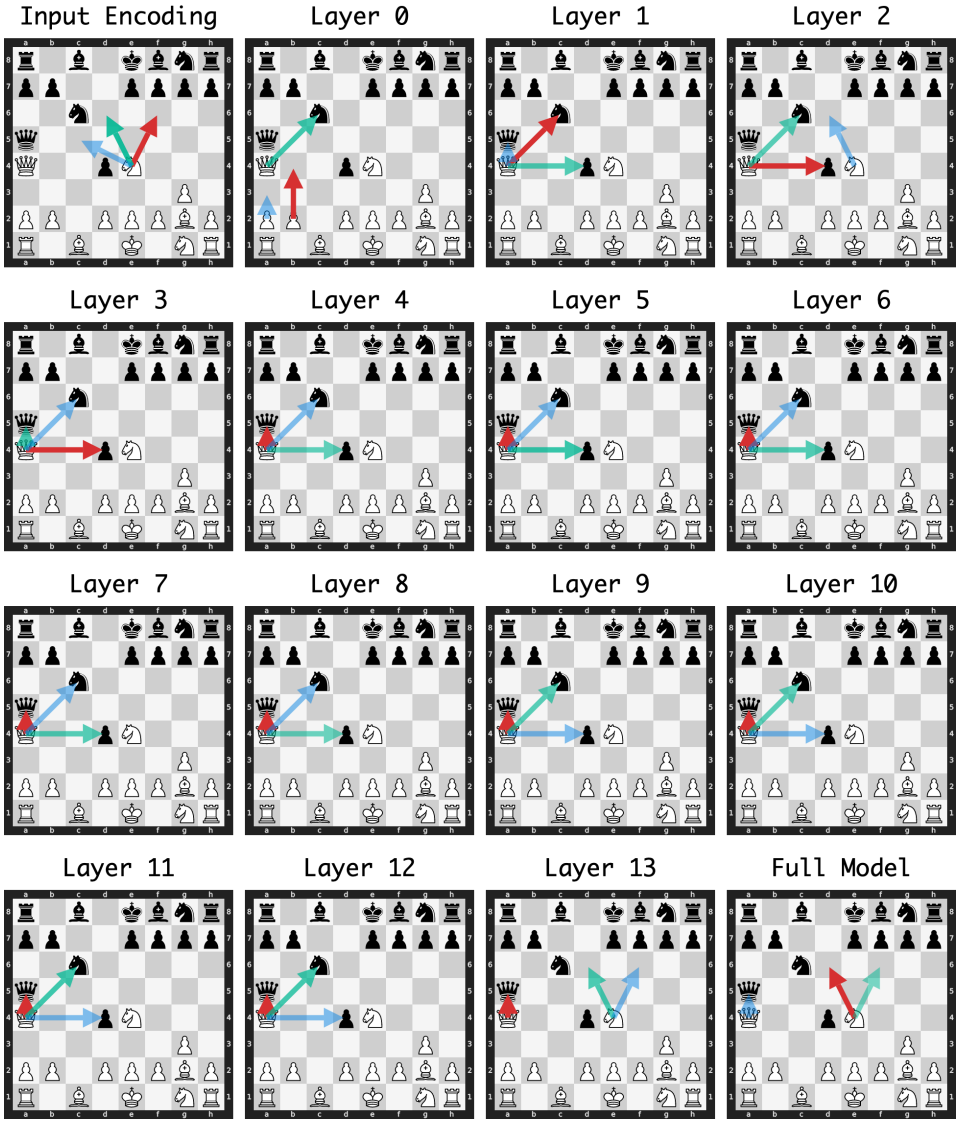


Figure 11: Layer-wise policy evolution for puzzle 9JvtA.

Table 7: Move probabilities by layer for puzzle 9JvtA (Part 1: Input to Layer 6)

Move	Input	Layer 0	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
♘d6+	32.13%	0.05%	3.60%	11.55%	0.14%	0.10%	0.75%	0.50%
♕×a5	0.20%	4.42%	6.22%	9.20%	26.54%	50.71%	47.82%	58.34%
♕×c6+	0.90%	26.71%	56.31%	19.67%	20.88%	16.16%	7.71%	5.78%
b4	1.44%	46.77%	0.36%	0.88%	0.58%	0.01%	0.01%	0.16%
♕×d4	1.50%	7.14%	22.75%	39.84%	34.48%	23.90%	29.99%	23.33%
♘f6+	36.04%	0.01%	0.17%	2.27%	0.05%	0.13%	0.01%	0.01%
♘c5	17.13%	0.04%	0.27%	0.60%	0.17%	0.09%	0.13%	0.09%
a3	0.03%	7.50%	2.85%	6.10%	0.13%	0.57%	0.16%	0.20%
♘f3	0.07%	0.01%	0.03%	0.20%	4.38%	0.12%	4.86%	3.17%
f4	3.27%	0.27%	0.03%	0.04%	0.04%	0.15%	0.05%	0.11%
♗f3	0.29%	0.02%	0.51%	0.37%	1.85%	0.86%	3.02%	1.93%
♕b4	0.10%	0.44%	0.51%	0.92%	2.59%	0.20%	0.11%	0.27%
♘g5	2.51%	0.05%	0.02%	0.31%	0.17%	0.29%	0.02%	0.05%
♕b3	0.09%	0.10%	0.20%	1.28%	2.22%	0.42%	0.98%	1.08%
♕a3	0.04%	1.51%	0.93%	1.41%	1.87%	0.50%	0.50%	0.87%
♘c3	0.36%	0.02%	1.86%	1.02%	0.21%	0.19%	0.04%	0.20%
g4	1.75%	0.30%	0.24%	0.71%	0.38%	0.25%	0.03%	0.02%
♖b1	0.16%	0.10%	0.09%	0.17%	0.21%	1.25%	1.09%	1.68%
f3	0.20%	0.00%	0.00%	0.00%	0.00%	0.11%	0.04%	0.08%
h3	0.03%	1.44%	0.12%	0.41%	0.03%	0.12%	0.22%	0.08%
♗h3	0.02%	0.06%	0.11%	0.15%	0.98%	0.96%	0.43%	0.24%
♔f1	0.11%	0.37%	0.96%	0.25%	0.25%	0.19%	0.17%	0.09%
b3	0.17%	0.21%	0.08%	0.92%	0.01%	0.03%	0.03%	0.01%
♗f1	0.01%	0.08%	0.87%	0.35%	0.51%	0.29%	0.17%	0.05%
♘h3	0.15%	0.83%	0.12%	0.21%	0.04%	0.13%	0.23%	0.22%
♔d1	0.12%	0.11%	0.05%	0.09%	0.41%	0.75%	0.40%	0.56%
e3	0.42%	0.49%	0.03%	0.23%	0.16%	0.75%	0.24%	0.25%
♕c4	0.25%	0.11%	0.03%	0.08%	0.16%	0.11%	0.18%	0.16%
h4	0.09%	0.52%	0.56%	0.45%	0.15%	0.21%	0.16%	0.14%
♕d1	0.11%	0.09%	0.03%	0.08%	0.07%	0.17%	0.16%	0.08%
♕b5	0.12%	0.12%	0.06%	0.11%	0.30%	0.13%	0.16%	0.15%
♕c2	0.16%	0.10%	0.04%	0.12%	0.06%	0.16%	0.14%	0.10%

Table 8: Move probabilities by layer for puzzle 9JvtA (Part 2: Layer 7 to Final)

Move	Layer 7	Layer 8	Layer 9	Layer 10	Layer 11	Layer 12	Layer 13	Final
♘**d6+**	1.22%	1.26%	2.22%	2.73%	1.72%	1.65%	24.37%	65.34%
♕**×a5**	58.41%	52.60%	50.88%	61.05%	56.84%	56.52%	46.15%	12.18%
♕**×c6+**	6.60%	14.67%	25.41%	22.02%	30.43%	28.79%	3.26%	0.61%
b4	0.44%	1.59%	0.40%	0.70%	1.71%	3.37%	3.06%	0.29%
♕**×d4**	22.78%	19.53%	14.91%	8.88%	4.27%	4.65%	0.79%	0.29%
♘**f6+**	0.01%	0.02%	0.02%	0.02%	0.02%	0.02%	11.00%	13.67%
♘**c5**	0.08%	0.12%	0.20%	0.13%	0.14%	0.09%	0.69%	0.29%
a3	0.93%	0.78%	0.28%	0.55%	0.22%	0.19%	1.90%	0.28%
♘**f3**	1.91%	1.70%	0.89%	0.83%	0.41%	0.14%	0.73%	0.28%
f4	0.40%	0.48%	0.31%	0.57%	1.95%	1.26%	1.22%	0.26%
♗**f3**	0.72%	0.55%	0.35%	0.16%	0.11%	0.51%	0.38%	0.29%
♕**b4**	0.68%	0.60%	0.26%	0.12%	0.08%	0.06%	0.05%	0.27%
♘**g5**	0.22%	0.45%	0.03%	0.03%	0.02%	0.19%	0.34%	0.27%
♕**b3**	1.94%	2.00%	0.67%	0.28%	0.12%	0.32%	0.26%	0.29%
♕**a3**	0.78%	1.30%	0.72%	0.37%	0.21%	0.14%	0.12%	0.29%
♘**c3**	0.25%	0.13%	0.07%	0.11%	0.06%	0.17%	0.66%	0.29%
g4	0.02%	0.02%	0.02%	0.04%	0.04%	0.04%	0.16%	0.29%
♖**b1**	0.55%	0.51%	0.77%	0.10%	0.10%	0.12%	0.38%	0.31%
f3	0.03%	0.06%	0.02%	0.02%	0.02%	0.13%	1.47%	0.26%
h3	0.06%	0.04%	0.04%	0.03%	0.05%	0.04%	0.33%	0.30%
♗**h3**	0.20%	0.22%	0.21%	0.05%	0.07%	0.05%	0.16%	0.32%
♔**f1**	0.07%	0.05%	0.08%	0.07%	0.04%	0.02%	0.13%	0.33%
b3	0.08%	0.05%	0.03%	0.04%	0.02%	0.01%	0.15%	0.25%
♗**f1**	0.04%	0.02%	0.03%	0.03%	0.05%	0.05%	0.15%	0.34%
♘**h3**	0.18%	0.17%	0.08%	0.03%	0.08%	0.19%	0.36%	0.29%
♔**d1**	0.54%	0.29%	0.37%	0.32%	0.40%	0.28%	0.21%	0.32%
e3	0.26%	0.25%	0.19%	0.18%	0.19%	0.10%	0.55%	0.28%
♕**c4**	0.14%	0.21%	0.19%	0.19%	0.25%	0.56%	0.32%	0.28%
h4	0.17%	0.11%	0.06%	0.06%	0.08%	0.06%	0.20%	0.33%
♕**d1**	0.08%	0.04%	0.05%	0.04%	0.04%	0.04%	0.15%	0.32%
♕**b5**	0.10%	0.13%	0.17%	0.17%	0.11%	0.13%	0.10%	0.29%
♕**c2**	0.11%	0.06%	0.07%	0.09%	0.14%	0.13%	0.18%	0.29%

I.2 Early and consistent solution lock-in

This puzzle (Figure 12) features a back-rank mate initiated by a queen sacrifice (PV: 1. ♕×**e8+** ♖×**e8** 2. ♖×**e8**#). The winning move ♕×**e8+** becomes the top candidate immediately at layer 0 (50.5%) and maintains dominance through nearly all subsequent layers. The only significant competitor is the pawn capture **b×c3**, which briefly overtakes the solution at layer 3 (56.3% vs 40.7%) and layer 7 (52.9% vs 42.9%), and only disappears in the final output. All other moves remain at or below 1.5% from layer 1 onwards. The solution reaches 88.9% in the final output. This early convergence contrasts with the delayed emergence seen in other case studies and may reflect the winning move combining both queen-move and capture biases present in initial layers. Full probabilities are in Tables 9 and 10.

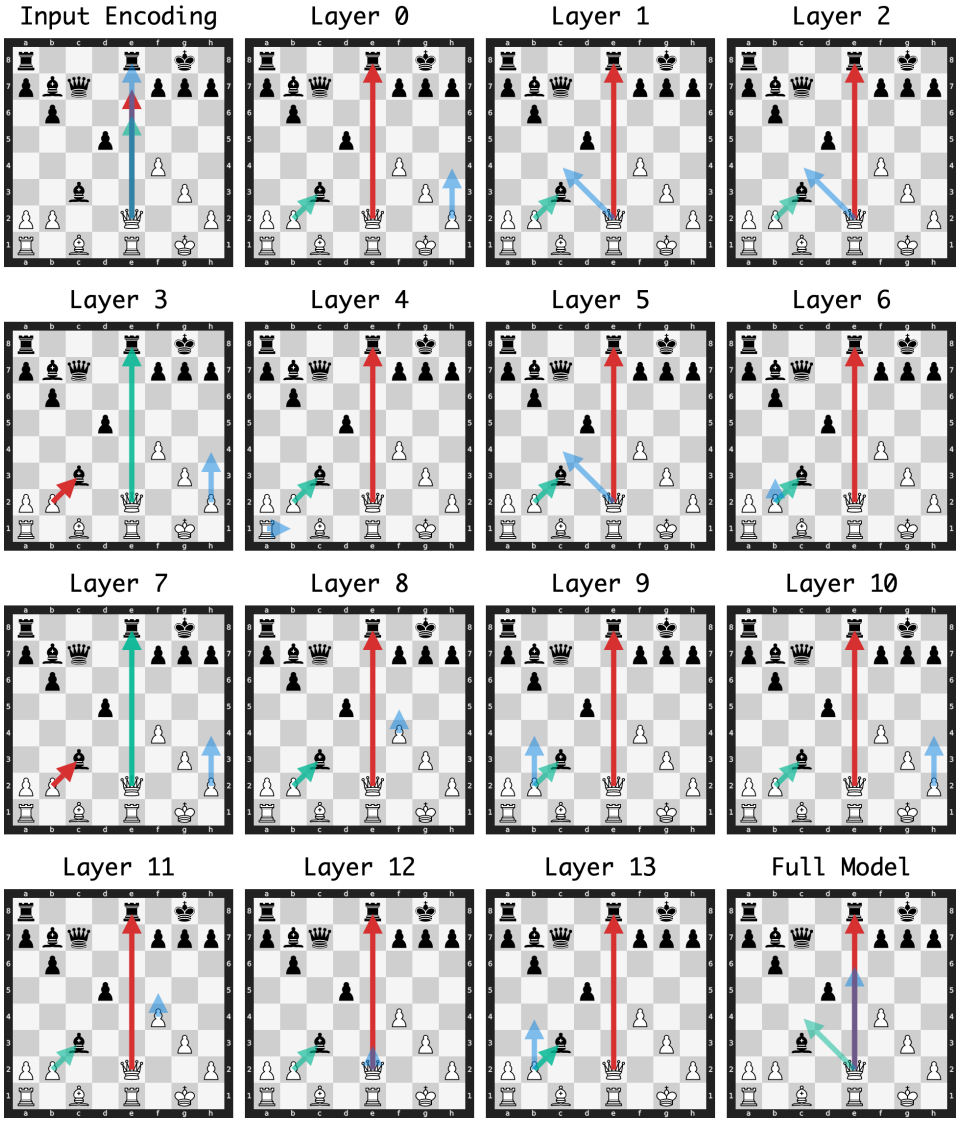


Figure 12: Layer-wise policy evolution for puzzle 70v1B.

Table 9: Move probabilities by layer for puzzle 70v1B (Part 1: Input to Layer 6)

Move	Input	Layer 0	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
♕×e8+	11.14%	50.54%	71.54%	80.98%	40.69%	64.80%	66.88%	68.29%
♕e7	56.93%	0.68%	0.16%	0.06%	0.08%	0.09%	0.06%	0.09%
b×c3	0.18%	20.37%	21.51%	13.64%	56.27%	30.75%	30.98%	27.02%
♕e6	20.58%	1.07%	1.00%	0.16%	0.12%	0.10%	0.10%	0.16%
h4	0.08%	6.79%	0.81%	0.44%	0.37%	0.11%	0.07%	0.29%
b4	0.72%	6.44%	1.49%	0.42%	0.06%	0.05%	0.02%	0.04%
a4	0.07%	4.65%	0.20%	0.08%	0.02%	0.02%	0.09%	0.14%
♕e5	0.69%	1.88%	0.49%	0.30%	0.22%	0.13%	0.06%	0.10%
♕f2	1.59%	0.15%	0.03%	0.12%	0.09%	0.03%	0.04%	0.10%
♕c4	0.62%	0.56%	1.50%	1.15%	0.20%	0.13%	0.12%	0.23%
f5	0.63%	1.44%	0.01%	0.01%	0.01%	0.01%	0.04%	0.14%
♖b1	0.11%	0.04%	0.02%	0.12%	0.19%	1.31%	0.06%	0.29%
♕g2	1.14%	0.10%	0.02%	0.07%	0.03%	0.04%	0.03%	0.08%
a3	0.02%	0.94%	0.05%	0.04%	0.04%	0.07%	0.06%	0.25%
g4	0.82%	0.83%	0.02%	0.04%	0.09%	0.04%	0.05%	0.12%
♕b5	0.62%	0.10%	0.06%	0.09%	0.05%	0.07%	0.08%	0.17%
♕g4	0.61%	0.34%	0.09%	0.23%	0.16%	0.12%	0.10%	0.19%
♖d1	0.13%	0.17%	0.06%	0.05%	0.04%	0.55%	0.07%	0.09%
♕d1	0.08%	0.49%	0.16%	0.26%	0.09%	0.14%	0.07%	0.14%
♗e3	0.05%	0.01%	0.02%	0.13%	0.02%	0.05%	0.08%	0.23%
♕f3	0.48%	0.17%	0.02%	0.05%	0.04%	0.07%	0.07%	0.11%
♕e3	0.42%	0.11%	0.03%	0.07%	0.03%	0.03%	0.09%	0.10%
b3	0.11%	0.34%	0.04%	0.06%	0.05%	0.07%	0.05%	0.30%
♕e4	0.40%	0.14%	0.15%	0.16%	0.11%	0.08%	0.06%	0.16%
♕a6	0.05%	0.31%	0.07%	0.06%	0.04%	0.05%	0.05%	0.08%
♕c2	0.37%	0.13%	0.04%	0.13%	0.06%	0.12%	0.07%	0.17%
♕h5	0.13%	0.30%	0.05%	0.11%	0.10%	0.09%	0.08%	0.18%
h3	0.02%	0.29%	0.05%	0.07%	0.04%	0.01%	0.03%	0.04%
♔h1	0.26%	0.06%	0.04%	0.06%	0.04%	0.10%	0.06%	0.13%
♔f2	0.10%	0.12%	0.07%	0.30%	0.28%	0.10%	0.03%	0.08%
♕d3	0.22%	0.07%	0.04%	0.09%	0.05%	0.10%	0.09%	0.15%
♔g2	0.11%	0.03%	0.03%	0.16%	0.11%	0.14%	0.06%	0.12%
♕d2	0.07%	0.09%	0.03%	0.04%	0.04%	0.05%	0.03%	0.04%
♗d2	0.10%	0.01%	0.01%	0.02%	0.02%	0.06%	0.05%	0.07%
♔f1	0.10%	0.04%	0.03%	0.10%	0.07%	0.12%	0.04%	0.04%
♕f1	0.10%	0.16%	0.03%	0.08%	0.04%	0.05%	0.03%	0.05%
♖f1	0.16%	0.04%	0.02%	0.07%	0.03%	0.13%	0.03%	0.03%

Table 10: Move probabilities by layer for puzzle 70v1B (Part 2: Layer 7 to Final)

Move	Layer 7	Layer 8	Layer 9	Layer 10	Layer 11	Layer 12	Layer 13	Final
♕×e8+	42.88%	60.35%	74.76%	69.45%	78.19%	75.30%	53.67%	88.91%
♕e7	0.11%	0.11%	0.12%	0.30%	0.43%	0.21%	0.26%	0.33%
b×c3	52.88%	35.65%	20.83%	25.08%	17.09%	20.41%	35.91%	0.28%
♕e6	0.13%	0.07%	0.08%	0.08%	0.15%	0.12%	0.25%	0.41%
h4	0.38%	0.25%	0.22%	0.86%	0.28%	0.14%	0.31%	0.34%
b4	0.05%	0.11%	0.47%	0.56%	0.10%	0.22%	1.02%	0.33%
a4	0.14%	0.10%	0.04%	0.09%	0.10%	0.17%	0.97%	0.29%
♕e5	0.11%	0.09%	0.13%	0.14%	0.24%	0.23%	0.22%	0.37%
♕f2	0.05%	0.04%	0.04%	0.04%	0.04%	0.07%	0.15%	0.24%
♕c4	0.17%	0.09%	0.16%	0.14%	0.12%	0.10%	0.33%	0.45%
f5	0.29%	1.17%	0.44%	0.36%	0.45%	0.22%	0.56%	0.30%
♖b1	0.08%	0.05%	0.13%	0.19%	0.17%	0.12%	0.37%	0.37%
♕g2	0.09%	0.04%	0.04%	0.04%	0.06%	0.10%	0.16%	0.27%
a3	0.18%	0.11%	0.10%	0.12%	0.07%	0.07%	0.50%	0.32%
g4	0.13%	0.14%	0.25%	0.44%	0.23%	0.22%	0.75%	0.30%
♕b5	0.13%	0.12%	0.10%	0.08%	0.08%	0.10%	0.21%	0.21%
♕g4	0.13%	0.08%	0.10%	0.11%	0.10%	0.06%	0.20%	0.35%
♖d1	0.09%	0.06%	0.10%	0.06%	0.06%	0.06%	0.26%	0.27%
♕d1	0.11%	0.05%	0.08%	0.12%	0.12%	0.06%	0.17%	0.29%
♗e3	0.15%	0.12%	0.20%	0.10%	0.06%	0.17%	0.49%	0.21%
♕f3	0.12%	0.05%	0.09%	0.09%	0.10%	0.07%	0.19%	0.34%
♕e3	0.07%	0.16%	0.12%	0.06%	0.12%	0.26%	0.23%	0.21%
b3	0.33%	0.15%	0.33%	0.25%	0.19%	0.09%	0.39%	0.41%
♕e4	0.13%	0.08%	0.13%	0.18%	0.27%	0.21%	0.23%	0.40%
♕a6	0.07%	0.09%	0.09%	0.10%	0.11%	0.06%	0.15%	0.38%
♕c2	0.20%	0.09%	0.14%	0.16%	0.15%	0.10%	0.21%	0.34%
♕h5	0.14%	0.10%	0.10%	0.14%	0.12%	0.07%	0.23%	0.37%
h3	0.10%	0.05%	0.06%	0.14%	0.07%	0.04%	0.27%	0.33%
♔h1	0.08%	0.06%	0.05%	0.03%	0.05%	0.07%	0.09%	0.33%
♔f2	0.04%	0.03%	0.04%	0.07%	0.09%	0.15%	0.10%	0.24%
♕d3	0.14%	0.12%	0.12%	0.08%	0.11%	0.11%	0.22%	0.28%
♔g2	0.07%	0.06%	0.05%	0.09%	0.19%	0.20%	0.11%	0.27%
♕d2	0.02%	0.03%	0.05%	0.02%	0.02%	0.04%	0.08%	0.27%
♗d2	0.10%	0.08%	0.16%	0.10%	0.07%	0.12%	0.27%	0.23%
♔f1	0.02%	0.02%	0.03%	0.05%	0.10%	0.11%	0.08%	0.26%
♕f1	0.05%	0.02%	0.03%	0.05%	0.10%	0.09%	0.17%	0.26%
♖f1	0.03%	0.01%	0.02%	0.02%	0.03%	0.05%	0.23%	0.25%

I.3 Knight sacrifice with rook mate

This puzzle (Figure 13), featured in Jenner et al. (2024), requires a knight sacrifice (PV: 1. ♘**g6+** **h×g6** 2. ♖**h4**#). The input encoding favors specific pieces, while layer 0 is dominated by the queen check ♕**g8+** (40.0%), which reappears briefly at layer 2. Through middle layers the model favors the rook move ♖**c4**, which attacks the black queen; it peaks at 69.4% (layer 4) and remains the top candidate through layer 8. Meanwhile, the queen maneuver ♕**e6** threatening the opposing queen and rook while being protected by the knight is also considered in layers 7–11. The correct sacrifice ♘**g6+** remains below 10% through layer 7, rises to 10.8% at layer 8, becomes the top candidate at layer 9 (14.7%), and climbs to a peak of 53.1% at layer 12. After a brief dip at layer 13 (33.2%), where the quiet pawn push **h4** is considered, the solution recovers to dominate the final output at 70.7%. Full probabilities are in Tables 11 and 12.

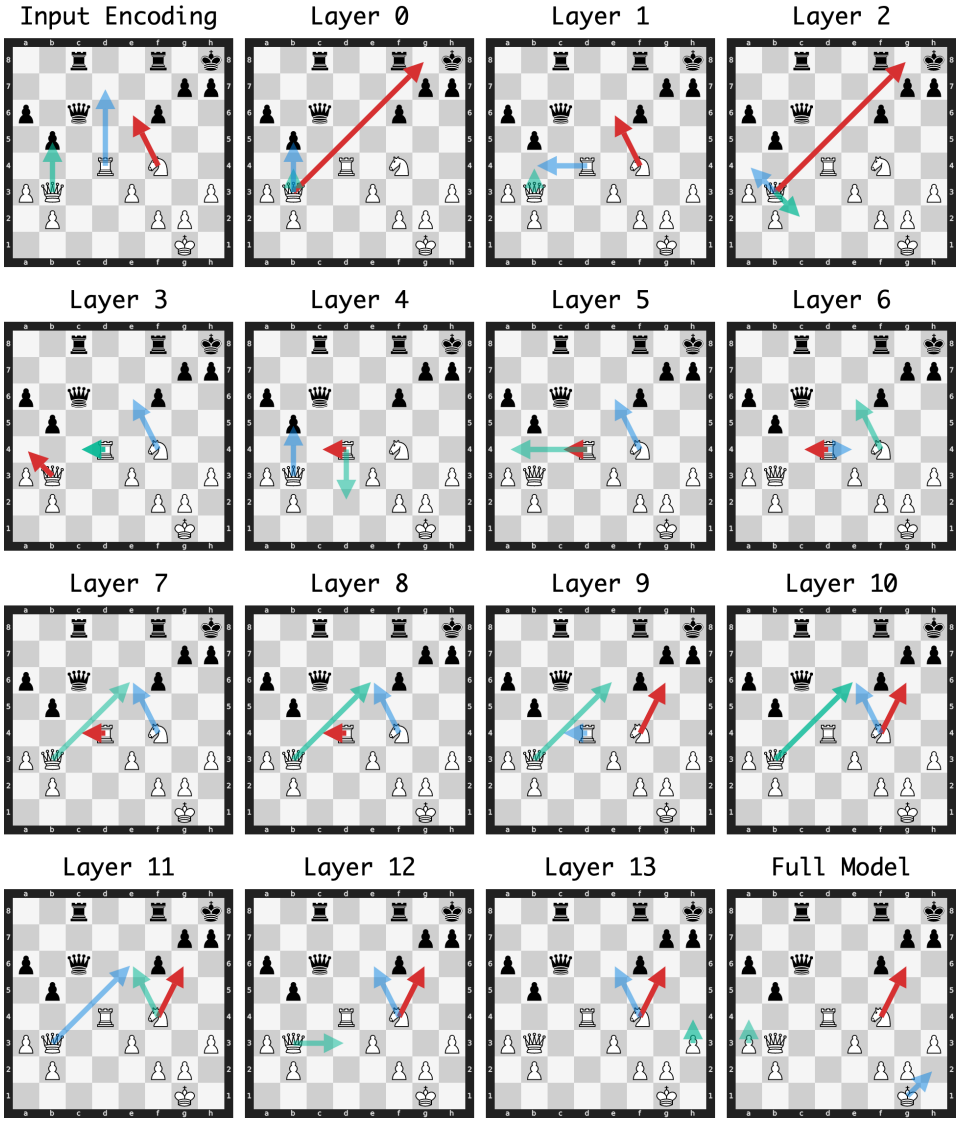


Figure 13: Layer-wise policy evolution for puzzle `0KBPy`.

Table 11: Move probabilities by layer for puzzle `0KBPy` (Part 1: Input to Layer 6)

Move	Input	Layer 0	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
♘**g6+**	9.76%	0.46%	0.88%	0.35%	6.34%	0.12%	1.38%	0.77%
♖**c4**	0.15%	0.32%	0.36%	1.20%	14.79%	69.39%	48.78%	41.74%
♕**g8+**	0.07%	40.00%	7.11%	18.17%	4.87%	0.36%	0.60%	0.77%
♘**e6**	19.23%	6.12%	30.11%	7.38%	8.51%	0.56%	10.91%	10.97%
h4	0.09%	6.94%	0.16%	0.36%	0.28%	0.07%	0.23%	0.04%
♕**a4**	0.05%	2.77%	5.95%	10.38%	16.30%	2.73%	5.80%	3.49%
♕**×b5**	16.21%	8.15%	7.62%	6.20%	2.83%	3.07%	2.87%	1.41%
♖**d7**	15.49%	0.31%	0.19%	0.42%	1.07%	0.75%	0.35%	0.89%
♕**c2**	0.38%	0.71%	1.63%	13.84%	4.56%	0.78%	0.24%	0.88%
♕**e6**	1.77%	4.49%	1.31%	0.20%	0.70%	0.16%	1.50%	5.27%
♖**a4**	0.12%	0.51%	1.20%	1.83%	3.50%	2.12%	12.92%	6.79%
♕**b4**	0.19%	12.14%	11.15%	3.00%	2.59%	0.49%	0.36%	0.37%
♖**d6**	10.35%	0.26%	0.23%	0.53%	1.96%	1.40%	0.44%	0.74%
♕**d3**	0.18%	0.28%	0.59%	0.45%	0.44%	0.22%	0.19%	0.31%
♘**d3**	0.33%	0.18%	0.06%	0.62%	1.27%	0.14%	1.53%	1.31%
♖**b4**	0.09%	2.99%	9.44%	0.75%	0.50%	0.25%	0.04%	0.04%
♘**d5**	9.23%	0.10%	0.21%	2.13%	3.00%	0.24%	0.92%	0.73%
♕**c4**	0.31%	1.70%	5.11%	8.20%	4.18%	1.06%	2.18%	2.54%
♖**e4**	0.15%	0.29%	1.43%	2.45%	2.82%	0.97%	1.38%	7.30%
a4	0.08%	3.03%	0.08%	0.24%	0.26%	0.11%	0.22%	0.07%
♔**h2**	0.20%	0.93%	0.20%	0.38%	0.18%	0.19%	0.66%	2.47%
♖**d8**	2.55%	0.12%	0.06%	0.31%	0.67%	1.80%	0.30%	3.34%
♖**d2**	0.18%	0.20%	0.24%	0.73%	2.22%	4.69%	1.23%	1.35%
♕**f7**	1.78%	0.73%	3.87%	4.31%	2.14%	0.47%	0.19%	0.20%
♔**f1**	0.13%	0.38%	0.12%	0.63%	1.72%	1.18%	0.41%	0.87%
♖**d5**	3.77%	0.27%	2.25%	1.07%	0.19%	0.07%	0.14%	0.65%
♕**d1**	0.12%	0.38%	0.27%	0.52%	1.35%	0.80%	0.28%	0.28%
♕**c3**	0.10%	0.51%	0.70%	3.10%	2.00%	1.10%	0.45%	0.82%
e4	0.54%	0.49%	0.91%	3.02%	0.92%	0.08%	0.10%	0.22%
♕**a2**	0.03%	1.39%	1.96%	2.76%	2.71%	1.38%	1.00%	1.24%
♘**h5**	2.21%	0.49%	0.05%	0.18%	0.51%	0.39%	0.16%	0.08%
f3	0.17%	0.12%	0.20%	0.60%	1.83%	0.13%	0.11%	0.08%
♘**e2**	1.09%	0.13%	0.90%	0.13%	0.05%	0.07%	0.27%	0.21%
♖**d3**	0.11%	0.20%	1.19%	0.63%	0.16%	0.08%	0.17%	0.08%
♔**h1**	0.34%	0.59%	0.71%	1.13%	0.27%	1.10%	0.59%	0.41%
g4	1.09%	0.48%	0.13%	0.69%	0.69%	0.11%	0.25%	0.25%
♖**d1**	0.16%	0.17%	0.26%	0.32%	0.41%	1.06%	0.53%	0.54%
♕**d5**	1.04%	0.23%	1.04%	0.50%	0.40%	0.22%	0.22%	0.26%
g3	0.17%	0.41%	0.09%	0.30%	0.80%	0.10%	0.12%	0.24%

Table 12: Move probabilities by layer for puzzle `0KBPy` (Part 2: Layer 7 to Final)

Move	Layer 7	Layer 8	Layer 9	Layer 10	Layer 11	Layer 12	Layer 13	Final
♘g6+	3.14%	10.83%	14.71%	16.18%	35.31%	53.13%	33.17%	70.65%
♖c4	33.25%	14.94%	13.26%	5.06%	1.01%	0.19%	0.28%	0.18%
♕g8+	1.51%	1.85%	5.92%	9.55%	1.02%	0.73%	0.52%	0.16%
♘e6	8.39%	12.37%	10.91%	10.09%	10.47%	9.26%	12.17%	1.46%
h4	0.04%	0.04%	0.01%	0.04%	1.05%	1.58%	17.72%	4.76%
♕a4	2.40%	3.49%	1.38%	2.76%	2.78%	0.47%	0.31%	0.14%
♕×b5	1.99%	4.93%	2.81%	2.12%	0.64%	0.62%	1.30%	0.16%
♖d7	1.08%	0.99%	1.22%	2.37%	3.61%	1.34%	0.26%	0.21%
♕c2	1.09%	2.72%	5.97%	6.75%	1.36%	0.28%	0.21%	0.16%
♕e6	8.67%	13.05%	13.50%	13.51%	8.71%	2.56%	3.14%	0.48%
♖a4	6.37%	4.91%	0.40%	2.20%	1.72%	0.23%	0.12%	0.15%
♕b4	0.48%	0.85%	0.70%	0.98%	0.74%	0.77%	0.58%	0.71%
♖d6	0.46%	0.31%	0.45%	0.50%	0.67%	0.48%	0.14%	0.18%
♕d3	1.20%	1.38%	2.80%	4.34%	7.01%	9.79%	6.29%	4.04%
♘d3	2.31%	2.88%	4.05%	2.28%	3.70%	3.28%	9.63%	0.49%
♖b4	0.04%	0.06%	0.04%	0.09%	0.10%	0.09%	0.05%	0.15%
♘d5	0.95%	0.92%	0.48%	0.29%	0.24%	0.24%	0.50%	0.15%
♕c4	1.41%	1.06%	0.93%	0.54%	0.30%	0.24%	0.22%	0.17%
♖e4	6.63%	3.24%	4.44%	4.90%	3.39%	1.89%	0.50%	0.19%
a4	0.18%	0.08%	0.01%	0.03%	0.09%	0.03%	1.02%	5.80%
♔h2	5.48%	3.97%	2.81%	2.75%	0.81%	2.87%	3.23%	5.54%
♖d8	2.37%	5.16%	1.86%	1.57%	2.43%	0.85%	0.37%	0.17%
♖d2	1.06%	0.74%	1.34%	0.25%	0.14%	0.19%	0.42%	0.25%
♕f7	0.41%	0.76%	0.87%	1.69%	1.04%	0.30%	0.31%	0.20%
♔f1	1.68%	2.08%	2.11%	1.84%	4.25%	3.32%	0.32%	0.15%
♖d5	0.94%	0.62%	0.33%	0.35%	0.64%	0.62%	0.42%	0.22%
♕d1	1.01%	0.93%	1.71%	1.89%	3.13%	1.05%	0.62%	0.46%
♕c3	0.63%	0.55%	0.65%	0.87%	0.45%	0.29%	0.17%	0.15%
e4	0.31%	0.30%	0.50%	0.64%	0.46%	0.48%	0.61%	0.18%
♕a2	1.54%	1.18%	1.05%	0.78%	0.45%	0.15%	0.34%	0.27%
♘h5	0.17%	0.17%	0.05%	0.19%	0.11%	0.16%	0.37%	0.18%
f3	0.30%	0.17%	0.21%	0.21%	0.17%	0.12%	0.57%	0.18%
♘e2	0.47%	0.30%	0.50%	0.38%	0.50%	0.67%	1.46%	0.18%
♖d3	0.10%	0.09%	0.05%	0.08%	0.06%	0.08%	0.24%	0.19%
♔h1	0.54%	0.79%	0.83%	0.95%	0.48%	0.69%	0.45%	0.19%
g4	0.29%	0.24%	0.11%	0.17%	0.23%	0.13%	0.25%	0.40%
♖d1	0.57%	0.36%	0.47%	0.15%	0.09%	0.30%	0.76%	0.25%
♕d5	0.45%	0.64%	0.55%	0.63%	0.63%	0.47%	0.68%	0.16%
g3	0.11%	0.05%	0.03%	0.03%	0.01%	0.06%	0.26%	0.29%

I.4 Queen sacrifice for rook mate

This puzzle (Figure 14) involves a queen sacrifice enabling a back-rank rook mate (PV: 1. ♕**e8+** ♖×**e8** 2. ♖×**e8**#). The input encoding favors quiet queen moves ♕**d7** (30.6%) and ♕**c7** (20.7%). From layer 0 onwards, the model shifts toward captures and checks: the queen capture ♕×**f7+** (28.9%–39.3%, layers 0–4), the material capture ♕×**a3** (33.8% at layer 0, declining to 2.9% by layer 12), and the rook capture ♖×**a3**, which persists throughout (10%–20% across most layers). The pawn check **g**×**f7+** dominates the middle layers, peaking at 43.5% (layer 6) and remaining the top candidate through layer 12—a forcing move that delivers check but lacks the mating continuation. The winning move ♕**e8+** receives minimal probability ($< 2\%$) through layer 7, then emerges gradually from layer 8 (3.9%). The solution only narrowly leads at layer 13 (22.3% vs 20.0% for **g**×**f7+**), with the decisive separation occurring entirely in the final layer where it reaches 58.9%. Full probabilities are in Tables 13 and 14.

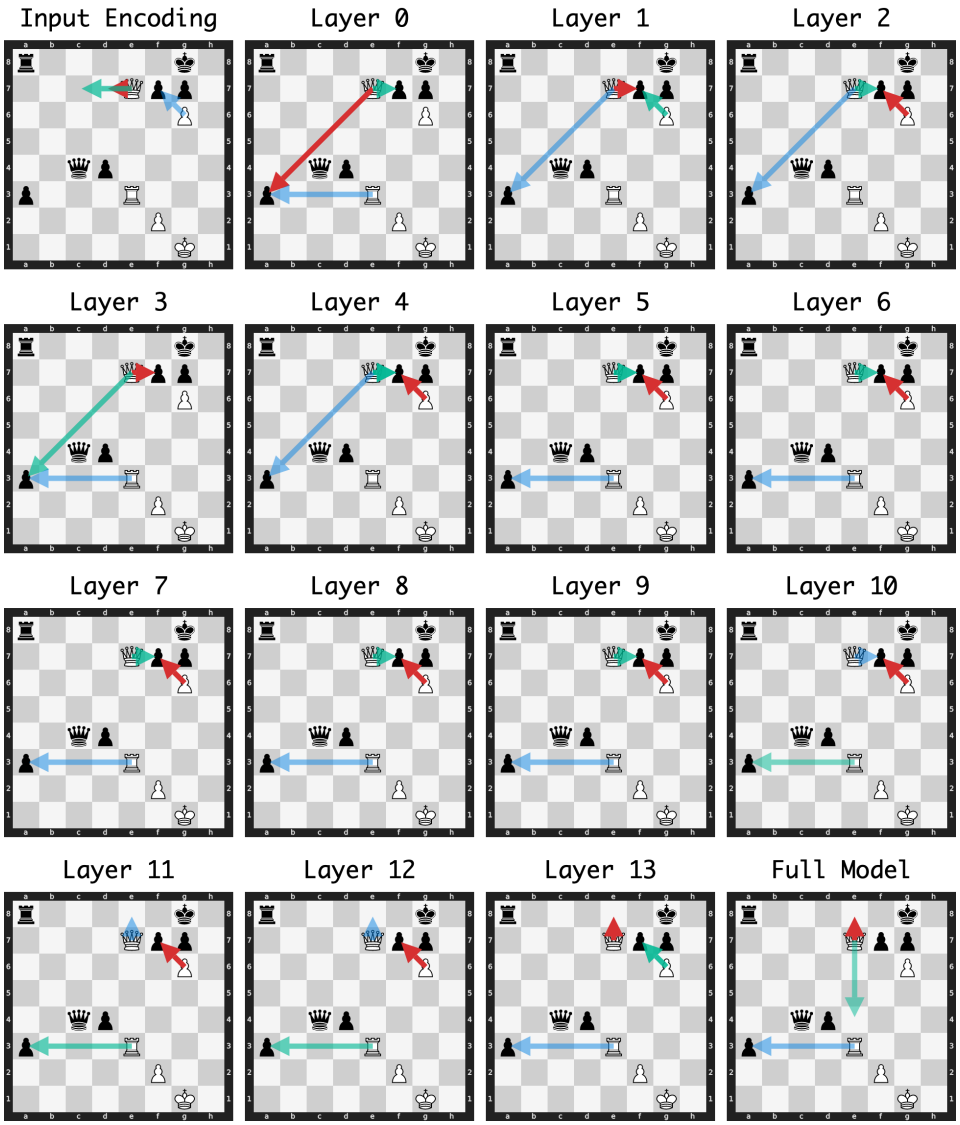


Figure 14: Layer-wise policy evolution for puzzle 09SS5.

Table 13: Move probabilities by layer for puzzle 09SS5 (Part 1: Input to Layer 6)

Move	Input	Layer 0	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
♕e8+	1.73%	0.11%	0.16%	0.24%	1.28%	0.02%	0.05%	1.11%
g×f7+	13.74%	2.54%	29.80%	33.00%	15.57%	37.26%	36.43%	43.50%
♕×f7+	4.30%	31.00%	39.29%	28.88%	32.40%	35.43%	32.63%	30.50%
♕×a3	0.43%	33.82%	15.83%	23.85%	25.18%	13.23%	13.23%	10.04%
♕d7	30.62%	0.06%	0.18%	0.05%	0.03%	0.03%	0.02%	0.02%
♖×a3	0.06%	26.89%	10.22%	8.50%	21.20%	12.43%	15.65%	11.68%
♕c7	20.69%	0.06%	0.04%	0.11%	0.06%	0.04%	0.02%	0.02%
♕e4	0.01%	0.05%	0.06%	0.12%	0.06%	0.05%	0.06%	0.03%
♖h3	0.01%	0.13%	0.08%	0.16%	0.22%	0.03%	0.38%	1.42%
♕f6	6.82%	0.05%	0.14%	0.15%	0.10%	0.06%	0.15%	0.11%
♖f3	0.04%	0.08%	0.23%	0.15%	0.09%	0.02%	0.18%	0.29%
♕b7	1.29%	0.10%	0.05%	0.04%	0.02%	0.02%	0.02%	0.02%
♕d6	4.19%	0.05%	0.18%	0.05%	0.04%	0.03%	0.03%	0.02%
♕e6	3.74%	0.06%	0.08%	0.33%	0.08%	0.03%	0.05%	0.03%
♖e4	0.05%	0.07%	0.06%	0.07%	0.12%	0.03%	0.12%	0.15%
♕g5	0.43%	2.70%	0.81%	0.21%	0.21%	0.09%	0.11%	0.07%
♕b4	2.24%	0.20%	0.20%	1.01%	0.38%	0.08%	0.03%	0.03%
♕f8+	1.79%	0.10%	0.74%	0.36%	1.50%	0.08%	0.10%	0.23%
♕d8+	1.67%	0.10%	0.21%	0.06%	0.09%	0.02%	0.02%	0.03%
♕h4	1.51%	0.27%	0.21%	0.82%	0.10%	0.05%	0.03%	0.02%
f4	0.42%	0.20%	0.07%	0.09%	0.07%	0.08%	0.04%	0.06%
♖e6	1.18%	0.07%	0.06%	0.06%	0.06%	0.03%	0.03%	0.03%
♖g3	0.03%	0.11%	0.10%	0.11%	0.09%	0.02%	0.18%	0.18%
♕a7	0.59%	0.08%	0.06%	0.05%	0.05%	0.04%	0.02%	0.01%
♕c5	0.75%	0.07%	0.04%	0.43%	0.14%	0.06%	0.03%	0.03%
♖e5	0.60%	0.08%	0.16%	0.07%	0.05%	0.04%	0.05%	0.05%
♕e5	0.59%	0.06%	0.11%	0.22%	0.09%	0.06%	0.04%	0.03%
♔g2	0.03%	0.06%	0.24%	0.39%	0.28%	0.06%	0.09%	0.08%
♖c3	0.03%	0.13%	0.08%	0.05%	0.12%	0.15%	0.04%	0.04%
♖b3	0.03%	0.20%	0.11%	0.09%	0.12%	0.12%	0.03%	0.03%
♔h1	0.08%	0.09%	0.08%	0.06%	0.04%	0.03%	0.03%	0.02%
♔h2	0.05%	0.09%	0.09%	0.11%	0.09%	0.05%	0.05%	0.05%
♖d3	0.04%	0.09%	0.06%	0.04%	0.03%	0.05%	0.03%	0.03%
♖e2	0.11%	0.06%	0.06%	0.03%	0.03%	0.05%	0.02%	0.02%
♖e1	0.05%	0.11%	0.07%	0.03%	0.02%	0.12%	0.02%	0.02%
f3	0.04%	0.06%	0.01%	0.01%	0.00%	0.00%	0.00%	0.00%

Table 14: Move probabilities by layer for puzzle 09SS5 (Part 2: Layer 7 to Final)

Move	Layer 7	Layer 8	Layer 9	Layer 10	Layer 11	Layer 12	Layer 13	Final
♕e8+	0.46%	3.87%	4.40%	7.59%	16.43%	15.84%	22.31%	58.86%
g×f7+	41.53%	37.46%	42.34%	36.43%	27.24%	36.29%	20.04%	1.46%
♕×f7+	29.73%	27.14%	28.31%	19.52%	15.49%	5.55%	1.83%	0.27%
♕×a3	10.11%	11.00%	8.21%	7.69%	4.35%	2.92%	6.87%	0.45%
♕d7	0.02%	0.01%	0.01%	0.01%	0.02%	0.02%	0.17%	0.24%
♖×a3	15.22%	16.47%	13.12%	19.88%	19.22%	20.38%	12.69%	9.17%
♕c7	0.03%	0.02%	0.04%	0.02%	0.02%	0.01%	0.26%	0.21%
♕e4	0.03%	0.07%	0.19%	0.30%	0.54%	0.87%	8.36%	9.86%
♖h3	0.93%	1.98%	1.26%	3.77%	6.90%	6.86%	6.19%	6.35%
♕f6	0.15%	0.06%	0.06%	0.07%	0.08%	0.04%	0.14%	0.18%
♖f3	0.21%	0.27%	0.19%	1.53%	5.85%	6.66%	5.66%	1.98%
♕b7	0.02%	0.01%	0.02%	0.02%	0.04%	0.20%	6.48%	4.20%
♕d6	0.03%	0.02%	0.03%	0.02%	0.02%	0.01%	0.08%	0.18%
♕e6	0.03%	0.02%	0.05%	0.03%	0.03%	0.04%	0.08%	0.18%
♖e4	0.11%	0.15%	0.07%	0.29%	1.69%	1.86%	2.77%	0.75%
♕g5	0.11%	0.06%	0.05%	0.05%	0.03%	0.02%	0.08%	0.20%
♕b4	0.07%	0.05%	0.10%	0.07%	0.05%	0.01%	0.09%	0.17%
♕f8+	0.19%	0.24%	0.27%	1.02%	0.21%	0.03%	0.41%	0.22%
♕d8+	0.02%	0.04%	0.04%	0.07%	0.21%	0.02%	0.07%	0.25%
♕h4	0.03%	0.05%	0.06%	0.06%	0.07%	0.06%	0.46%	0.71%
f4	0.15%	0.30%	0.38%	0.75%	0.40%	0.56%	1.42%	0.16%
♖e6	0.04%	0.03%	0.04%	0.05%	0.06%	0.04%	0.11%	0.24%
♖g3	0.14%	0.19%	0.10%	0.27%	0.42%	1.08%	1.05%	0.54%
♕a7	0.02%	0.01%	0.02%	0.02%	0.01%	0.01%	0.94%	0.77%
♕c5	0.04%	0.03%	0.06%	0.05%	0.04%	0.02%	0.07%	0.22%
♖e5	0.04%	0.05%	0.04%	0.05%	0.14%	0.18%	0.16%	0.21%
♕e5	0.03%	0.02%	0.02%	0.02%	0.02%	0.01%	0.08%	0.18%
♔g2	0.10%	0.07%	0.10%	0.06%	0.02%	0.01%	0.10%	0.18%
♖c3	0.12%	0.05%	0.10%	0.04%	0.09%	0.08%	0.16%	0.35%
♖b3	0.07%	0.05%	0.08%	0.06%	0.08%	0.07%	0.26%	0.19%
♔h1	0.02%	0.02%	0.03%	0.03%	0.09%	0.11%	0.13%	0.21%
♔h2	0.04%	0.05%	0.04%	0.05%	0.02%	0.01%	0.11%	0.19%
♖d3	0.06%	0.07%	0.11%	0.10%	0.09%	0.10%	0.16%	0.18%
♖e2	0.04%	0.03%	0.05%	0.02%	0.03%	0.02%	0.10%	0.17%
♖e1	0.03%	0.02%	0.02%	0.01%	0.01%	0.01%	0.12%	0.17%
f3	0.02%	0.00%	0.01%	0.00%	0.00%	0.00%	0.01%	0.15%

I.5 Bishop sacrifice for discovered attack

This puzzle (Figure 15) features a bishop sacrifice winning material via a discovered attack (PV: 1. **. . .♗h2+** 2. ♔×**h2** ♖×**c6**). The input encoding strongly favors the pawn push **a4** (57.5%). From layer 1, the immediate material capture ♖×**e3** dominates, peaking at 86.8% (layer 3) and remaining the top candidate through layer 6. The rook centralization ♖**e5** then emerges as a competitor, and the two alternate as the top choice through layers 7–11, with ♖×**e3** leading at layers 8 and 10 and ♖**e5** at layers 7, 9, and 11. The winning move ♗**h2+** receives negligible probability ($< 2\%$) through layer 5, then builds steadily from layer 6 (14.6%) to $\sim$23% across layers 8–11, before overtaking all competitors at layer 12 (51.2%) and reaching 92.9% in the final output, demonstrating a late-layer override of a simple material-gain heuristic in favor of a deeper sacrificial sequence. Full probabilities are in Tables 15 and 16.

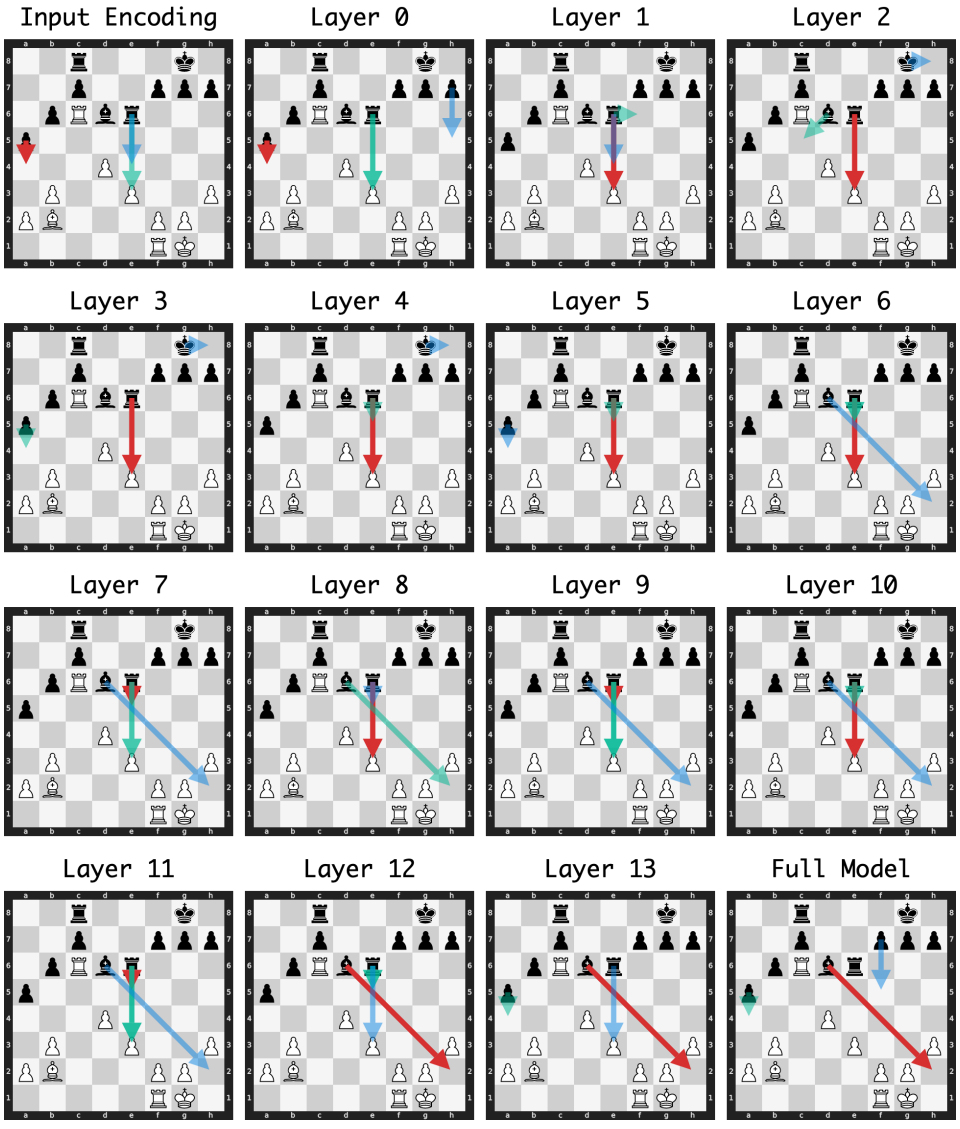


Figure 15: Layer-wise policy evolution for puzzle 7UUlN.

Table 15: Move probabilities by layer for puzzle 7UUlN (Part 1: Input to Layer 6)

Move	Input	Layer 0	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
♗h2+	0.08%	0.02%	0.54%	0.16%	0.28%	0.29%	1.79%	14.62%
♖×e3	15.44%	30.90%	82.76%	82.54%	86.84%	86.62%	76.43%	43.60%
a4	57.52%	43.79%	0.40%	0.10%	3.44%	0.02%	5.46%	3.59%
♖e5	0.49%	0.01%	0.07%	0.13%	0.44%	2.98%	9.25%	31.57%
h5	0.20%	13.90%	1.77%	0.11%	0.02%	0.00%	0.01%	0.01%
b5	2.34%	8.17%	0.40%	0.36%	0.01%	0.02%	0.08%	0.05%
♖e4	5.54%	0.02%	1.78%	0.61%	1.55%	0.30%	0.40%	0.25%
♗c5	1.33%	0.02%	0.99%	4.38%	0.59%	1.23%	0.27%	0.12%
f5	3.35%	0.57%	0.13%	0.02%	0.02%	0.00%	0.01%	0.04%
♔h8	0.63%	0.11%	1.07%	2.96%	1.91%	2.12%	0.38%	0.18%
♖f6	0.37%	0.02%	2.26%	0.32%	0.16%	0.37%	0.33%	1.45%
g5	2.25%	0.46%	0.22%	0.05%	0.01%	0.01%	0.05%	0.08%
f6	0.18%	0.13%	0.03%	0.06%	0.00%	0.01%	0.02%	0.13%
h6	0.05%	0.73%	0.13%	0.08%	0.02%	0.03%	0.07%	0.11%
♗g3	0.75%	0.03%	0.34%	0.26%	0.16%	0.30%	0.18%	0.16%
♖a8	0.74%	0.12%	0.17%	1.38%	0.30%	0.38%	0.30%	0.27%
♖g6	0.31%	0.02%	1.38%	0.28%	0.33%	0.18%	0.13%	0.11%
♖h6	0.10%	0.02%	1.29%	0.36%	0.36%	0.57%	0.40%	0.67%
♗e7	1.22%	0.01%	0.08%	0.15%	0.16%	0.18%	0.14%	0.04%
♖ee8	0.39%	0.18%	1.14%	0.88%	0.30%	0.09%	0.27%	0.09%
♗e5	1.10%	0.01%	0.21%	1.13%	0.47%	0.91%	0.37%	0.57%
♗a3	1.10%	0.02%	0.21%	0.12%	0.61%	0.24%	0.75%	0.26%
♗f4	0.99%	0.02%	0.22%	0.58%	0.17%	0.41%	0.39%	0.43%
♖ce8	0.28%	0.26%	0.99%	0.48%	0.45%	0.10%	0.45%	0.13%
g6	0.31%	0.22%	0.03%	0.04%	0.00%	0.01%	0.01%	0.01%
♖e7	0.79%	0.01%	0.44%	0.10%	0.14%	0.31%	0.25%	0.19%
♗b4	0.60%	0.02%	0.12%	0.31%	0.34%	0.32%	0.62%	0.19%
♖b8	0.53%	0.04%	0.09%	0.69%	0.19%	0.74%	0.44%	0.57%
♔f8	0.26%	0.06%	0.23%	0.39%	0.31%	0.57%	0.29%	0.16%
♖d8	0.34%	0.05%	0.21%	0.50%	0.17%	0.37%	0.25%	0.24%
♖f8	0.39%	0.05%	0.23%	0.27%	0.13%	0.17%	0.12%	0.07%
♗f8	0.04%	0.02%	0.09%	0.18%	0.11%	0.15%	0.11%	0.05%

Table 16: Move probabilities by layer for puzzle 7UUlN (Part 2: Layer 7 to Final)

Move	Layer 7	Layer 8	Layer 9	Layer 10	Layer 11	Layer 12	Layer 13	Final
♗h2+	16.54%	23.41%	22.23%	23.31%	23.15%	51.16%	67.02%	92.87%
♖×e3	30.72%	50.54%	35.04%	42.78%	31.82%	5.67%	3.56%	0.17%
a4	1.12%	0.35%	0.10%	0.27%	0.90%	1.69%	5.71%	1.08%
♖e5	48.65%	21.68%	38.76%	29.97%	35.14%	35.87%	3.34%	0.17%
h5	0.01%	0.01%	0.01%	0.01%	0.02%	0.02%	0.37%	0.26%
b5	0.10%	0.12%	0.05%	0.07%	0.37%	0.03%	1.04%	0.15%
♖e4	0.06%	0.12%	0.06%	0.12%	0.29%	0.14%	0.75%	0.16%
♗c5	0.07%	0.20%	0.15%	0.13%	0.15%	0.18%	0.72%	0.16%
f5	0.15%	0.44%	0.06%	0.06%	0.30%	0.17%	0.73%	0.71%
♔h8	0.12%	0.19%	0.29%	0.35%	0.77%	0.78%	1.39%	0.19%
♖f6	0.61%	0.29%	0.53%	0.34%	1.86%	0.40%	0.37%	0.16%
g5	0.07%	0.08%	0.07%	0.06%	0.24%	0.11%	0.63%	0.16%
f6	0.09%	0.06%	0.03%	0.05%	0.10%	0.06%	1.98%	0.18%
h6	0.04%	0.01%	0.02%	0.02%	0.18%	0.06%	1.74%	0.14%
♗g3	0.07%	0.14%	0.21%	0.27%	0.64%	0.69%	1.46%	0.19%
♖a8	0.08%	0.07%	0.07%	0.06%	0.07%	0.06%	0.26%	0.30%
♖g6	0.03%	0.04%	0.03%	0.04%	0.31%	0.09%	0.50%	0.15%
♖h6	0.24%	0.12%	0.14%	0.13%	0.80%	0.18%	0.49%	0.15%
♗e7	0.02%	0.08%	0.07%	0.08%	0.09%	0.23%	0.53%	0.16%
♖ee8	0.03%	0.03%	0.02%	0.01%	0.13%	0.09%	0.47%	0.21%
♗e5	0.19%	0.42%	0.27%	0.23%	0.80%	0.52%	0.71%	0.16%
♗a3	0.15%	0.25%	0.26%	0.16%	0.23%	0.13%	0.89%	0.16%
♗f4	0.25%	0.49%	0.59%	0.63%	0.40%	0.62%	1.07%	0.18%
♖ce8	0.08%	0.08%	0.10%	0.11%	0.17%	0.16%	0.27%	0.39%
g6	0.01%	0.00%	0.00%	0.01%	0.04%	0.01%	0.81%	0.14%
♖e7	0.04%	0.10%	0.09%	0.08%	0.21%	0.10%	0.49%	0.17%
♗b4	0.08%	0.18%	0.25%	0.13%	0.14%	0.28%	0.78%	0.15%
♖b8	0.13%	0.11%	0.15%	0.09%	0.15%	0.08%	0.40%	0.17%
♔f8	0.11%	0.15%	0.10%	0.14%	0.19%	0.14%	0.44%	0.17%
♖d8	0.06%	0.07%	0.11%	0.09%	0.12%	0.08%	0.28%	0.27%
♖f8	0.04%	0.06%	0.08%	0.12%	0.18%	0.15%	0.44%	0.17%
♗f8	0.03%	0.10%	0.07%	0.06%	0.03%	0.07%	0.36%	0.14%

J Complete probabilities for the forgotten example puzzle from Figure 1

This puzzle (Figure 16) features a mate-in-two requiring a queen sacrifice (PV: 1. . . . ♕×**f1**+ 2. ♔×**f1** ♖**d1**#). The winning move ♕×**f1**+ becomes the top candidate immediately at layer 0 (45.7%) and remains decisively the top choice through layer 13, with its primary competitor being the queen check ♕×**h2**+, which maintains a persistent secondary role (23%–31% across layers 0–11 except layer 1). The solution peaks sharply at layer 12 (88.3%) as ♕×**h2**+ drops to 3.3%, but then falls to 30.6% in layer 13 and is abandoned in the final output (5.8%). The losing move ♕**c8**—a quiet queen retreat to the back rank—receives negligible probability ($< 0.2\%$) through layer 12, then surges to 2.4% (layer 13) and 37.0% in the final output. A second quiet queen move ♕**d7** follows the same pattern, rising from 0.3% (layer 12) to 24.7% in the final output. The model replaces a winning queen sacrifice with quiet queen retreats, consistent with the safety-prior mechanism identified in the main text. The value head evaluates the current position as losing (96.0% loss) despite the mate-in-two available, yet correctly assigns near-certain victory (99.8% win) to the position after ♕×**f1**+ while evaluating all alternatives as losing (98.9%–100% loss), including the model's preferred ♕**c8** at 98.9% loss (Table 17). Full probabilities are in Tables 18 and 19.

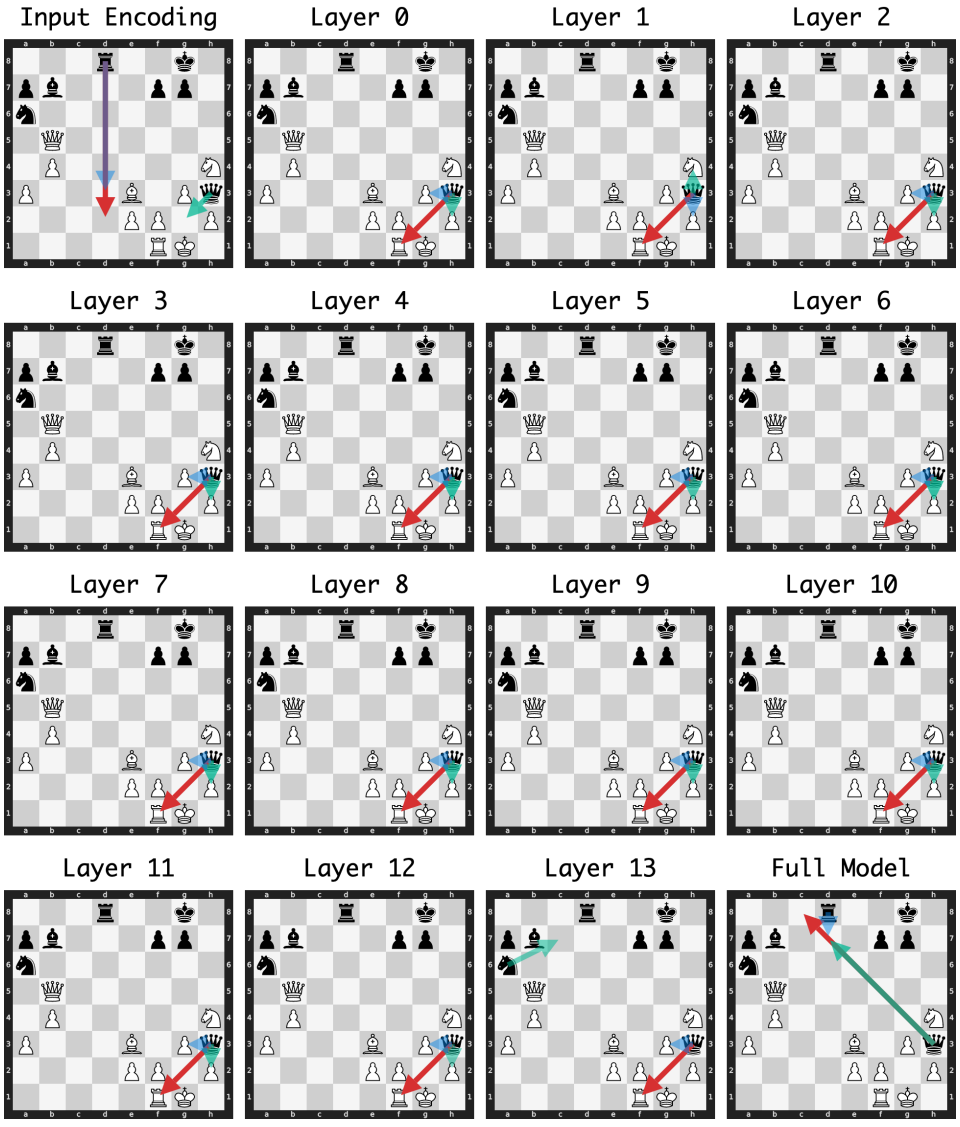


Figure 16: Layer-wise policy evolution for puzzle ANklq.

Table 17: Move evaluation for puzzle ANklq: Stockfish evaluation at depth 20 and model WDL prediction for resulting positions

Move	Stockfish	Δ (cp)	Win	Draw	Loss	Δ Win
Current position	$+\infty$	—	1.4%	2.6%	96.0%	—
♕×f1+ ⋆	$+\infty$	—	99.8%	0.1%	0.0%	+98.4%
♗a8	−4.34	$-\infty$	0.1%	0.2%	99.7%	−1.3%
♖d7	−4.36	$-\infty$	0.3%	0.6%	99.1%	−1.1%
♗c8	−4.51	$-\infty$	0.3%	0.6%	99.1%	−1.1%
♕c8	−4.87	$-\infty$	0.3%	0.7%	98.9%	−1.1%
♕d7	−4.91	$-\infty$	0.2%	0.5%	99.3%	−1.2%
♖b8	−4.95	$-\infty$	0.2%	0.3%	99.5%	−1.2%
♘c5	−4.97	$-\infty$	0.1%	0.1%	99.8%	−1.3%
♗h1	−5.23	$-\infty$	0.1%	0.2%	99.7%	−1.3%
♗e4	−5.34	$-\infty$	0.1%	0.2%	99.7%	−1.3%
♗d5	−5.37	$-\infty$	0.1%	0.2%	99.6%	−1.3%
♘×b4	−5.37	$-\infty$	0.0%	0.1%	99.9%	−1.4%
♘c7	−5.83	$-\infty$	0.0%	0.1%	99.8%	−1.4%
♘b8	−6.19	$-\infty$	0.0%	0.1%	99.9%	−1.4%
♗f3	−6.21	$-\infty$	0.0%	0.1%	99.9%	−1.4%
♗g2	−6.44	$-\infty$	0.0%	0.0%	99.9%	−1.4%
♕g4	−6.53	$-\infty$	0.0%	0.0%	99.9%	−1.4%
♖f8	−6.78	$-\infty$	0.0%	0.0%	99.9%	−1.4%
♖c8	−6.78	$-\infty$	0.0%	0.0%	100.0%	−1.4%
♕e6	−6.82	$-\infty$	0.0%	0.1%	99.8%	−1.4%
♖d6	−6.87	$-\infty$	0.0%	0.0%	100.0%	−1.4%
♗c6	−6.90	$-\infty$	0.0%	0.2%	99.8%	−1.4%
f6	−6.92	$-\infty$	0.0%	0.1%	99.9%	−1.4%
♔h8	−6.93	$-\infty$	0.0%	0.0%	100.0%	−1.4%
♔f8	−6.95	$-\infty$	0.0%	0.0%	100.0%	−1.4%
g6	−6.95	$-\infty$	0.0%	0.0%	99.9%	−1.4%
♔h7	−6.97	$-\infty$	0.0%	0.0%	99.9%	−1.4%
♖d3	−7.06	$-\infty$	0.0%	0.0%	100.0%	−1.4%
♖d5	−7.07	$-\infty$	0.0%	0.1%	99.9%	−1.4%
♖a8	−7.21	$-\infty$	0.0%	0.0%	100.0%	−1.4%
♖d2	−7.30	$-\infty$	0.0%	0.0%	100.0%	−1.4%
f5	−7.32	$-\infty$	0.0%	0.0%	100.0%	−1.4%
♕×h4	−7.72	$-\infty$	0.0%	0.1%	99.9%	−1.4%
♖d4	−7.94	$-\infty$	0.0%	0.1%	99.9%	−1.4%
♖d1	−8.33	$-\infty$	0.0%	0.4%	99.6%	−1.4%
♕×g3+	−8.48	$-\infty$	0.0%	0.0%	100.0%	−1.4%
♕g2+	−8.48	$-\infty$	0.0%	0.0%	100.0%	−1.4%
♕×h2+	−8.71	$-\infty$	0.0%	0.0%	100.0%	−1.4%
♕f5	−9.18	$-\infty$	0.0%	0.0%	100.0%	−1.4%
♖e8	−9.33	$-\infty$	0.0%	0.0%	100.0%	−1.4%
g5	−9.38	$-\infty$	0.0%	0.0%	100.0%	−1.4%

Table 18: Move probabilities by layer for puzzle ANklq (Part 1: Input to Layer 6)

Move	Input	Layer 0	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
♕×f1+	0.46%	45.73%	49.14%	46.74%	44.85%	47.54%	66.46%	47.93%
♖d2	41.09%	0.07%	0.04%	0.15%	0.44%	0.81%	0.08%	0.11%
♕c8	0.10%	0.02%	0.04%	0.03%	0.03%	0.13%	0.01%	0.02%
♕×h2+	0.51%	23.84%	13.22%	25.78%	30.20%	31.29%	23.55%	31.01%
♕g2+	26.47%	0.03%	0.37%	0.34%	0.67%	0.04%	0.08%	0.57%
♕d7	0.01%	0.01%	0.03%	0.02%	0.04%	0.05%	0.02%	0.02%
♕×h4	0.18%	13.43%	24.41%	3.60%	1.43%	0.72%	0.68%	0.59%
♕×g3+	1.06%	15.02%	9.23%	17.08%	15.67%	9.74%	5.29%	12.33%
♖d7	0.04%	0.02%	0.03%	0.04%	0.04%	0.10%	0.02%	0.04%
♖d3	12.98%	0.03%	0.02%	0.11%	0.63%	1.50%	0.27%	0.41%
♘c7	0.18%	0.19%	0.01%	0.02%	0.14%	0.05%	0.45%	0.37%
♖d1	5.56%	0.06%	0.06%	0.08%	1.16%	0.07%	0.16%	0.88%
♗c8	0.10%	0.01%	0.01%	0.01%	0.01%	0.02%	0.01%	0.01%
♘×b4	4.36%	0.04%	1.58%	3.76%	1.96%	1.45%	0.97%	0.26%
♗g2	0.11%	0.03%	0.18%	0.22%	0.27%	0.04%	0.27%	2.32%
♖d4	2.29%	0.06%	0.04%	0.11%	0.26%	0.83%	0.11%	0.23%
f5	1.00%	0.06%	0.01%	0.02%	0.01%	0.01%	0.01%	0.01%
♗d5	0.13%	0.01%	0.01%	0.02%	0.05%	0.03%	0.06%	0.06%
♗h1	0.04%	0.05%	0.08%	0.16%	0.41%	0.04%	0.57%	1.02%
♗f3	0.29%	0.16%	0.34%	0.09%	0.27%	0.07%	0.05%	0.14%
♖b8	0.17%	0.03%	0.01%	0.03%	0.05%	1.18%	0.08%	0.29%
♕g4	0.48%	0.02%	0.08%	0.05%	0.08%	0.10%	0.03%	0.04%
♖d5	0.01%	0.02%	0.05%	0.05%	0.04%	0.10%	0.03%	0.02%
♘c5	0.09%	0.20%	0.37%	0.76%	0.44%	0.46%	0.16%	0.28%
♗a8	0.02%	0.02%	0.02%	0.01%	0.01%	0.02%	0.01%	0.01%
♕e6	0.00%	0.02%	0.08%	0.05%	0.05%	0.05%	0.01%	0.01%
♖a8	0.23%	0.05%	0.01%	0.04%	0.07%	0.49%	0.04%	0.05%
♖d6	0.01%	0.02%	0.03%	0.08%	0.08%	0.80%	0.08%	0.24%
g6	0.09%	0.05%	0.01%	0.01%	0.01%	0.01%	0.00%	0.00%
♗e4	0.33%	0.01%	0.01%	0.02%	0.03%	0.04%	0.04%	0.03%
♗c6	0.05%	0.01%	0.01%	0.01%	0.02%	0.03%	0.03%	0.02%
♖c8	0.11%	0.02%	0.05%	0.03%	0.04%	0.73%	0.02%	0.03%
♕f5	0.01%	0.02%	0.04%	0.05%	0.04%	0.04%	0.02%	0.03%
g5	0.68%	0.09%	0.11%	0.20%	0.15%	0.02%	0.04%	0.16%
f6	0.09%	0.03%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%
♖e8	0.10%	0.02%	0.03%	0.02%	0.04%	0.33%	0.06%	0.12%
♘b8	0.07%	0.38%	0.07%	0.05%	0.05%	0.45%	0.13%	0.23%
♖f8	0.12%	0.02%	0.04%	0.03%	0.04%	0.27%	0.03%	0.05%
♔h7	0.11%	0.02%	0.04%	0.04%	0.08%	0.13%	0.02%	0.01%
♔h8	0.19%	0.05%	0.05%	0.03%	0.05%	0.10%	0.02%	0.02%
♔f8	0.08%	0.02%	0.04%	0.03%	0.08%	0.15%	0.02%	0.03%

Table 19: Move probabilities by layer for puzzle ANklq (Part 2: Layer 7 to Final)

Move	Layer 7	Layer 8	Layer 9	Layer 10	Layer 11	Layer 12	Layer 13	Final
♕×f1+	53.31%	51.84%	56.08%	59.27%	60.25%	88.28%	30.59%	5.79%
♖d2	0.10%	0.06%	0.12%	0.04%	0.08%	0.05%	0.60%	0.22%
♕c8	0.02%	0.02%	0.03%	0.04%	0.06%	0.07%	2.40%	37.04%
♕×h2+	30.50%	27.80%	26.32%	25.86%	23.28%	3.29%	3.05%	0.18%
♕g2+	0.50%	2.37%	1.78%	0.47%	0.58%	0.16%	1.41%	0.21%
♕d7	0.03%	0.05%	0.14%	0.30%	0.59%	0.31%	7.38%	24.66%
♕×h4	0.28%	0.19%	0.11%	0.10%	0.22%	0.25%	2.00%	0.26%
♕×g3+	6.65%	9.07%	8.91%	8.23%	6.30%	2.32%	11.04%	0.17%
♖d7	0.01%	0.01%	0.01%	0.00%	0.00%	0.02%	0.94%	13.27%
♖d3	0.47%	0.37%	0.55%	0.34%	0.52%	0.12%	0.36%	0.21%
♘c7	0.23%	0.23%	0.32%	0.20%	0.38%	0.35%	12.47%	2.19%
♖d1	1.45%	2.10%	0.87%	1.02%	3.03%	0.38%	0.80%	0.32%
♗c8	0.01%	0.02%	0.02%	0.07%	0.14%	0.35%	2.62%	4.44%
♘×b4	0.39%	0.35%	0.50%	0.29%	0.17%	0.23%	1.36%	0.30%
♗g2	2.99%	2.86%	1.29%	0.80%	0.30%	0.22%	1.28%	0.23%
♖d4	0.17%	0.19%	0.46%	0.15%	0.31%	0.24%	0.40%	0.20%
f5	0.01%	0.01%	0.02%	0.07%	0.11%	0.14%	2.19%	0.21%
♗d5	0.12%	0.16%	0.19%	0.23%	0.45%	0.60%	2.13%	0.88%
♗h1	0.68%	0.28%	0.35%	0.94%	0.44%	0.18%	1.80%	1.07%
♗f3	0.17%	0.16%	0.25%	0.31%	0.33%	0.10%	1.48%	0.20%
♖b8	0.40%	0.14%	0.07%	0.01%	0.02%	0.01%	0.27%	1.23%
♕g4	0.04%	0.05%	0.06%	0.04%	0.09%	0.06%	1.13%	0.24%
♖d5	0.02%	0.01%	0.01%	0.00%	0.00%	0.02%	1.06%	0.42%
♘c5	0.27%	0.45%	0.17%	0.24%	0.32%	0.45%	0.70%	1.05%
♗a8	0.01%	0.01%	0.01%	0.01%	0.01%	0.02%	0.33%	1.02%
♕e6	0.02%	0.05%	0.05%	0.13%	0.28%	0.55%	1.01%	0.31%
♖a8	0.06%	0.05%	0.12%	0.09%	0.15%	0.16%	0.97%	0.20%
♖d6	0.18%	0.07%	0.16%	0.05%	0.10%	0.04%	0.87%	0.21%
g6	0.01%	0.01%	0.01%	0.01%	0.02%	0.04%	0.86%	0.20%
♗e4	0.04%	0.04%	0.05%	0.06%	0.11%	0.06%	0.83%	0.59%
♗c6	0.03%	0.03%	0.04%	0.07%	0.11%	0.09%	0.76%	0.24%
♖c8	0.04%	0.06%	0.07%	0.02%	0.02%	0.03%	0.24%	0.23%
♕f5	0.02%	0.02%	0.03%	0.04%	0.09%	0.13%	0.71%	0.19%
g5	0.05%	0.06%	0.02%	0.04%	0.05%	0.04%	0.50%	0.27%
f6	0.01%	0.01%	0.01%	0.00%	0.01%	0.01%	0.67%	0.21%
♖e8	0.37%	0.59%	0.56%	0.21%	0.64%	0.28%	0.39%	0.17%
♘b8	0.19%	0.12%	0.07%	0.03%	0.05%	0.04%	0.62%	0.36%
♖f8	0.12%	0.05%	0.10%	0.05%	0.10%	0.06%	0.53%	0.21%
♔h7	0.02%	0.02%	0.02%	0.03%	0.05%	0.05%	0.49%	0.21%
♔h8	0.02%	0.02%	0.03%	0.08%	0.10%	0.09%	0.45%	0.22%
♔f8	0.03%	0.03%	0.04%	0.04%	0.12%	0.09%	0.30%	0.19%

K Forgotten puzzle case studies

K.1 Rook sacrifice for queen capture

This puzzle (Figure 17) presents a rook sacrifice winning the queen (PV: 1. ♖×**g7+** ♔×**g7** 2. ♕×**h4**). The winning move ♖×**g7+** dominates throughout nearly all layers, serving as the top candidate from layer 0 onwards (excluding layer 1) with probabilities exceeding 50% from layer 2 and peaking at 82.72% at layer 10. Its primary competitor, the queen capture ♕×**a7**, maintains 27%–52% probability through layers 0–7 before declining steadily, since this move leaves the rook hanging and gains insufficient compensation. The solution is overtaken in the final layer by ♔**f1**—a king move that receives negligible probability ($< 1\%$) throughout layers 0–10, then surges to 6.20% (layer 12), 30.42% (layer 13), and 52.98% (final output). Notably, the model's value head evaluates the current position as unfavorable (71.8% loss probability), yet when applied to the position after each legal move, correctly distinguishes the forcing sequence: it assigns near-certain victory (99.5% win) to the position after ♖×**g7+** while evaluating all alternatives as losing positions (91%–100% loss) (Table 20). Full probabilities are in Tables 21 and 22.

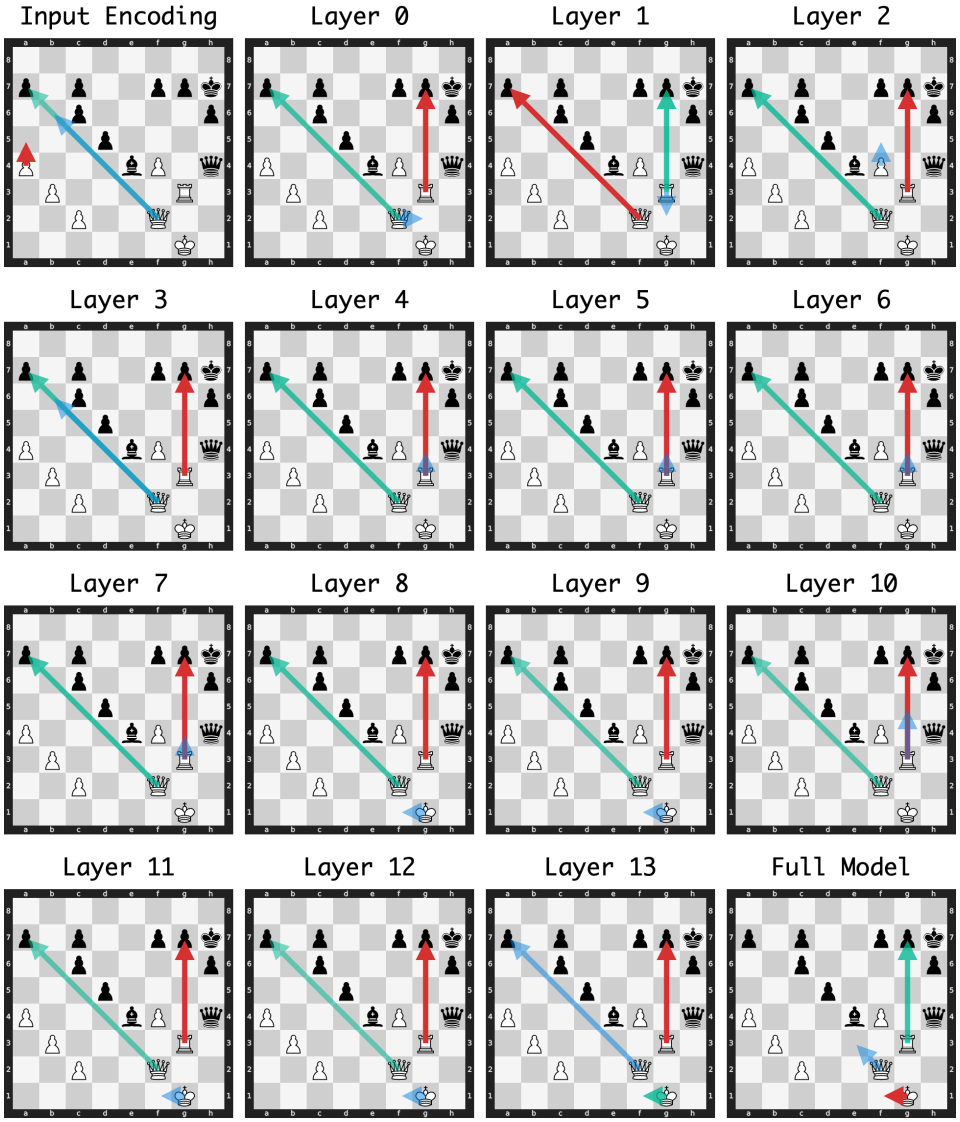


Figure 17: Layer-wise policy evolution for puzzle `58Ib0`.

Table 20: Move evaluation for puzzle `58Ib0`: Stockfish evaluation at depth 20 and model WDL prediction for resulting positions

Move	Stockfish	Δ (cp)	Win	Draw	Loss	Δ Win
Current position	+6.18	—	20.0%	8.2%	71.8%	—
♖×**g7+** ⋆	+6.15	−3	99.5%	0.4%	0.1%	+79.5%
♔**f1**	−2.47	−865	0.9%	7.9%	91.1%	−19.1%
♕**e3**	−3.01	−919	0.4%	2.5%	97.1%	−19.6%
♕**h2**	−3.29	−947	0.5%	2.9%	96.6%	−19.5%
♖**g2**	−3.64	−982	0.3%	1.5%	98.2%	−19.7%
♕**e1**	−4.53	−1071	0.2%	0.7%	99.1%	−19.8%
♖**f3**	−7.32	−1350	0.1%	0.5%	99.4%	−19.9%
♕×**a7**	−7.41	−1359	0.0%	0.0%	100.0%	−20.0%
♖**h3**	−8.01	−1419	0.0%	0.1%	99.9%	−20.0%
♕**g2**	−8.17	−1435	0.0%	0.1%	99.9%	−20.0%
♕**c5**	−8.60	−1478	0.0%	0.0%	100.0%	−20.0%
♕**f3**	−8.60	−1478	0.0%	0.0%	100.0%	−20.0%
♕**d4**	−8.64	−1482	0.0%	0.0%	100.0%	−20.0%
♕**d2**	−8.87	−1505	0.0%	0.0%	100.0%	−20.0%
♕**e2**	−9.30	−1548	0.0%	0.0%	100.0%	−20.0%
♖**g6**	−∞	−∞	0.0%	0.0%	100.0%	−20.0%
♖**g5**	−∞	−∞	0.0%	0.0%	100.0%	−20.0%
♖**g4**	−∞	−∞	0.0%	0.0%	100.0%	−20.0%
♖**e3**	−∞	−∞	0.0%	0.2%	99.8%	−20.0%
♖**d3**	−∞	−∞	0.0%	0.0%	100.0%	−20.0%
♖**c3**	−∞	−∞	0.0%	0.3%	99.6%	−19.9%
♕**b6**	−∞	−∞	0.0%	0.0%	100.0%	−20.0%
♕**f1**	−∞	−∞	0.0%	0.0%	100.0%	−20.0%
f5	−∞	−∞	0.0%	0.3%	99.6%	−19.9%
a5	−∞	−∞	0.1%	0.4%	99.6%	−19.9%
b4	−∞	−∞	0.0%	0.3%	99.6%	−19.9%
c3	−∞	−∞	0.1%	0.8%	99.1%	−19.9%
c4	−∞	−∞	0.0%	0.2%	99.8%	−20.0%

Table 21: Move probabilities by layer for puzzle 58Ib0 (Part 1: Input to Layer 6)

Move	Input	Layer 0	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
♖×g7+	2.93%	52.97%	34.97%	56.12%	51.65%	54.48%	63.68%	64.57%
a5	59.08%	3.06%	0.04%	0.01%	0.02%	0.01%	0.02%	0.05%
♔f1	0.31%	0.06%	0.34%	0.29%	0.13%	0.17%	0.26%	0.45%
♕×a7	11.45%	27.21%	52.41%	31.89%	27.95%	31.24%	33.11%	29.93%
♕g2	0.39%	10.35%	2.08%	0.34%	0.19%	0.18%	0.04%	0.05%
♕b6	6.17%	0.13%	0.19%	2.31%	8.67%	2.10%	0.45%	0.37%
f5	0.69%	1.39%	2.83%	3.29%	5.62%	0.12%	0.44%	1.04%
♖g4	0.41%	0.06%	0.09%	0.63%	2.00%	5.14%	0.71%	1.27%
♕c5	4.78%	0.10%	0.85%	0.12%	0.02%	0.06%	0.04%	0.11%
♕e3	0.92%	0.07%	0.21%	0.22%	0.07%	0.31%	0.03%	0.04%
♖g2	0.15%	2.88%	3.08%	0.20%	0.15%	0.24%	0.05%	0.08%
b4	2.64%	0.58%	0.12%	0.14%	0.01%	0.01%	0.01%	0.02%
♕h2	0.48%	0.08%	0.15%	0.07%	0.02%	0.05%	0.02%	0.02%
c4	2.27%	0.23%	0.11%	0.10%	0.00%	0.04%	0.07%	0.26%
♖g6	1.81%	0.08%	0.50%	0.91%	0.52%	0.21%	0.29%	0.24%
c3	0.40%	0.06%	0.05%	0.02%	0.00%	0.00%	0.01%	0.05%
♖g5	0.92%	0.05%	0.19%	1.21%	1.45%	1.40%	0.15%	0.51%
♖h3	0.07%	0.05%	0.09%	0.34%	0.31%	1.19%	0.15%	0.22%
♕f1	0.02%	0.07%	0.28%	0.09%	0.03%	0.08%	0.03%	0.02%
♖d3	0.26%	0.06%	0.09%	0.48%	0.34%	0.97%	0.06%	0.11%
♕d4	0.85%	0.09%	0.19%	0.22%	0.09%	0.15%	0.04%	0.10%
♕e2	0.81%	0.06%	0.20%	0.10%	0.06%	0.13%	0.02%	0.02%
♕f3	0.75%	0.06%	0.13%	0.22%	0.25%	0.23%	0.06%	0.13%
♕d2	0.66%	0.06%	0.16%	0.11%	0.08%	0.33%	0.03%	0.04%
♖c3	0.21%	0.06%	0.22%	0.13%	0.03%	0.05%	0.03%	0.03%
♖f3	0.21%	0.05%	0.09%	0.19%	0.20%	0.53%	0.12%	0.21%
♖e3	0.23%	0.05%	0.22%	0.13%	0.07%	0.40%	0.05%	0.06%
♕e1	0.15%	0.07%	0.12%	0.14%	0.07%	0.16%	0.02%	0.02%

Table 22: Move probabilities by layer for puzzle 58Ib0 (Part 2: Layer 7 to Final)

Move	Layer 7	Layer 8	Layer 9	Layer 10	Layer 11	Layer 12	Layer 13	Final
♖×g7+	62.81%	68.93%	82.50%	82.72%	80.06%	80.68%	50.84%	31.61%
a5	0.10%	0.09%	0.14%	0.08%	0.34%	0.07%	0.61%	0.26%
♔f1	0.31%	0.51%	0.55%	0.33%	1.22%	6.20%	30.42%	52.98%
♕×a7	32.68%	27.58%	14.43%	14.94%	13.42%	8.13%	2.23%	0.29%
♕g2	0.05%	0.05%	0.08%	0.08%	0.25%	0.35%	1.01%	0.33%
♕b6	0.60%	0.31%	0.17%	0.13%	0.13%	0.05%	0.27%	0.29%
f5	0.39%	0.28%	0.04%	0.05%	0.31%	0.07%	0.79%	0.23%
♖g4	0.76%	0.34%	0.29%	0.11%	0.11%	0.11%	0.29%	0.28%
♕c5	0.17%	0.15%	0.18%	0.11%	0.60%	0.23%	0.76%	0.30%
♕e3	0.07%	0.06%	0.06%	0.06%	0.16%	0.29%	1.40%	4.46%
♖g2	0.06%	0.07%	0.08%	0.07%	0.18%	0.14%	0.88%	1.53%
b4	0.08%	0.07%	0.02%	0.02%	0.08%	0.03%	0.77%	0.24%
♕h2	0.04%	0.04%	0.04%	0.03%	0.10%	0.17%	1.58%	2.60%
c4	0.35%	0.22%	0.06%	0.05%	0.18%	0.13%	0.93%	0.24%
♖g6	0.28%	0.19%	0.19%	0.24%	0.44%	0.12%	0.31%	0.34%
c3	0.10%	0.05%	0.02%	0.03%	0.15%	0.23%	1.75%	0.19%
♖g5	0.35%	0.22%	0.30%	0.34%	0.36%	0.42%	0.45%	0.29%
♖h3	0.20%	0.09%	0.11%	0.07%	0.12%	0.21%	0.31%	0.28%
♕f1	0.04%	0.04%	0.05%	0.06%	0.23%	0.79%	0.97%	0.37%
♖d3	0.08%	0.05%	0.06%	0.04%	0.14%	0.19%	0.34%	0.26%
♕d4	0.14%	0.30%	0.18%	0.14%	0.47%	0.42%	0.71%	0.30%
♕e2	0.03%	0.03%	0.03%	0.03%	0.06%	0.12%	0.33%	0.29%
♕f3	0.07%	0.08%	0.10%	0.08%	0.30%	0.31%	0.30%	0.26%
♕d2	0.03%	0.03%	0.04%	0.02%	0.05%	0.07%	0.26%	0.29%
♖c3	0.03%	0.08%	0.05%	0.03%	0.18%	0.10%	0.56%	0.27%
♖f3	0.10%	0.06%	0.09%	0.07%	0.19%	0.20%	0.38%	0.45%
♖e3	0.06%	0.07%	0.10%	0.06%	0.10%	0.05%	0.16%	0.37%
♕e1	0.03%	0.03%	0.03%	0.02%	0.05%	0.12%	0.39%	0.37%

K.2 Pawn sacrifice for queen capture

This puzzle (Figure 18) presents a pawn sacrifice forcing a queen capture (PV: 1. **...g5+** 2. **f×g5 ♕×e5**). The winning move **g5+** exhibits highly non-monotonic behavior with substantial probability jumps between consecutive layers: 0.55% to 18.46% (layers 4–5), 22.21% to 0.38% (layers 8–9), and 0.77% to 41.93% (layers 12–13). The solution briefly emerges as the top candidate at layer 13 (41.93%), but drops to fourth place in the final output (11.17%). Through most middle layers (2–11), the model strongly favors the materially conservative queen trade **♕×e5**, peaking at 89.57% (layer 4). The final layers instead prioritize queen retreats to safety: **♕d3** (29.14%), **♕c4** (25.06%), and **♕d1** (14.27%)—all receiving minimal probability through layer 11. The sharp inter-layer transitions suggest algorithmic computation of forcing sequences at layer 13, presumably through look-ahead, rather than gradual accumulation of tactical heuristics. The model's value head evaluates the current position as unfavorable (75.9% loss probability), yet correctly assigns near-certain victory (98.3% win) to the position after **g5+** while evaluating all alternatives as losing positions (89.1%–100% loss) (Table 23). Full probabilities are in Tables 24 and 25.

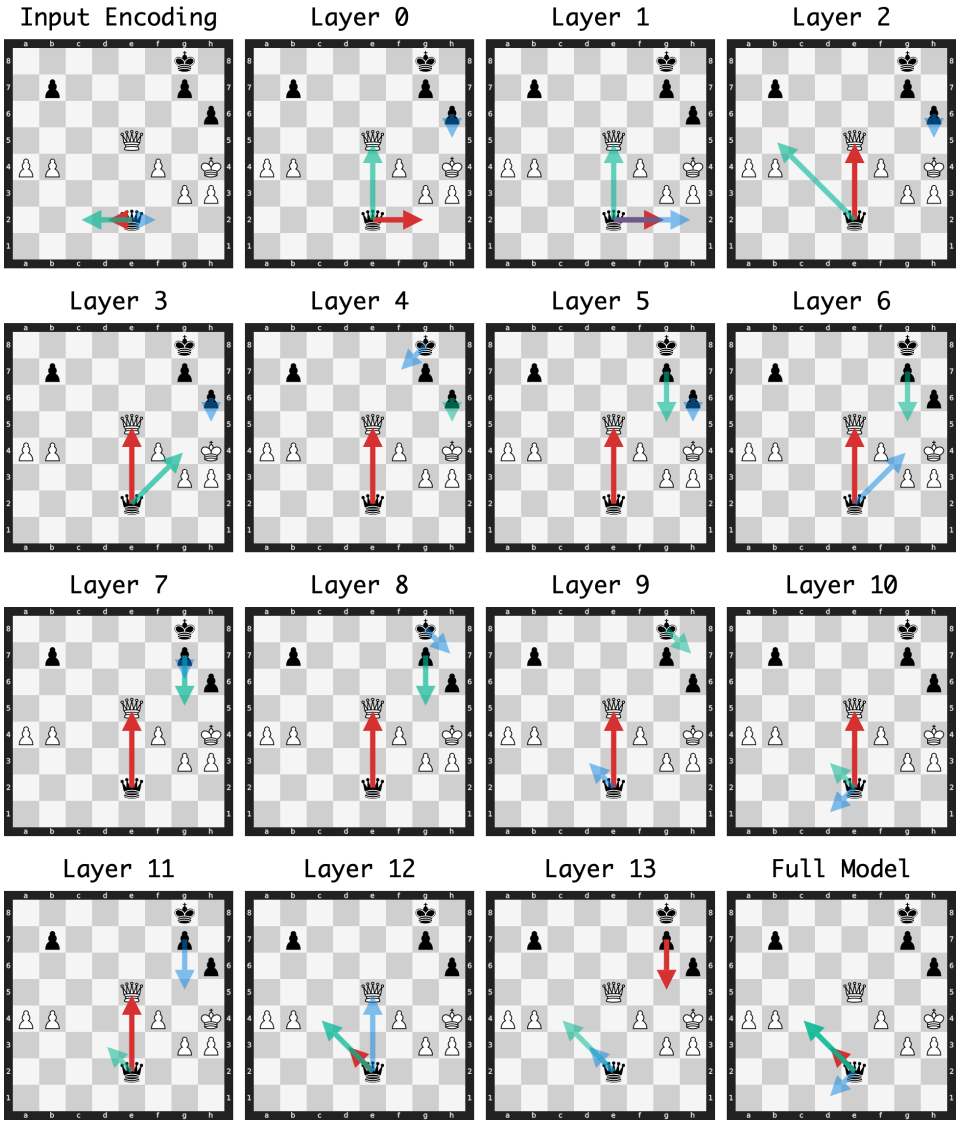


Figure 18: Layer-wise policy evolution for puzzle `6PIrs`.

Table 23: Move evaluation for puzzle `6PIrs`: Stockfish evaluation at depth 20 and model WDL prediction for resulting positions

Move	Stockfish	Δ (cp)	Win	Draw	Loss	Δ Win
Current position	+7.17	—	1.8%	22.3%	75.9%	—
g5+ ⋆	+8.54	+137	98.3%	1.5%	0.2%	+96.5%
♕d3	−2.70	−987	0.8%	9.7%	89.5%	−1.0%
♕c4	−3.09	−1026	0.8%	10.1%	89.1%	−1.0%
♕a2	−3.09	−1026	0.5%	3.6%	95.9%	−1.3%
♕f3	−3.19	−1036	0.5%	4.9%	94.6%	−1.3%
♕d2	−3.24	−1041	0.8%	8.3%	90.9%	−1.0%
♕a6	−3.26	−1043	0.5%	3.8%	95.7%	−1.3%
♕c2	−3.45	−1062	0.6%	5.1%	94.3%	−1.2%
♕d1	−3.70	−1087	0.7%	8.0%	91.4%	−1.1%
♕h2	−3.72	−1089	0.2%	0.9%	98.8%	−1.6%
♕f1	−3.84	−1101	0.3%	2.0%	97.6%	−1.5%
♕f2	−4.05	−1122	0.3%	1.5%	98.3%	−1.5%
♕g2	−4.09	−1126	0.4%	2.8%	96.8%	−1.4%
♕×e5	−6.70	−1387	0.1%	0.1%	99.8%	−1.7%
♔f8	−12.22	−1939	0.0%	0.0%	100.0%	−1.8%
♕b2	−12.29	−1946	0.0%	0.0%	100.0%	−1.8%
g6	−12.97	−2014	0.0%	0.0%	100.0%	−1.8%
♔h8	−∞	−∞	0.0%	0.1%	99.9%	−1.8%
♔h7	−∞	−∞	0.0%	0.2%	99.7%	−1.8%
♔f7	−∞	−∞	0.0%	0.0%	100.0%	−1.8%
♕h5+	−∞	−∞	0.0%	0.0%	100.0%	−1.8%
♕b5	−∞	−∞	0.0%	0.0%	100.0%	−1.8%
♕g4+	−∞	−∞	0.0%	0.0%	100.0%	−1.8%
♕e4	−∞	−∞	0.0%	0.1%	99.9%	−1.8%
♕e3	−∞	−∞	0.0%	0.0%	100.0%	−1.8%
♕e1	−∞	−∞	0.0%	0.1%	99.9%	−1.8%
b6	−∞	−∞	0.0%	0.1%	99.9%	−1.8%
h5	−∞	−∞	0.0%	0.0%	100.0%	−1.8%
b5	−∞	−∞	0.0%	0.1%	99.9%	−1.8%

Table 24: Move probabilities by layer for puzzle `6PIrs` (Part 1: Input to Layer 6)

Move	Input	Layer 0	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
♕×e5	0.03%	19.22%	16.74%	41.20%	46.85%	89.57%	57.18%	49.90%
♕g2	5.38%	51.44%	65.55%	2.96%	1.62%	0.13%	0.24%	0.19%
g5+	0.22%	1.01%	0.11%	6.18%	6.27%	0.55%	18.46%	14.62%
♕d3	6.21%	0.13%	0.13%	0.39%	0.22%	0.15%	0.37%	0.50%
♕d2	25.48%	0.14%	0.36%	0.48%	0.17%	0.11%	0.12%	0.17%
♕c4	1.15%	0.17%	0.16%	0.55%	0.24%	0.14%	0.40%	0.59%
♕g4+	0.45%	0.40%	1.46%	4.21%	21.21%	0.84%	3.49%	10.09%
♕d1	5.23%	0.17%	0.33%	0.50%	0.35%	0.15%	0.61%	1.19%
♕c2	14.22%	0.16%	0.14%	0.36%	0.24%	0.14%	0.29%	0.29%
♕b5	2.00%	0.34%	0.46%	14.07%	2.75%	0.40%	0.67%	0.65%
♕f3	7.58%	0.24%	2.69%	3.25%	0.89%	0.18%	0.93%	1.16%
h5	0.02%	10.08%	0.55%	6.22%	9.38%	2.67%	8.66%	4.21%
♕f2	8.87%	0.17%	0.43%	1.03%	0.21%	0.12%	0.22%	0.21%
♕h5+	0.59%	1.09%	1.88%	3.04%	2.39%	0.81%	3.19%	6.33%
♕h2	1.73%	6.20%	4.46%	1.02%	0.61%	0.17%	0.14%	0.13%
♕e3	5.99%	0.16%	0.14%	0.71%	0.23%	0.15%	0.11%	0.12%
♕e1	5.84%	0.26%	0.27%	0.34%	0.23%	0.12%	0.17%	0.15%
♕f1	5.49%	1.69%	2.47%	1.32%	0.30%	0.14%	0.18%	0.17%
♔h7	0.03%	0.12%	0.11%	1.70%	1.30%	0.41%	0.99%	2.21%
g6	0.03%	0.57%	0.06%	0.69%	0.11%	0.20%	1.66%	4.30%
b5	0.19%	3.90%	0.28%	4.28%	1.88%	0.19%	0.37%	0.74%
♔f7	0.01%	0.09%	0.25%	1.68%	0.81%	0.88%	0.43%	0.66%
♕b2	1.57%	0.23%	0.14%	0.36%	0.16%	0.13%	0.11%	0.11%
♕e4	1.21%	0.17%	0.11%	0.75%	0.24%	0.16%	0.16%	0.49%
b6	0.03%	0.99%	0.03%	0.07%	0.01%	0.00%	0.01%	0.01%
♕a6	0.10%	0.49%	0.28%	0.94%	0.18%	0.18%	0.26%	0.19%
♔f8	0.02%	0.10%	0.15%	0.87%	0.64%	0.69%	0.26%	0.28%
♔h8	0.06%	0.15%	0.09%	0.51%	0.35%	0.51%	0.16%	0.14%
♕a2	0.26%	0.11%	0.16%	0.32%	0.17%	0.12%	0.16%	0.18%

Table 25: Move probabilities by layer for puzzle `6PIrs` (Part 2: Layer 7 to Final)

Move	Layer 7	Layer 8	Layer 9	Layer 10	Layer 11	Layer 12	Layer 13	Final
♕×e5	53.49%	59.50%	77.55%	55.82%	56.34%	11.45%	2.52%	0.49%
♕g2	0.13%	0.07%	0.08%	0.12%	0.10%	0.12%	0.18%	0.36%
g5+	22.91%	22.21%	0.38%	0.34%	6.39%	0.77%	41.93%	11.17%
♕d3	0.88%	0.94%	2.92%	10.24%	9.81%	29.23%	13.54%	29.14%
♕d2	0.16%	0.15%	0.34%	1.66%	1.47%	4.29%	5.51%	9.92%
♕c4	1.00%	1.53%	1.49%	4.79%	5.07%	19.61%	14.48%	25.06%
♕g4+	3.88%	1.70%	0.76%	1.28%	1.32%	0.56%	0.19%	0.17%
♕d1	1.12%	0.94%	1.70%	5.72%	5.08%	11.34%	7.67%	14.27%
♕c2	0.27%	0.16%	0.20%	0.46%	0.43%	1.20%	1.15%	1.99%
♕b5	0.43%	0.88%	1.05%	1.15%	0.53%	0.47%	0.29%	0.17%
♕f3	1.21%	0.92%	0.90%	2.81%	2.88%	10.71%	3.29%	2.98%
h5	1.38%	0.99%	1.02%	0.59%	0.17%	0.16%	0.21%	0.18%
♕f2	0.13%	0.07%	0.08%	0.14%	0.14%	0.13%	0.23%	0.21%
♕h5+	2.19%	1.67%	1.97%	4.22%	2.31%	0.60%	0.17%	0.17%
♕h2	0.12%	0.09%	0.11%	0.15%	0.12%	0.11%	0.21%	0.20%
♕e3	0.09%	0.07%	0.09%	0.18%	0.25%	0.20%	0.30%	0.18%
♕e1	0.13%	0.08%	0.11%	0.21%	0.23%	0.17%	0.17%	0.18%
♕f1	0.14%	0.08%	0.10%	0.16%	0.11%	0.11%	0.21%	0.25%
♔h7	3.22%	2.93%	4.00%	4.05%	2.22%	4.85%	2.57%	0.18%
g6	4.15%	2.10%	1.11%	1.13%	0.15%	0.36%	1.07%	0.20%
b5	0.96%	1.30%	2.17%	1.52%	1.93%	0.61%	1.09%	0.19%
♔f7	0.70%	0.43%	0.43%	0.62%	0.73%	0.91%	0.30%	0.18%
♕b2	0.10%	0.07%	0.10%	0.16%	0.15%	0.13%	0.26%	0.18%
♕e4	0.46%	0.41%	0.49%	1.19%	0.89%	0.25%	0.38%	0.18%
b6	0.02%	0.02%	0.01%	0.01%	0.02%	0.01%	0.71%	0.17%
♕a6	0.17%	0.20%	0.20%	0.30%	0.22%	0.40%	0.60%	0.81%
♔f8	0.27%	0.21%	0.33%	0.42%	0.64%	0.80%	0.20%	0.17%
♔h8	0.16%	0.19%	0.22%	0.41%	0.17%	0.28%	0.28%	0.18%
♕a2	0.13%	0.09%	0.10%	0.16%	0.12%	0.16%	0.31%	0.48%

K.3 Queen sacrifice to back-rank mate

This puzzle (Figure 19) presents a forced mate-in-two sequence requiring a queen sacrifice (PV: 1. ♕×**c8+** ♗×**c8** 2. ♖**e8**#). The winning move ♕×**c8+** dominates from layers 3–13, peaking at 72.8% (layer 5). However, this solution is abandoned in the final layer, where ♕×**a7** surges to 63.7% in the output—a materially safe queen capture that does not lead to a sacrifice, receiving only 1.8%–12.4% probability in layers 4–12. The reversal may reflect a safety prior against queen sacrifices overriding mid-layer tactical calculations. The model's value head evaluates the current position as unfavorable (82.3% loss) despite the mate-in-two available, yet correctly assigns near-certain victory (100% win) to the position after ♕×**c8+** while evaluating all alternatives as near-certain losses (99%–100% loss) (Table 26). This illustrates a failure mode where sound mid-layer analysis is overwritten by conservative final-layer adjustments despite accurate position evaluation capabilities. Full probabilities are in Tables 27 and 28.

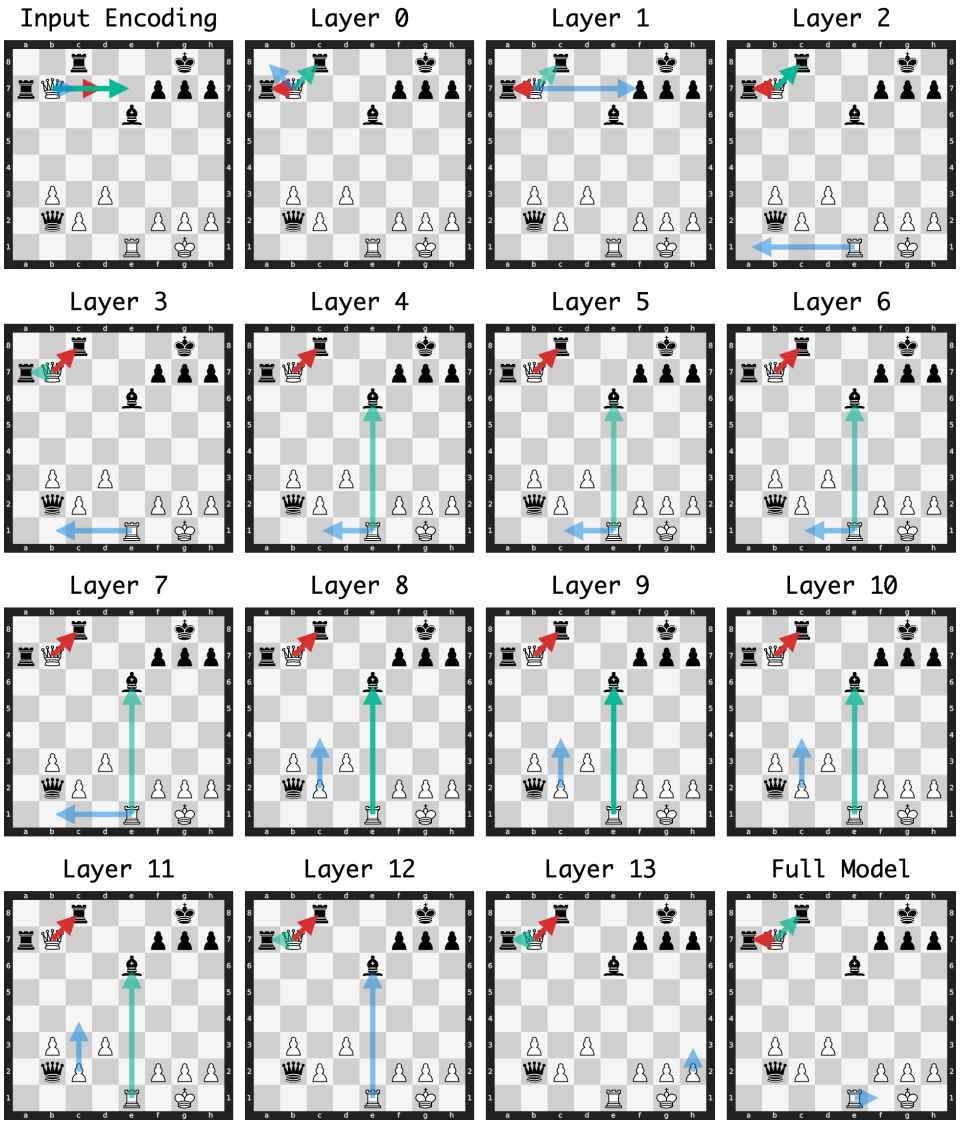


Figure 19: Layer-wise policy evolution for puzzle 1Egyn.

Table 26: Move evaluation for puzzle 1Egyn: Stockfish evaluation at depth 20 and model WDL prediction for resulting positions

Move	Stockfish	Δ (cp)	Win	Draw	Loss	Δ Win
Current position	$+\infty$	—	12.3%	5.4%	82.3%	—
♕×c8+ ⋆	$+\infty$	—	100.0%	0.0%	0.0%	+87.7%
♕×a7	−4.62	$-\infty$	0.3%	0.5%	99.2%	−12.0%
♕e4	−7.11	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♕b4	−7.23	$-\infty$	0.0%	0.0%	99.9%	−12.3%
♕c6	−7.39	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♕b6	−7.64	$-\infty$	0.0%	0.0%	99.9%	−12.3%
♕f3	−7.67	$-\infty$	0.0%	0.0%	99.9%	−12.3%
♕b5	−7.70	$-\infty$	0.0%	0.0%	100.0%	−12.3%
h4	−8.96	$-\infty$	0.0%	0.0%	100.0%	−12.3%
h3	−9.11	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♕e7	−9.17	$-\infty$	0.0%	0.0%	100.0%	−12.3%
d4	−9.18	$-\infty$	0.0%	0.0%	100.0%	−12.3%
c4	−9.22	$-\infty$	0.0%	0.0%	100.0%	−12.3%
b4	−9.33	$-\infty$	0.0%	0.0%	100.0%	−12.3%
g3	−9.33	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♕c7	−9.35	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♖f1	−9.44	$-\infty$	0.0%	0.0%	100.0%	−12.3%
g4	−9.46	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♔h1	−9.47	$-\infty$	0.0%	0.0%	100.0%	−12.3%
c3	−9.52	$-\infty$	0.0%	0.0%	100.0%	−12.3%
f3	−9.55	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♖d1	−9.66	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♔f1	−9.68	$-\infty$	0.0%	0.0%	100.0%	−12.3%
f4	−10.00	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♕a8	−10.26	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♕b8	−10.29	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♕×f7+	−11.09	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♕a6	−11.11	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♕d7	−12.92	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♕d5	$-\infty$	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♖×e6	$-\infty$	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♖e5	$-\infty$	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♖e4	$-\infty$	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♖e3	$-\infty$	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♖e2	$-\infty$	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♖c1	$-\infty$	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♖b1	$-\infty$	$-\infty$	0.0%	0.0%	100.0%	−12.3%
♖a1	$-\infty$	$-\infty$	0.0%	0.0%	100.0%	−12.3%

Table 27: Move probabilities by layer for puzzle 1Egyn (Part 1: Input to Layer 6)

Move	Input	Layer 0	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
♕×a7	0.13%	58.43%	72.93%	38.65%	12.37%	3.65%	3.45%	2.67%
♕×c8+	5.06%	35.05%	9.71%	36.67%	49.39%	54.67%	72.84%	48.69%
♖×e6	1.67%	0.17%	0.20%	0.40%	3.85%	14.24%	8.29%	13.42%
♕d7	27.16%	0.05%	0.33%	0.68%	0.40%	0.11%	0.10%	0.09%
♕e7	26.77%	0.05%	0.49%	0.36%	0.62%	0.20%	0.09%	0.20%
c4	0.33%	0.04%	0.38%	0.20%	0.31%	0.32%	0.84%	2.51%
♕c7	11.15%	0.16%	0.10%	0.08%	0.16%	0.15%	0.11%	0.15%
♕b8	7.99%	0.11%	0.37%	0.53%	1.13%	0.67%	0.40%	0.88%
♖c1	0.07%	0.04%	0.29%	1.46%	5.14%	4.71%	3.90%	7.98%
♖b1	0.10%	0.05%	0.34%	3.68%	6.77%	3.81%	2.34%	5.70%
h3	0.01%	0.05%	0.20%	0.57%	1.23%	0.19%	0.38%	4.53%
♕×f7+	2.32%	1.37%	2.62%	2.94%	2.18%	3.33%	1.76%	0.84%
♕c6	4.90%	0.05%	0.15%	0.36%	0.54%	0.21%	0.15%	0.12%
♕a8	4.40%	2.71%	1.75%	1.08%	0.67%	0.42%	0.37%	1.33%
♖a1	0.10%	0.15%	0.60%	4.11%	3.70%	3.51%	1.24%	2.08%
♖e2	0.03%	0.04%	0.24%	0.44%	1.20%	1.75%	0.41%	3.42%
♖d1	0.07%	0.05%	0.32%	0.49%	2.01%	2.44%	1.32%	1.62%
f3	0.05%	0.05%	0.37%	0.15%	0.17%	0.06%	0.04%	0.09%
♕b6	1.74%	0.04%	0.17%	0.28%	0.37%	0.24%	0.27%	0.25%
♖e5	1.63%	0.04%	0.20%	0.74%	1.06%	1.35%	0.31%	1.64%
♕d5	1.61%	0.05%	1.44%	0.27%	0.42%	0.11%	0.06%	0.03%
♖e4	0.01%	0.04%	0.41%	0.20%	0.27%	1.05%	0.11%	0.29%
d4	0.15%	0.06%	0.73%	1.03%	0.57%	0.08%	0.08%	0.08%
c3	0.06%	0.09%	0.96%	0.39%	0.27%	0.06%	0.02%	0.02%
h4	0.03%	0.12%	0.17%	0.27%	0.35%	0.08%	0.08%	0.18%
♔h1	0.12%	0.21%	0.87%	0.29%	0.49%	0.39%	0.08%	0.11%
♕b5	0.37%	0.04%	0.24%	0.37%	0.69%	0.22%	0.14%	0.10%
♖e3	0.00%	0.04%	0.20%	0.39%	0.36%	0.67%	0.05%	0.07%
♕b4	0.37%	0.07%	0.47%	0.51%	0.60%	0.14%	0.10%	0.05%
♕a6	0.03%	0.05%	0.15%	0.27%	0.59%	0.34%	0.25%	0.30%
b4	0.38%	0.09%	0.36%	0.46%	0.34%	0.08%	0.07%	0.11%
g4	0.38%	0.09%	0.57%	0.17%	0.11%	0.03%	0.01%	0.01%
f4	0.57%	0.05%	0.29%	0.17%	0.13%	0.03%	0.02%	0.04%
♕f3	0.03%	0.05%	0.37%	0.34%	0.44%	0.13%	0.07%	0.07%
♔f1	0.05%	0.09%	0.24%	0.31%	0.42%	0.27%	0.05%	0.07%
g3	0.06%	0.04%	0.25%	0.15%	0.23%	0.07%	0.07%	0.10%
♕e4	0.03%	0.04%	0.33%	0.36%	0.27%	0.14%	0.08%	0.14%
♖f1	0.07%	0.05%	0.19%	0.19%	0.15%	0.05%	0.03%	0.03%

Table 28: Move probabilities by layer for puzzle 1Egyn (Part 2: Layer 7 to Final)

Move	Layer 7	Layer 8	Layer 9	Layer 10	Layer 11	Layer 12	Layer 13	Final
♕×a7	1.84%	6.79%	5.08%	3.98%	5.37%	12.37%	23.09%	63.67%
♕×c8+	47.17%	30.34%	31.97%	42.62%	58.43%	61.50%	54.05%	28.66%
♖×e6	13.02%	25.77%	29.43%	29.42%	20.49%	10.78%	3.06%	0.16%
♕d7	0.06%	0.07%	0.20%	0.12%	0.09%	0.06%	0.12%	0.21%
♕e7	0.10%	0.11%	0.19%	0.08%	0.07%	0.06%	0.23%	0.22%
c4	3.99%	9.02%	11.81%	16.30%	10.39%	8.67%	1.58%	0.23%
♕c7	0.10%	0.09%	0.21%	0.13%	0.08%	0.04%	0.13%	0.20%
♕b8	1.38%	0.85%	1.00%	0.39%	0.43%	0.23%	0.16%	0.19%
♖c1	6.78%	0.34%	1.03%	0.51%	0.47%	0.54%	0.14%	0.22%
♖b1	7.56%	4.31%	3.81%	0.66%	0.49%	0.39%	0.25%	0.20%
h3	7.37%	7.22%	2.36%	1.75%	1.06%	3.46%	6.99%	0.23%
♕×f7+	1.35%	5.57%	2.49%	1.07%	0.35%	0.28%	1.52%	0.19%
♕c6	0.07%	0.08%	0.16%	0.09%	0.07%	0.05%	0.45%	0.21%
♕a8	1.16%	0.45%	1.08%	0.41%	0.33%	0.16%	0.10%	0.21%
♖a1	2.89%	3.44%	1.64%	0.32%	0.23%	0.24%	0.18%	0.22%
♖e2	1.88%	0.45%	0.79%	0.13%	0.06%	0.09%	0.27%	0.22%
♖d1	0.76%	1.58%	2.81%	0.33%	0.19%	0.06%	0.13%	0.21%
f3	0.35%	0.31%	0.24%	0.02%	0.03%	0.04%	1.76%	0.20%
♕b6	0.21%	0.19%	0.28%	0.11%	0.07%	0.04%	0.19%	0.23%
♖e5	0.58%	0.98%	1.41%	0.63%	0.77%	0.26%	0.37%	0.21%
♕d5	0.02%	0.03%	0.05%	0.04%	0.02%	0.04%	0.17%	0.21%
♖e4	0.11%	0.07%	0.14%	0.03%	0.03%	0.05%	0.62%	0.21%
d4	0.06%	0.23%	0.07%	0.03%	0.02%	0.02%	0.32%	0.21%
c3	0.02%	0.04%	0.08%	0.06%	0.03%	0.02%	0.12%	0.24%
h4	0.24%	0.33%	0.23%	0.06%	0.08%	0.15%	0.94%	0.22%
♔h1	0.09%	0.13%	0.33%	0.36%	0.09%	0.08%	0.25%	0.18%
♕b5	0.07%	0.09%	0.14%	0.05%	0.03%	0.03%	0.16%	0.23%
♖e3	0.03%	0.04%	0.04%	0.01%	0.00%	0.02%	0.12%	0.22%
♕b4	0.03%	0.07%	0.05%	0.01%	0.01%	0.01%	0.15%	0.22%
♕a6	0.19%	0.11%	0.25%	0.04%	0.03%	0.02%	0.21%	0.23%
b4	0.13%	0.39%	0.10%	0.02%	0.01%	0.01%	0.58%	0.22%
g4	0.01%	0.01%	0.02%	0.01%	0.01%	0.02%	0.13%	0.19%
f4	0.04%	0.08%	0.18%	0.08%	0.05%	0.08%	0.35%	0.18%
♕f3	0.09%	0.19%	0.12%	0.03%	0.03%	0.02%	0.24%	0.23%
♔f1	0.05%	0.05%	0.05%	0.03%	0.01%	0.02%	0.15%	0.23%
g3	0.08%	0.07%	0.07%	0.01%	0.03%	0.05%	0.39%	0.19%
♕e4	0.07%	0.09%	0.08%	0.04%	0.04%	0.04%	0.26%	0.24%
♖f1	0.05%	0.02%	0.02%	0.01%	0.00%	0.02%	0.08%	0.25%

K.4 Rook sacrifice for material gain

This puzzle (Figure 20) presents a rook sacrifice forcing material advantage (PV: 1. **...♖×g4+** 2. **♔×g4 ♔×e6**). The winning move **♖×g4+** dominates from layers 4–13, maintaining probabilities above 61% throughout and peaking at 88% at layer 10. However, the solution drops dramatically in the final layer to third place (12.14%), overtaken by **♖f1+** (40.4%) and **♖h1** (23%)—two materially conservative rook moves that receive minimal probability ($< 8\%$) through layers 0–12. Unlike previous cases, these alternative moves are strategically sound rather than clearly losing: Stockfish evaluates **♖f1+** at -0.05 pawns and **♖h1** at -0.15 pawns, both near-neutral positions. The model's value head partly reflects this when applied to the position after each move: while correctly assigning near-certain victory (97.9% win) to the position after **♖×g4+**, it evaluates the alternatives as likely losing (**♖f1+**: 73.6% loss, **♖h1**: 63.3% loss), rather than the near-certain losses (89%+) seen in previous puzzles (Table 29). This represents the most substantial late-layer reversal observed (88% to 12.14%) besides the puzzle in Figure 1, demonstrating that the safety prior against piece sacrifices can override even stronger mid-layer confidence than in previous cases. Full probabilities are in Tables 30 and 31.

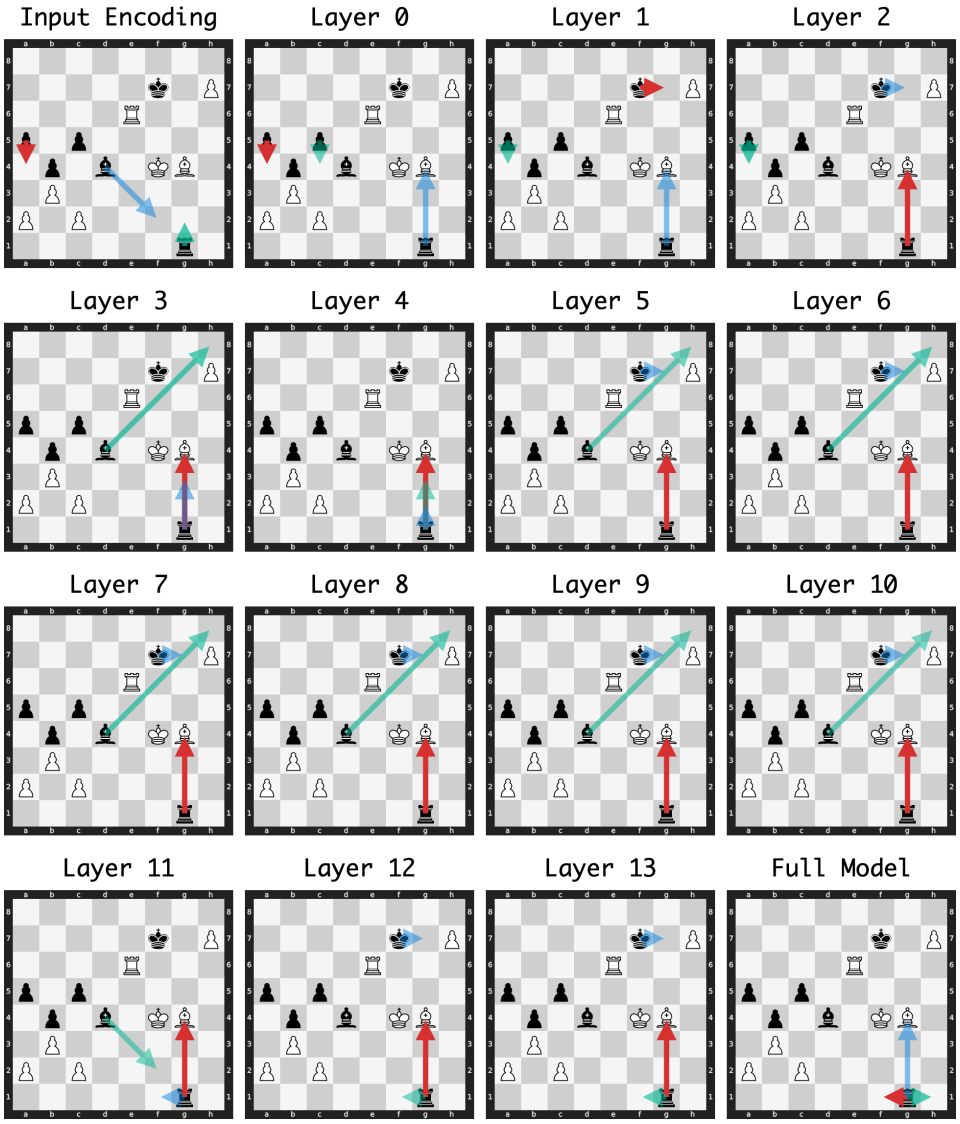


Figure 20: Layer-wise policy evolution for puzzle DsaGi.

Table 29: Move evaluation for puzzle DsaGi: Stockfish evaluation at depth 20 and model WDL prediction for resulting positions

Move	Stockfish	Δ (cp)	Win	Draw	Loss	Δ Win
Current position	+5.76	—	4.5%	23.1%	72.4%	—
♖×**g4+** ⋆	+6.10	+34	97.9%	1.9%	0.2%	+93.4%
♖**f1+**	−0.05	−581	0.8%	25.6%	73.6%	−3.7%
♖**h1**	−0.15	−591	0.9%	35.8%	63.3%	−3.7%
♔**g7**	−2.46	−822	0.6%	8.9%	90.5%	−4.0%
♖**c1**	−2.75	−851	0.3%	3.7%	96.0%	−4.2%
♖**d1**	−2.77	−853	0.1%	0.9%	99.0%	−4.4%
♖**a1**	−2.81	−857	0.2%	1.9%	97.9%	−4.3%
♖**b1**	−2.86	−862	0.2%	1.3%	98.5%	−4.3%
♗**c3**	−3.28	−904	0.2%	2.4%	97.3%	−4.3%
♗**b2**	−3.48	−924	0.3%	4.1%	95.6%	−4.2%
♗**g7**	−3.50	−926	0.2%	1.1%	98.7%	−4.4%
♗**h8**	−3.56	−932	0.1%	0.9%	98.9%	−4.4%
♔**f8**	−3.63	−939	0.1%	0.7%	99.1%	−4.4%
♖**g2**	−3.64	−940	0.3%	5.0%	94.7%	−4.2%
♗**a1**	−3.75	−951	0.2%	1.1%	98.8%	−4.4%
a4	−3.75	−951	0.5%	13.1%	86.4%	−4.1%
c4	−3.88	−964	0.2%	1.4%	98.4%	−4.4%
♗**e5+**	−7.19	−1295	0.0%	0.0%	99.9%	−4.5%
♖**g3**	−7.42	−1318	0.0%	0.1%	99.9%	−4.5%
♖**e1**	−7.42	−1318	0.0%	0.1%	99.9%	−4.5%
♗**f6**	−7.49	−1325	0.0%	0.1%	99.8%	−4.5%
♗**e3+**	−7.88	−1364	0.0%	0.1%	99.9%	−4.5%
♗**f2**	−9.17	−1493	0.0%	0.1%	99.9%	−4.5%

Table 30: Move probabilities by layer for puzzle `DsaGi` (Part 1: Input to Layer 6)

Move	Input	Layer 0	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
a4	15.39%	88.66%	14.83%	15.77%	10.39%	0.24%	1.31%	0.93%
♖×g4+	0.38%	0.66%	11.54%	22.23%	29.10%	71.66%	80.50%	75.47%
♖f1+	3.45%	0.12%	1.81%	7.09%	4.04%	0.47%	0.29%	0.60%
♔g7	0.12%	0.31%	25.84%	10.54%	5.47%	1.05%	2.15%	1.43%
♖h1	5.32%	0.13%	2.62%	2.47%	3.66%	0.35%	0.57%	1.02%
♗h8	0.13%	0.06%	0.54%	0.51%	18.57%	1.73%	8.11%	17.35%
♖g3	7.11%	0.10%	4.71%	10.33%	14.05%	6.37%	0.68%	0.31%
♖g2	13.65%	0.19%	2.17%	3.87%	3.22%	4.06%	0.38%	0.28%
♗f2	12.36%	0.08%	1.08%	4.40%	0.77%	1.30%	0.27%	0.21%
c4	1.25%	8.55%	10.83%	2.58%	0.51%	0.05%	0.51%	0.43%
♗e3+	9.81%	0.05%	1.11%	2.74%	0.73%	0.76%	0.12%	0.05%
♗c3	9.74%	0.07%	1.04%	0.38%	0.49%	0.37%	1.61%	0.20%
♖a1	3.41%	0.07%	5.73%	2.58%	0.84%	0.75%	0.18%	0.11%
♖b1	4.89%	0.14%	2.50%	1.50%	1.51%	2.38%	0.56%	0.18%
♗e5+	0.42%	0.05%	2.37%	4.46%	0.68%	1.05%	0.15%	0.06%
♗g7	0.45%	0.29%	3.97%	0.22%	0.58%	0.22%	0.98%	0.37%
♖e1	3.29%	0.08%	0.60%	1.26%	1.12%	1.86%	0.27%	0.22%
♖d1	3.17%	0.11%	0.97%	1.12%	0.92%	1.85%	0.31%	0.16%
♖c1	1.29%	0.08%	2.23%	1.43%	0.90%	1.35%	0.28%	0.19%
♗a1	2.02%	0.04%	1.18%	0.96%	1.16%	0.74%	0.29%	0.15%
♗b2	1.62%	0.06%	0.74%	1.25%	0.42%	0.65%	0.09%	0.03%
♔f8	0.00%	0.07%	1.25%	1.61%	0.70%	0.48%	0.28%	0.19%
♗f6	0.73%	0.04%	0.34%	0.69%	0.16%	0.24%	0.11%	0.03%

Table 31: Move probabilities by layer for puzzle `DsaGi` (Part 2: Layer 7 to Final)

Move	Layer 7	Layer 8	Layer 9	Layer 10	Layer 11	Layer 12	Layer 13	Final
a4	0.96%	0.48%	0.04%	0.20%	0.86%	0.64%	1.17%	0.72%
♖×g4+	72.17%	73.29%	83.14%	88.00%	78.25%	86.89%	61.03%	12.14%
♖f1+	1.23%	2.08%	0.56%	2.04%	4.41%	3.45%	16.89%	40.43%
♔g7	3.84%	3.85%	2.77%	2.18%	3.21%	3.16%	5.34%	8.01%
♖h1	0.84%	1.16%	0.30%	0.87%	1.89%	1.41%	4.99%	23.00%
♗h8	17.62%	14.62%	9.44%	3.12%	1.55%	0.36%	0.52%	0.50%
♖g3	0.40%	0.47%	0.85%	0.85%	1.88%	0.49%	0.26%	0.19%
♖g2	0.32%	0.37%	0.27%	0.35%	0.51%	0.20%	0.52%	5.38%
♗f2	0.24%	0.53%	0.78%	1.01%	4.98%	1.25%	0.60%	0.23%
c4	0.35%	0.06%	0.02%	0.04%	0.16%	0.20%	0.59%	0.31%
♗e3+	0.05%	0.06%	0.04%	0.03%	0.02%	0.00%	0.15%	0.27%
♗c3	0.05%	0.13%	0.05%	0.02%	0.02%	0.03%	0.12%	1.32%
♖a1	0.08%	0.09%	0.05%	0.04%	0.08%	0.03%	0.12%	1.05%
♖b1	0.24%	0.29%	0.07%	0.24%	0.43%	0.25%	2.95%	0.36%
♗e5+	0.06%	0.10%	0.05%	0.03%	0.04%	0.01%	0.09%	0.24%
♗g7	0.52%	1.22%	0.67%	0.26%	0.24%	0.28%	1.36%	0.50%
♖e1	0.21%	0.27%	0.39%	0.22%	0.56%	0.61%	0.17%	0.20%
♖d1	0.16%	0.19%	0.08%	0.11%	0.16%	0.12%	2.18%	0.32%
♖c1	0.21%	0.20%	0.13%	0.10%	0.23%	0.15%	0.29%	2.40%
♗a1	0.08%	0.20%	0.13%	0.05%	0.14%	0.13%	0.13%	0.28%
♗b2	0.02%	0.03%	0.02%	0.02%	0.03%	0.04%	0.13%	1.62%
♔f8	0.31%	0.28%	0.15%	0.22%	0.35%	0.30%	0.27%	0.27%
♗f6	0.04%	0.04%	0.02%	0.01%	0.01%	0.02%	0.14%	0.27%

K.5 Narrowly forgotten queen sacrifice

This puzzle (Figure 21) presents a forced mate-in-two sequence requiring a queen sacrifice (PV: 1. **...** ♕**×b1+** 2. ♔**×b1** ♖**a1**#). The winning move ♕**×b1+** leads from layer 8 through layer 13, maintaining probabilities between 18.87% and 62.90% from layers 0–12 and rising again to 56.96% in layer 13. However, in the final layer, the materially conservative pawn advance **b5** marginally overtakes the solution (36.12% vs 35.99%), despite receiving only 8.85% probability in the preceding layer. This final-layer reversal demonstrates another instance of a safety prior against queen sacrifices overriding established tactical analysis, though the margin is narrower than in previous cases. Notably, the model's value head evaluates the current position as unfavorable (81.1% loss probability), yet when applied to the position after each legal move, correctly assigns near-certain victory (99.3% win) to the position after ♕**×b1+** while evaluating all alternatives as losing positions (99.4%–100% loss) (Table 32). Full probabilities are in Tables 33 and 34.

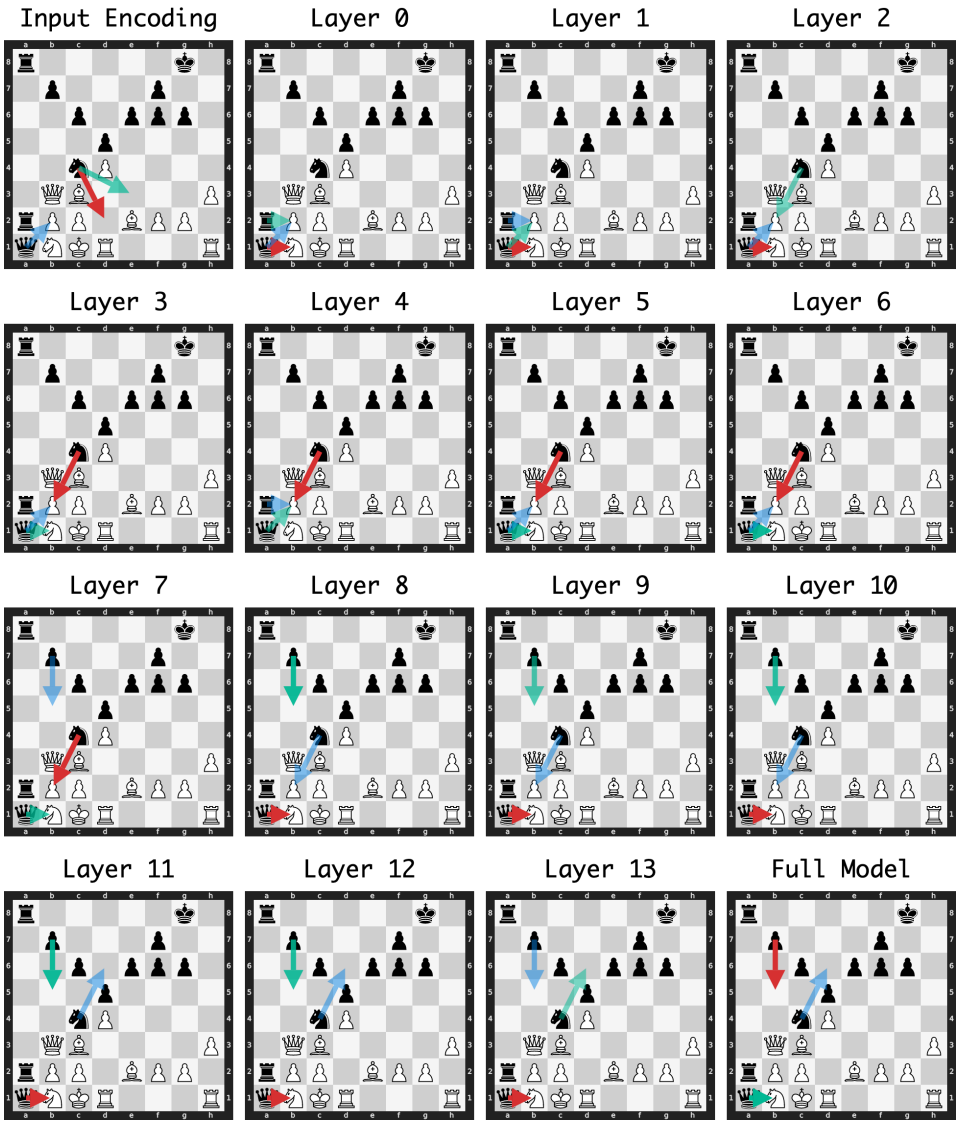


Figure 21: Layer-wise policy evolution for puzzle `00aDl`.

Table 32: Move evaluation for puzzle `00aD1`: Stockfish evaluation at depth 20 and model WDL prediction for resulting positions

Move	Stockfish	Δ (cp)	Win	Draw	Loss	Δ Win
Current position	$+\infty$	—	8.3%	10.6%	81.1%	—
♕×b1+ ⋆	$+\infty$	—	99.3%	0.5%	0.2%	+91.0%
b5	−4.12	$-\infty$	0.2%	0.4%	99.4%	−8.1%
♖2a4	−4.57	$-\infty$	0.2%	0.3%	99.5%	−8.1%
♖2a7	−4.72	$-\infty$	0.1%	0.1%	99.8%	−8.2%
♘a5	−4.74	$-\infty$	0.0%	0.1%	99.9%	−8.3%
♖2a6	−4.77	$-\infty$	0.1%	0.1%	99.8%	−8.2%
c5	−4.78	$-\infty$	0.0%	0.1%	99.9%	−8.3%
♘d6	−5.07	$-\infty$	0.1%	0.2%	99.7%	−8.2%
♔g7	−5.40	$-\infty$	0.0%	0.1%	99.9%	−8.3%
b6	−5.42	$-\infty$	0.0%	0.1%	99.9%	−8.3%
e5	−5.45	$-\infty$	0.0%	0.1%	99.9%	−8.3%
♔h8	−5.49	$-\infty$	0.0%	0.1%	99.9%	−8.3%
f5	−5.49	$-\infty$	0.0%	0.1%	99.9%	−8.3%
g5	−5.53	$-\infty$	0.0%	0.1%	99.9%	−8.3%
♖2a5	−5.55	$-\infty$	0.0%	0.1%	99.9%	−8.3%
♖8a4	−5.56	$-\infty$	0.1%	0.2%	99.8%	−8.2%
♖8a6	−5.60	$-\infty$	0.0%	0.1%	99.9%	−8.3%
♖8a5	−5.60	$-\infty$	0.0%	0.1%	99.9%	−8.3%
♖c8	−5.63	$-\infty$	0.0%	0.1%	99.9%	−8.3%
♖d8	−5.75	$-\infty$	0.0%	0.1%	99.9%	−8.3%
♖8a7	−5.75	$-\infty$	0.0%	0.1%	99.9%	−8.3%
♖b8	−5.76	$-\infty$	0.0%	0.1%	99.9%	−8.3%
♔f8	−5.82	$-\infty$	0.0%	0.1%	99.9%	−8.3%
♔h7	−5.87	$-\infty$	0.0%	0.1%	99.9%	−8.3%
♘e5	−5.96	$-\infty$	0.0%	0.0%	99.9%	−8.3%
♖f8	−5.98	$-\infty$	0.0%	0.0%	99.9%	−8.3%
♖e8	−6.17	$-\infty$	0.0%	0.1%	99.9%	−8.3%
♘e3	−6.29	$-\infty$	0.0%	0.0%	99.9%	−8.3%
♘b6	−6.78	$-\infty$	0.0%	0.0%	99.9%	−8.3%
♘×b2	−7.09	$-\infty$	0.0%	0.1%	99.9%	−8.3%
♕×b2+	−7.11	$-\infty$	0.0%	0.0%	100.0%	−8.3%
♖×b2	−7.23	$-\infty$	0.0%	0.1%	99.8%	−8.3%
♘a3	−7.65	$-\infty$	0.0%	0.0%	99.9%	−8.3%
♘d2	−7.77	$-\infty$	0.0%	0.0%	100.0%	−8.3%
♖2a3	−8.16	$-\infty$	0.0%	0.0%	100.0%	−8.3%
♖8a3	−8.34	$-\infty$	0.0%	0.1%	99.9%	−8.3%

Table 33: Move probabilities by layer for puzzle `00aDl` (Part 1: Input to Layer 6)

Move	Input	Layer 0	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
♕×b1+	1.80%	62.90%	47.54%	29.77%	23.48%	18.87%	27.94%	27.42%
♘d2	56.64%	0.41%	0.72%	0.18%	2.89%	0.03%	1.47%	0.89%
b5	0.43%	0.31%	0.18%	0.03%	0.37%	0.13%	2.56%	12.77%
♘×b2	1.28%	0.66%	0.69%	24.30%	29.59%	35.56%	31.40%	28.96%
♘e3	26.36%	0.10%	0.37%	0.02%	0.04%	0.06%	0.02%	0.02%
♕×b2+	8.75%	11.39%	23.80%	24.03%	20.94%	22.84%	19.88%	13.99%
♖×b2	0.35%	17.28%	18.29%	20.38%	20.92%	20.66%	13.81%	13.55%
♘d6	0.09%	0.11%	0.09%	0.05%	0.17%	0.08%	0.57%	0.33%
♘a5	0.58%	0.59%	0.87%	0.03%	0.27%	0.10%	1.49%	0.23%
b6	0.06%	0.06%	0.12%	0.02%	0.01%	0.05%	0.14%	0.62%
♘a3	0.14%	0.35%	1.00%	0.03%	0.16%	0.05%	0.05%	0.18%
♖2a4	0.03%	0.08%	0.13%	0.15%	0.09%	0.03%	0.02%	0.06%
♖8a4	0.07%	0.13%	0.73%	0.04%	0.01%	0.02%	0.01%	0.03%
♔f8	0.05%	2.16%	0.25%	0.03%	0.09%	0.08%	0.02%	0.04%
♖8a3	0.05%	0.14%	1.61%	0.15%	0.04%	0.11%	0.03%	0.06%
♖8a5	0.05%	0.20%	0.90%	0.05%	0.03%	0.07%	0.04%	0.09%
♖2a6	0.12%	0.12%	0.15%	0.03%	0.05%	0.12%	0.02%	0.02%
f5	0.80%	0.05%	0.07%	0.01%	0.02%	0.03%	0.03%	0.06%
♔g7	0.04%	0.24%	0.05%	0.25%	0.15%	0.07%	0.03%	0.04%
♖2a3	0.01%	0.07%	0.12%	0.03%	0.06%	0.08%	0.03%	0.08%
♖2a7	0.09%	0.09%	0.12%	0.04%	0.04%	0.05%	0.01%	0.02%
c5	0.39%	0.26%	0.14%	0.07%	0.03%	0.02%	0.02%	0.05%
♖2a5	0.09%	0.08%	0.07%	0.03%	0.07%	0.07%	0.05%	0.05%
♔h8	0.14%	0.23%	0.19%	0.04%	0.08%	0.09%	0.03%	0.04%
♖8a6	0.03%	0.13%	0.35%	0.04%	0.04%	0.17%	0.03%	0.06%
e5	0.21%	0.04%	0.15%	0.00%	0.04%	0.03%	0.04%	0.06%
♖f8	0.08%	0.52%	0.13%	0.02%	0.01%	0.02%	0.01%	0.02%
♔h7	0.08%	0.46%	0.05%	0.05%	0.13%	0.10%	0.03%	0.03%
g5	0.49%	0.41%	0.29%	0.01%	0.04%	0.01%	0.03%	0.05%
♖b8	0.05%	0.05%	0.10%	0.02%	0.02%	0.06%	0.01%	0.02%
♖8a7	0.04%	0.09%	0.17%	0.02%	0.02%	0.05%	0.01%	0.02%
♘b6	0.31%	0.09%	0.14%	0.02%	0.02%	0.10%	0.09%	0.06%
♖d8	0.06%	0.05%	0.10%	0.02%	0.02%	0.04%	0.01%	0.02%
♖e8	0.06%	0.05%	0.09%	0.02%	0.01%	0.03%	0.01%	0.01%
♖c8	0.05%	0.06%	0.13%	0.02%	0.02%	0.06%	0.01%	0.02%
♘e5	0.12%	0.04%	0.10%	0.01%	0.01%	0.05%	0.03%	0.04%

Table 34: Move probabilities by layer for puzzle 00aDl (Part 2: Layer 7 to Final)

Move	Layer 7	Layer 8	Layer 9	Layer 10	Layer 11	Layer 12	Layer 13	Final
♕×b1+	25.28%	27.69%	24.79%	33.23%	40.50%	41.78%	56.96%	35.99%
♘d2	0.68%	0.26%	7.69%	1.87%	0.16%	0.17%	0.21%	0.32%
b5	16.94%	26.56%	17.03%	25.45%	35.78%	33.58%	8.85%	36.12%
♘×b2	31.09%	17.79%	12.84%	12.49%	4.64%	0.73%	0.70%	0.35%
♘e3	0.02%	0.01%	0.04%	0.03%	0.04%	0.04%	0.20%	0.32%
♕×b2+	10.75%	11.81%	11.53%	8.22%	3.26%	0.36%	0.50%	0.29%
♖×b2	9.97%	10.30%	8.75%	6.71%	2.18%	0.32%	0.49%	0.44%
♘d6	1.16%	1.37%	5.72%	2.96%	5.18%	10.97%	9.70%	5.41%
♘a5	0.95%	0.83%	4.04%	1.76%	0.93%	2.08%	5.94%	1.85%
b6	1.28%	1.19%	1.84%	1.99%	2.01%	0.87%	2.58%	0.62%
♘a3	0.46%	0.31%	2.31%	0.77%	0.49%	0.84%	1.79%	0.94%
♖2a4	0.16%	0.24%	0.46%	0.35%	0.63%	1.68%	2.24%	2.28%
♖8a4	0.07%	0.19%	0.27%	0.29%	0.29%	0.92%	1.06%	2.18%
♔f8	0.03%	0.05%	0.11%	0.22%	0.25%	0.21%	0.16%	0.40%
♖8a3	0.11%	0.20%	0.47%	0.32%	0.19%	0.60%	0.74%	0.64%
♖8a5	0.08%	0.12%	0.11%	0.13%	0.26%	0.22%	0.45%	1.35%
♖2a6	0.02%	0.02%	0.05%	0.04%	0.08%	0.22%	0.65%	0.93%
f5	0.14%	0.17%	0.06%	0.25%	0.50%	0.21%	0.49%	0.54%
♔g7	0.04%	0.03%	0.05%	0.08%	0.02%	0.04%	0.20%	0.77%
♖2a3	0.10%	0.16%	0.42%	0.19%	0.21%	0.72%	0.64%	0.65%
♖2a7	0.02%	0.02%	0.05%	0.06%	0.12%	0.27%	0.47%	0.71%
c5	0.04%	0.07%	0.10%	0.25%	0.15%	0.34%	0.68%	0.70%
♖2a5	0.06%	0.08%	0.16%	0.11%	0.32%	0.51%	0.67%	0.52%
♔h8	0.06%	0.06%	0.24%	0.66%	0.35%	0.17%	0.29%	0.41%
♖8a6	0.03%	0.03%	0.05%	0.07%	0.06%	0.18%	0.27%	0.61%
e5	0.11%	0.10%	0.14%	0.09%	0.07%	0.08%	0.41%	0.54%
♖f8	0.02%	0.01%	0.03%	0.07%	0.04%	0.08%	0.18%	0.40%
♔h7	0.04%	0.04%	0.16%	0.35%	0.35%	0.12%	0.22%	0.51%
g5	0.10%	0.10%	0.10%	0.32%	0.40%	0.45%	0.41%	0.41%
♖b8	0.02%	0.01%	0.03%	0.04%	0.03%	0.08%	0.16%	0.47%
♖8a7	0.02%	0.02%	0.04%	0.07%	0.07%	0.28%	0.26%	0.47%
♘b6	0.07%	0.07%	0.14%	0.13%	0.18%	0.27%	0.43%	0.32%
♖d8	0.02%	0.01%	0.03%	0.07%	0.03%	0.09%	0.26%	0.42%
♖e8	0.02%	0.01%	0.03%	0.11%	0.04%	0.12%	0.23%	0.39%
♖c8	0.02%	0.01%	0.04%	0.13%	0.04%	0.15%	0.27%	0.39%
♘e5	0.04%	0.06%	0.07%	0.11%	0.15%	0.27%	0.23%	0.34%

L Look-ahead analysis details

This appendix provides full methodological details for the three look-ahead analyses replicated from Jenner et al. (2024) on forgotten puzzles. Error bars and confidence intervals follow their methodology throughout. Model variant details for look-ahead experiments are provided in Appendix D.

Dataset construction We start from the full Lichess puzzle database of approximately 900,000 puzzles and apply the same filtering criteria as Jenner et al. (2024): puzzles must have a principal variation of at least three moves, a weaker sparring model must assign less than 5% probability to the first move (ensuring difficulty), and more than 70% to the opponent's response (ensuring the line is forcing). We use the same sparring model as Jenner et al. (2024), available in their public inference framework. Mate-in-one puzzles are excluded. We then select forgotten puzzles: positions where the correct first move is the top candidate at some intermediate layer but not at the final layer. To ensure these represent genuine forgetting rather than absence of look-ahead, we additionally require the full model to assign $> 50\%$ probability to all subsequent player moves in the principal variation, confirming that the model demonstrably understands the winning continuation and thus plausibly employs look-ahead. This yields 8,648 forgotten puzzles. For comparison, Jenner et al. (2024) apply the same filtering but select positions the model solves correctly; repeating this selection with the original model and full game history (Appendix D) yields 23,601 positions. Neither dataset directly filters for the presence of look-ahead mechanisms; any effects observed in the analyses below are emergent rather than selected for.

Corruption generation Following Jenner et al. (2024), we generate corrupted board states by applying small modifications to the original position, such as adding or removing pawns or moving non-pawn pieces to empty squares. Illegal board states are discarded. Corruptions are selected to substantially affect the full model while minimally affecting the sparring model, thereby targeting mechanisms beyond simple heuristics. Although the model does not select the correct move in forgotten puzzles, it still assigns nonzero probability to it—consistent with the solution being computed but overridden—and log-odds reduction remains a meaningful metric for measuring causal importance. Specifically, a corruption is retained if (1) the log-odds of the correct move under the full model decrease by at least 1.0, (2) the corruption worsens the position (value change < -0.1, where position value is $p_{\text{win}} - p_{\text{loss}}$), (3) the sparring model's log-odds of the correct move decrease by less than 0.2, and (4) the Jensen-Shannon divergence between the sparring model's clean and corrupted policies is below 0.2. Among the corruptions that pass, we use the one with the lowest divergence. The log-odds criterion differs from Jenner et al. (2024), who require the correct move's probability to drop below 10%; since the correct move already has low probability in forgotten puzzles, we use a relative decrease instead. For solved puzzles we keep their probability criterion. Of the 8,648 forgotten puzzles, 7,460 (86.3%) yield valid corruptions; of the 23,601 solved puzzles, 20,670 (87.6%).

Corrupted positions do not arise from an actual game and therefore have no move history. We feed each corrupted position together with the history of its clean counterpart, i.e., only the current position is modified. This makes the input slightly out of distribution, since the last position in the history no longer leads to the current one by a legal move. To check that this does not affect our conclusions, we repeated corruption generation and patching in two alternative settings: (i) filling all history slots of both the clean and the corrupted board with the current position, which is how Leela handles positions given without history, and (ii) using the fine-tuned model that ignores history (Appendix D). Both yield the same qualitative results.

Activation patching For each position-corruption pair, we run a clean forward pass and a corrupted forward pass. Token representations from the corrupted pass are patched into the clean pass one square at a time at each layer, and the effect on the log-odds of the correct move is measured. We report the mean effect across positions for four categories of squares: the first move target, the third move target together with the target squares of all later moves in the principal variation, the corrupted square(s), and all remaining squares, taking the largest effect within each category that contains several squares. Following Jenner et al. (2024), we exclude puzzles in which these categories overlap, which happens when a corrupted square is the target of a principal-variation move or when the first move's target square is also targeted by a later move. This leaves 5,597 forgotten and 12,080 solved puzzles.

Attention head ablation We replicate the attention ablation from Jenner et al. (2024). For each forgotten puzzle, we zero the attention score where the first move's target square (query) attends to the third move's target square (key), preventing information flow between these squares. We measure the log-odds reduction of the correct move. As a baseline, we simultaneously zero all remaining 4,095 attention entries in the same head. Results for L12H12 are presented in the main text (Figure 5b). Figure 22 shows results for four additional heads identified by Cruz (2025). L12H17 and L13H3 show a similar pattern to L12H12, with the single look-ahead entry producing outsized effects compared to the baseline, while L11H13 and L11H10 show no visible effects on forgotten puzzles.

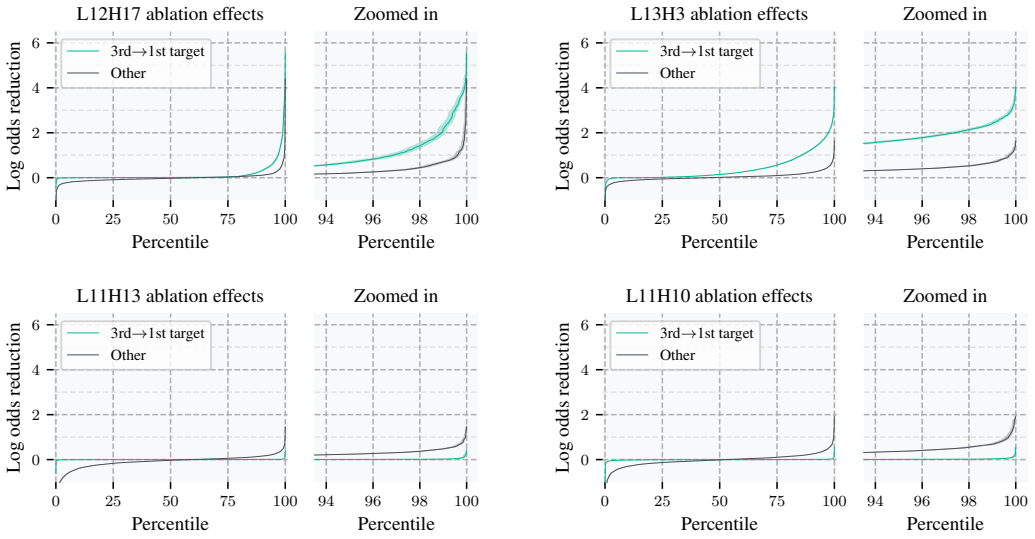


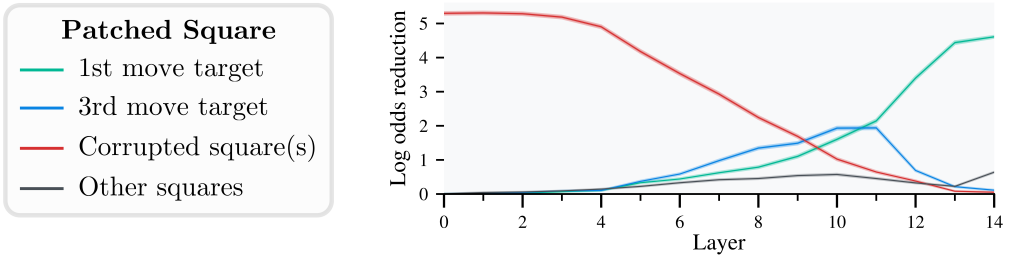
Figure 22: Attention head ablation results on forgotten puzzles for four additional look-ahead heads identified by Cruz (2025). Format follows Figure 5b: the **single attention entry** between first and third move target squares versus all **other** 4,095 entries, shown as percentile curves sorted by effect size.

Presence and necessity of look-ahead Neither dataset filters for look-ahead, so both the forgotten and the solved set may contain positions where it plays no role. We count look-ahead as present in a puzzle if patching the **3rd move target** square lowers the log-odds of the correct move at the full model by more than the 95th percentile of the same patch applied to a random square. This holds for 85.9% of the 5,597 forgotten puzzles used for patching. To get a lower bound on how often look-ahead is necessary for the intermediate solution, we cut the L12H12 entry described above and read out the policy at the penultimate layer. Among the 2,068 puzzles in which look-ahead is present and the correct move is on top at the penultimate layer, cutting this entry removes the correct move from the top in 29.9% of cases, against 9.8% when all other 4,095 entries are cut and 5.8% when only the most-attended other entry is cut, and lowers its log-odds by more than 1 in 14.7% of cases, against 1.6% and 1.4%. The controls flip the top move this often because the correct move is often only narrowly on top at this layer, with a median probability of 0.39. The corresponding entry in L13H3 partly backs up L12H12: cutting both removes the correct move from the top in 44.4% of cases and lowers its log-odds by more than 1 in 26.6% of cases, against 7.5% and 1.7% when each head's most-attended other entry is cut. Without the presence condition, among all 3,781 forgotten puzzles with the correct move on top at the penultimate layer, cutting both entries removes it from the top in 32.2% of cases, against 8.4% for the control, and lowers its log-odds by more than 1 in 18.4% of cases, against 1.9%. Since look-ahead information can also reach the **1st move target** square through other paths, these rates are lower bounds on how often look-ahead is necessary for the intermediate solution.

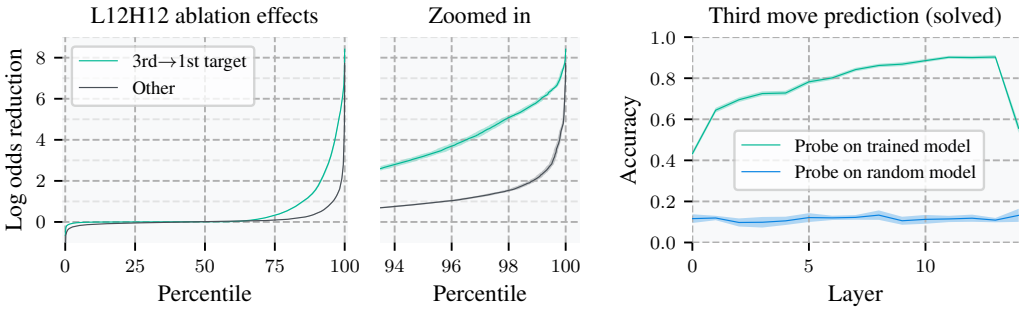
Probing We train bilinear probes following the two-step architecture of Jenner et al. (2024): first predicting the target square of the third move from activations at the first move's target square, then predicting the source square conditioned on the predicted target. On forgotten puzzles, we condition on the ground-truth first move target square rather than the model's predicted move, since

by definition the model does not select the correct first move. Probes are trained on a $70/30$ train/test split with five independent training runs. As a baseline, following Jenner et al. (2024), we train probes with the same architecture on a randomly initialized model.

Comparison with solved puzzles For reference, Figure 23 shows the same analyses, following Jenner et al. (2024), on our solved puzzle set, enabling direct visual comparison with the forgotten puzzle results in Figure 5.



(a) Activation patching



(b) Attention head ablation

(c) Probing

Figure 23: Look-ahead analyses following Jenner et al. (2024) on our solved puzzle set, for direct comparison with the forgotten puzzle results in Figure 5.

M Value head evaluation

The policy and value heads share the transformer encoder and are trained on the same self-play games: the policy head to predict the search outcome and the value head to predict the game outcome (Appendix D.4). The value head therefore approximates the value of a position under that kind of play rather than an objective assessment of it, so the two heads should behave consistently. Yet Section 3.2 reports that the value head prefers the position after the solution move in 97.0% of forgotten puzzles, which raises two questions. First, does the value head see the win at the puzzle position itself, where the policy misses it? Second, if it does not, why does it still assign some probability to a win on forgotten puzzles, and to a loss on solved ones? We answer the first by evaluating the value head at the puzzle position, and the second with the Bellman equation, which relates the value of a position to the moves the policy plays from it.

At the puzzle position The comparison in the introduction of Section 3.2 is between the positions that result from playing a move, so there is no same-input contradiction between the two heads. At the puzzle position itself, the value head does not see a win in the forgotten positions: a draw is the most likely single outcome, and loss is more likely than win (Table 35). The policy and the value head agree here. The value head evaluates forgotten positions markedly lower than solved ones, with a mean win probability of 21.6% against 63.0%. What it evaluates accurately is the position after the move has been played, where the winning line is usually clear: after the solution move, its mean win probability on forgotten puzzles is 75.6%. Search reaches the solution only because this position is evaluated correctly; if it were not, the search would abandon that branch (Appendix P).

Table 35: Mean value-head prediction at the puzzle position (side to move) and the win probability after the solution move, from the mover's side.

		Puzzle position			After solution move
Set	n	**Win**	**Draw**	**Loss**	**Win**
Forgotten puzzles	8,648	21.6%	42.1%	36.3%	75.6%
Solved puzzles	23,601	63.0%	21.6%	15.4%	88.6%

Bellman consistency For any policy π, the Bellman equation (Sutton and Barto, 2018) relates its value function at a position to the values of the positions its own moves lead to, $V^\pi(s) = \sum_m \pi(m \mid s) \cdot V^\pi(s \to m)$, with win and loss swapped to the perspective of the player at s. The value of a position is the mean of the values of the resulting positions, weighted by the probability the policy assigns to the moves leading to them. We write $\hat{V}(s)$ for the right-hand side, the model's own one-ply expansion, and compare it with the value head's direct reading at the puzzle position, $V(s)$. This is a theoretical property of the value function rather than something the loss explicitly imposes, so measuring it checks how well the head has learned the value of its own policy. It holds in aggregate (Table 36). The two components in question match most closely: the win probability on forgotten puzzles, 21.6% against 23.0%, and the loss probability on solved puzzles, 15.4% against 16.0%. Both therefore come from the moves the policy plays. Residual mass on the correct move raises the value of a forgotten puzzle without making it solved, and mass on losing moves lowers the value of a solved one without making it unsolved. The value head reflects the whole distribution; solving depends on the argmax alone. One systematic deviation remains: the direct evaluation is slightly less resolved than the same evaluation one move later, and puts more mass on a draw than its own expansion, by 6.2 percentage points on forgotten and 8.0 on solved puzzles.

Table 36: The value head's direct prediction at the puzzle position, $V(s)$, and its one-ply expansion under the model's own policy, $\hat{V}(s)$, averaged over puzzles. Bold marks the components discussed in the text.

Set		**Win**	**Draw**	**Loss**
Forgotten puzzles ($n = 8{,}648$)	$V(s)$	**21.6%**	42.1%	36.3%
	$\hat{V}(s)$	**23.0%**	36.0%	41.1%
Solved puzzles ($n = 23{,}601$)	$V(s)$	63.0%	21.6%	**15.4%**
	$\hat{V}(s)$	70.4%	13.6%	**16.0%**

Where the win mass comes from Table 37 decomposes the win component of the expansion on forgotten puzzles by what Stockfish says each move leads to. More than half of the win mass, 56%, is the term for the correct move, and most of the rest comes from drawing and near-drawing alternatives to which the value head still assigns some win probability.

Table 37: Win component of the one-ply expansion on forgotten puzzles, split by the Stockfish 17.1 evaluation of each legal move at depth 12, in pawns from the mover's side. Policy mass and contribution are means over puzzles; the win probability after the move is the policy-weighted mean within each band, so that policy mass times win probability gives the contribution.

Moves	**Policy mass**	**Win after move**	**Contribution**	**Share**
The solution move	16.9%	75.7%	12.8%	56%
Drawing (-0.5 to $+0.5$)	28.3%	15.3%	4.3%	19%
Slight edge for the player ($+0.5$ to $+2$)	8.4%	36.4%	3.1%	13%
Slight edge for the opponent (-2 to -0.5)	18.7%	8.4%	1.6%	7%
Losing (below -2)	27.0%	3.3%	0.9%	4%
Another winning move (above $+2$ or mate)	0.7%	47.0%	0.3%	1%
Total	100.0%		23.0%	100%

N Stockfish concept preference analysis

We analyze how different network layers prioritize chess concepts by measuring the expected change in concept values induced by each layer's move distribution. Unlike McGrath et al. (2022), who probed for concept *representations* using linear classifiers on intermediate activations, we directly measure concept *preferences* from policy outputs. This approach reveals what concepts each layer prioritizes when selecting moves, complementing representation-based analyses.

Chess concepts We use Stockfish 8's (Stockfish developers, 2016) handcrafted evaluation function as our source of chess concepts, following McGrath et al. (2022) to enable direct comparison with prior work from Sadler and Regan (2019). Stockfish decomposes position evaluation into interpretable components including material balance, piece-specific features, king safety, threats, mobility, passed pawns, and spatial control. Each concept is evaluated separately for the midgame (mg) and the endgame (eg). A phase-interpolated value (ph) combines the two as a weighted sum based on the game phase. All concepts are represented as continuous values. Table 38 summarizes the main concept categories we analyze.

Table 38: Summary of chess concepts from Stockfish 8's evaluation function taken from McGrath et al. (2022). Concepts are enumerated as `<concept_name> <side> <game_phase>`, where side is `[mine|opponent|t]` for current player, opponent, or total (difference), and game phase is `[mg|eg|ph]` for midgame, endgame, or phase-interpolated value.

Concept	Description
`material t [mg|eg|ph]`	Material score, where each piece on the board has a predefined value that changes depending on the phase of the game.
`imbalance t [mg|eg|ph]`	Imbalance score that compares the piece count of each piece type for both colors. E.g., it awards having a pair of bishops vs a bishop and a knight.
`pawns t [mg|eg|ph]`	Evaluation of the pawn structure. E.g., the evaluation considers isolated, doubled, connected, backward, blocked, weak, etc. pawns.
`knights [mine|opponent|t] [mg|eg|ph]`	Evaluation of knights. E.g., extra points are given to knights that occupy outposts protected by pawns.
`bishops [mine|opponent|t] [mg|eg|ph]`	Evaluation of bishops. E.g., bishops that occupy the same color squares as pawns are penalized.
`rooks [mine|opponent|t] [mg|eg|ph]`	Evaluation of rooks. E.g., rooks that occupy open or semi-open files have higher valuation.
`queens [mine|opponent|t] [mg|eg|ph]`	Evaluation of queens. E.g., queens that have relative pin or discovered attack against them are penalized.
`mobility [mine|opponent|t] [mg|eg|ph]`	Evaluation of piece mobility score. It depends on the number of squares attacked by the pieces.
`king safety [mine|opponent|t] [mg|eg|ph]`	A complex concept related to king safety. It depends on the number and type of pieces that attack squares around the king, shelter strength, number of pawns around the king, penalties for being on a pawnless flank, etc.
`threats [mine|opponent|t] [mg|eg|ph]`	Evaluation of threats to pieces, such as whether a pawn can safely advance and attack an opponent's higher value piece, hanging pieces, possible x-ray attacks by rooks, etc.
`passed pawns [mine|opponent|t] [mg|eg|ph]`	Evaluates bonuses for passed pawns. The closer a pawn is to the promotion rank, the higher the bonus.
`space [mine|opponent|t] [mg|eg|ph]`	Evaluation of the space. It depends on the number of safe squares available for minor pieces on the central four files on ranks 2 to 4.
`total t [mg|eg|ph]`	The total evaluation of a given position. It encapsulates all the above concepts.

Stockfish version Stockfish evaluated positions with a handcrafted function from its first release in 2008 until version 12 in 2020, which made Efficiently Updatable Neural Networks, NNUE (Nasu, 2018), the main evaluation (Stockfish developers, 2020). NNUE is a small neural network that runs on a CPU fast enough to evaluate every position in Stockfish's alpha-beta search, whereas Leela's much larger network runs on a GPU and evaluates far fewer positions in its Monte Carlo tree search. The handcrafted evaluation remained in the engine for some time after that until it was removed entirely in 2023, so no newer interpretable evaluation exists. Stockfish 8, released in 2016, was AlphaZero's opponent in its evaluation matches (Silver et al., 2018), and Sadler and Regan (2019) study AlphaZero's play using games from these matches in a book widely read in the chess community. McGrath et al. (2022) use Stockfish 8's terms for this reason, and following them keeps our concept measurements comparable with theirs and legible to the chess community. We use the terms as interpretable functions of the position rather than as an evaluation oracle, so the engine's playing strength does not matter here.

Perspective normalization Stockfish evaluates all positions from White's perspective, with positive values favoring White. To ensure consistency with Leela's current-player perspective, we convert Stockfish's evaluations to player-relative coordinates following McGrath et al. (2022). For concepts with side-specific values (knights, bishops, rooks, queens, mobility, king safety, threats, passed pawns, space), we relabel White/Black as Mine/Opponent based on the side to move. **When White is to move**, White concepts become Mine and Black concepts become Opponent, with values unchanged. **When Black is to move**, Black concepts become Mine and White concepts become Opponent, with all values negated to reflect the perspective change. For aggregate concepts (material, imbalance, pawns, total), we negate values when Black is to move. This ensures that positive concept values always represent advantages for the player to move.

Concept delta calculation For each position s with player p to move, we compute concept preferences as follows:

1. **Evaluate initial position**: Use Stockfish 8 to obtain concept values $c(s)$ for all concepts c, normalized to player p's perspective.
2. **Generate and evaluate legal moves**: For each legal move m leading to position s', evaluate $c(s')$ using Stockfish and normalize to player p's perspective (accounting for the perspective flip after the move).
3. **Calculate concept deltas**: For each move-concept pair, compute:
$$\Delta c_m = c(s') - c(s) \tag{16}$$
where both $c(s')$ and $c(s)$ are evaluated from player p's perspective, ensuring Δc_m consistently represents concept change from the moving player's viewpoint.
4. **Compute layer-wise preferences**: For each layer ℓ, obtain the move probability distribution π_ℓ using the logit lens and calculate the expected concept delta:
$$\Delta c_\ell = \sum_{m \in \text{legal}(s)} \pi_\ell(m) \cdot \Delta c_m = \mathbb{E}_{\pi_\ell}[\Delta c_m] \tag{17}$$

This weighted average represents the expected change in concept c when sampling moves according to layer ℓ's policy.

Dataset and sampling We sample $10{,}000$ positions from the CCRL dataset (Leela Chess Zero team, 2018), a standard benchmark consisting of 2.5 million computer games from CCRL $40/40$ and $40/4$ tournaments. The dataset contains games between strong chess engines and provides diverse positions across different game phases and strategic themes.

Positions are sampled using a two-stage hierarchical process until $10{,}000$ unique positions are collected:

1. Sample 10% of games uniformly from the dataset
2. From each selected game, sample 10% of positions uniformly
3. Filter duplicate positions to ensure uniqueness

Implementation details For each sampled position, we:

- Extract move policies π_ℓ for the input embedding layer plus all 15 transformer layers, where the last layer corresponds to the final model output
- Evaluate the initial position and all legal move outcomes (typically 30–40 moves per position) using Stockfish 8
- Calculate concept deltas and probability-weighted averages for all 93 Stockfish concepts across midgame, endgame, and phase-interpolated variants
- Store results for analysis and visualization

Figures in the main paper present phase-interpolated (ph) concept values, as these represent Stockfish's actual evaluation for a given position. Comprehensive results showing all concept variants (mg, eg, ph) across all layers are provided in this appendix (Figures 24–30).

Statistical analysis We report mean Δc_ℓ values across all 10,000 positions with 95% confidence intervals computed via the t-distribution. Many concepts have values of zero—and therefore zero deltas—in positions lacking relevant pieces (e.g., bishop concepts when no bishops are present, or passed-pawn bonuses without passed pawns) or where the concept is undefined. The endgame variants of the queen and space terms are zero in every position, so their concept preference is zero at every layer and they yield no steering direction. This produces zero-inflated distributions that bias means toward zero. We do not control for this effect, which makes absolute Δc_ℓ values across concepts skewed and not directly comparable. Nevertheless, relative trends across layers remain informative for understanding how concept preferences evolve with depth.

Interpretation of concept evolution Analysis of concept preferences across layers reveals three distinct patterns corresponding to the computational phases identified in playing strength progression. In the early phase (layers `Input` to 5), most concepts exhibit erratic, volatile shifts with substantial fluctuations in preference values. During the middle phase (layers 6 to 10), concept preferences stabilize significantly, with most concepts maintaining relatively constant values across this range, mirroring the performance plateau observed in tournament play and puzzle solving. In the late phase (layers 11 to `Final`), concept preferences show consistent, smooth trends—typically monotonic increases or decreases for each concept, with minimal erratic behavior. This systematic evolution coincides with the sharp capability improvements and emergence of look-ahead mechanisms in late layers. The shift from aggressive to safety-oriented concepts occurs primarily in this phase: the model increasingly prefers moves that raise **own king safety** and reduce **opponent threats** (prioritizing defensive considerations), while its preference for moves that raise **opponent king vulnerability** and **own threats** decreases (de-emphasizing aggressive play). These patterns hold across most concepts, though some show different dynamics requiring deeper chess expertise to interpret. The consistency of trends in middle and late phases is particularly striking given the volatility of early layers, suggesting fundamentally different computational regimes across network depth.

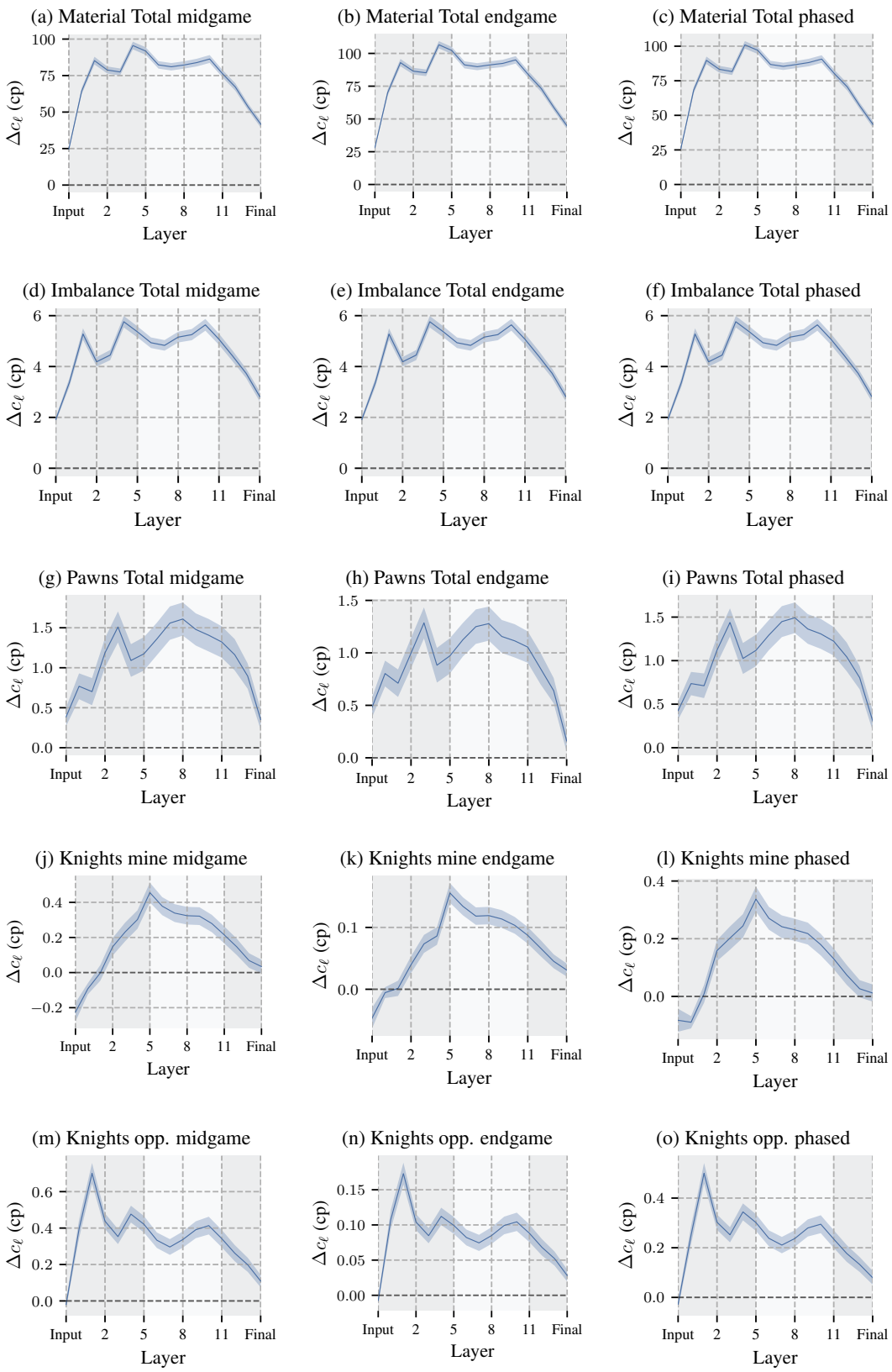


Figure 24: Stockfish evaluation concepts across layers (Part 1 of 7). Lines show mean probability-weighted concept delta across positions; shaded regions show 95% confidence intervals. All concepts are evaluated from the current player's perspective.

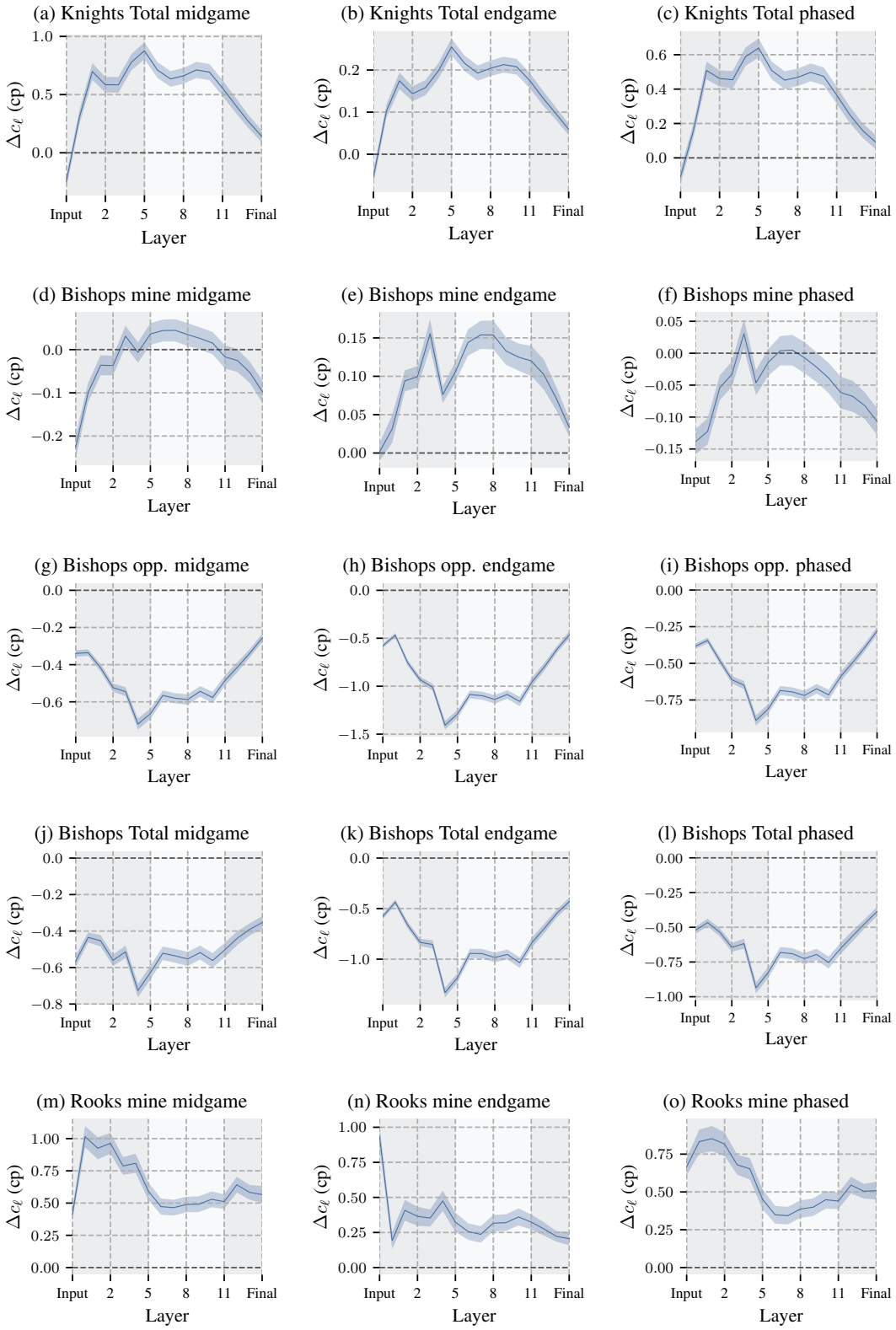


Figure 25: Stockfish evaluation concepts across layers (Part 2 of 7). Lines show mean probability-weighted concept delta across positions; shaded regions show 95% confidence intervals. All concepts are evaluated from the current player's perspective.

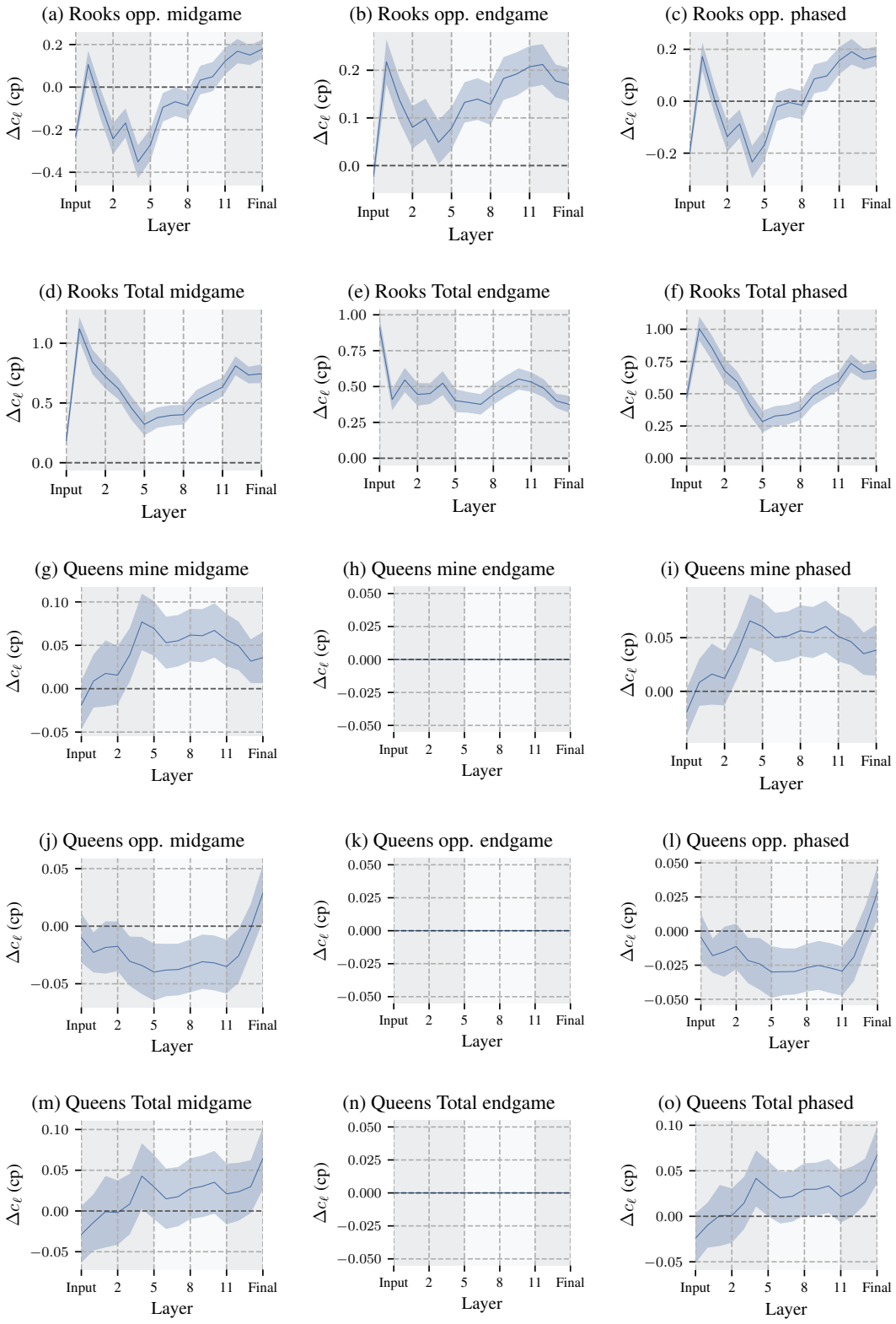


Figure 26: Stockfish evaluation concepts across layers (Part 3 of 7). Lines show mean probability-weighted concept delta across positions; shaded regions show 95% confidence intervals. All concepts are evaluated from the current player's perspective.

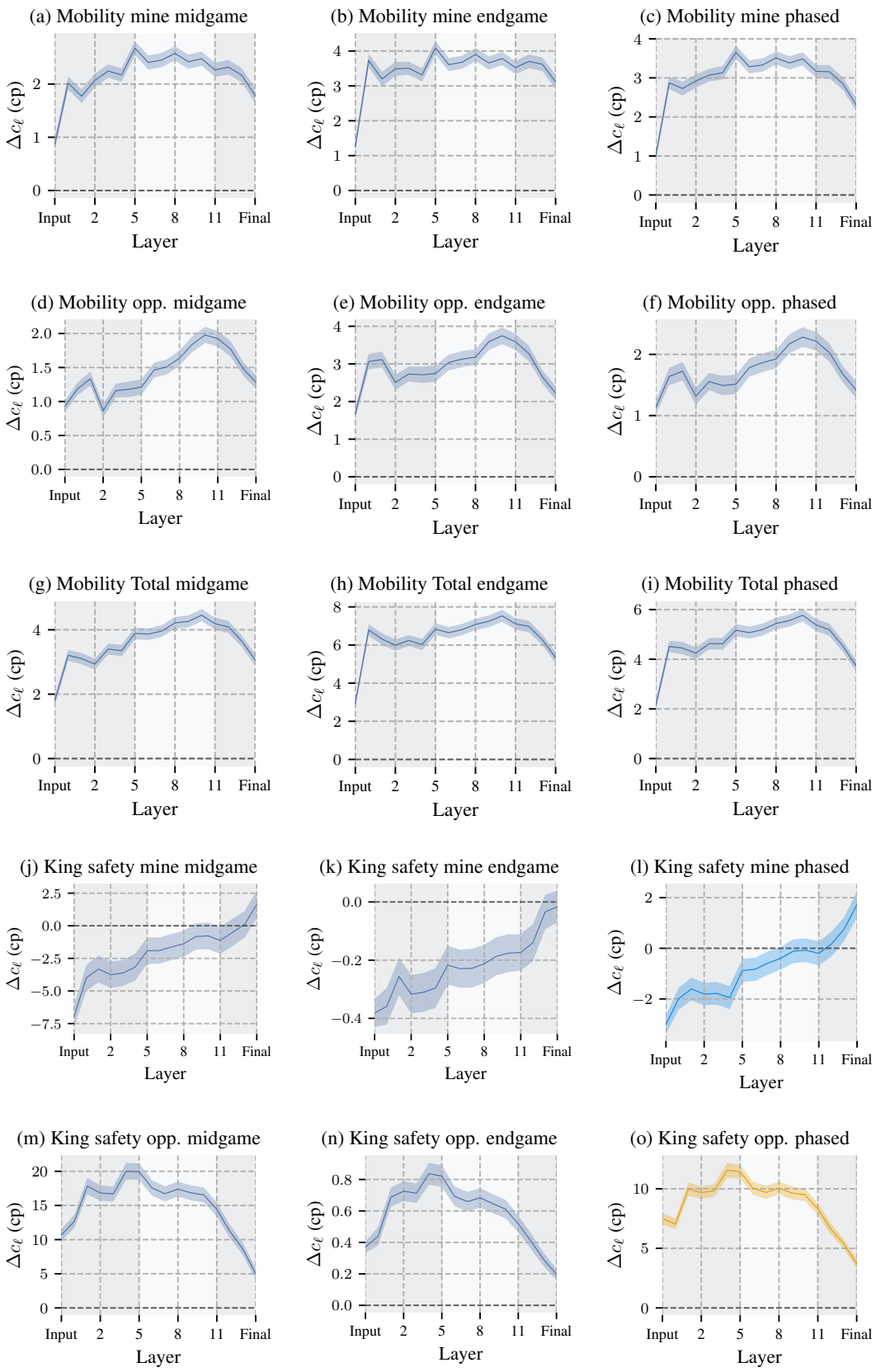


Figure 27: Stockfish evaluation concepts across layers (Part 4 of 7). Lines show mean probability-weighted concept delta across positions; shaded regions show 95% confidence intervals. All concepts are evaluated from the current player's perspective.

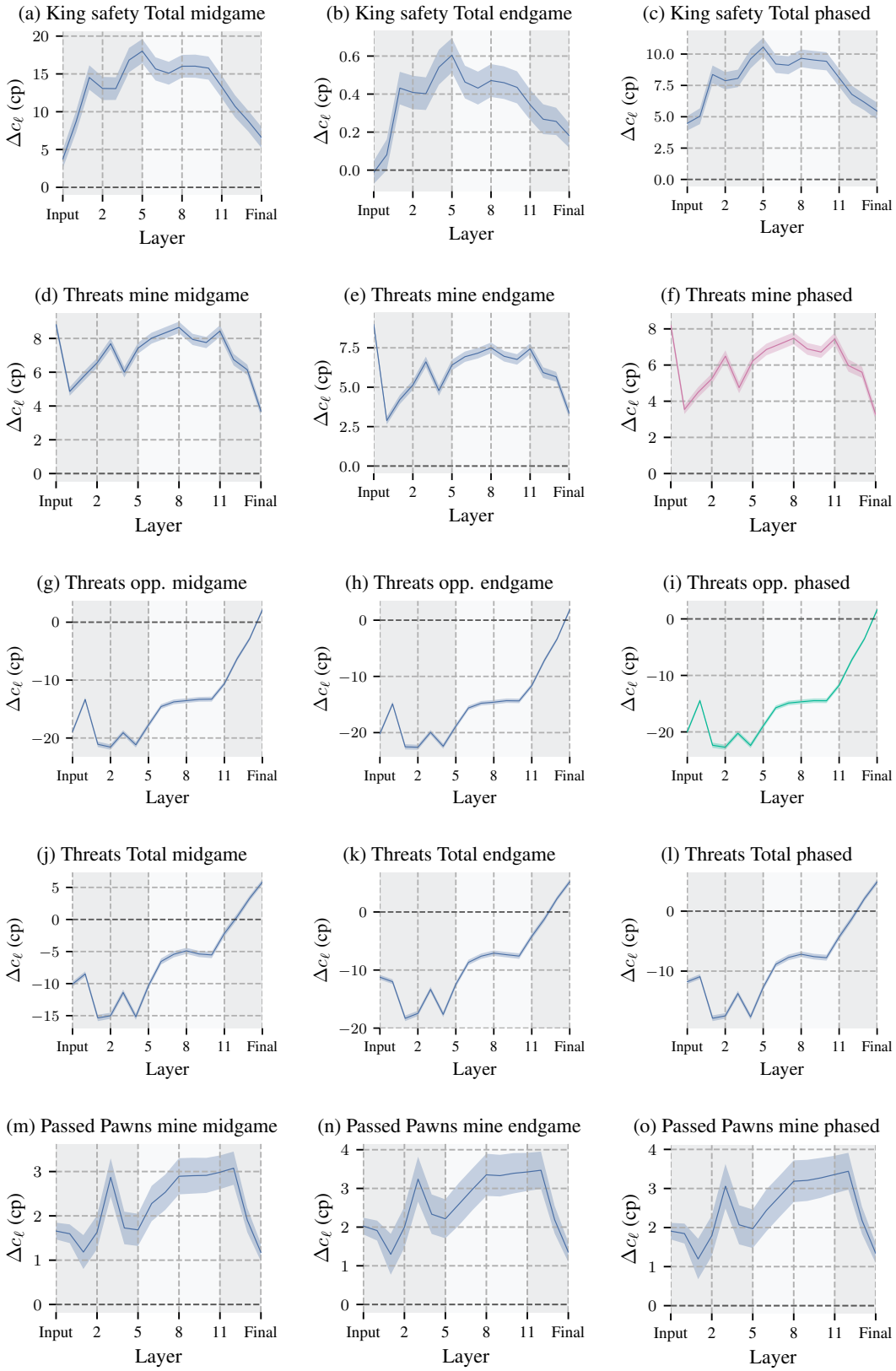


Figure 28: Stockfish evaluation concepts across layers (Part 5 of 7). Lines show mean probability-weighted concept delta across positions; shaded regions show 95% confidence intervals. All concepts are evaluated from the current player's perspective.

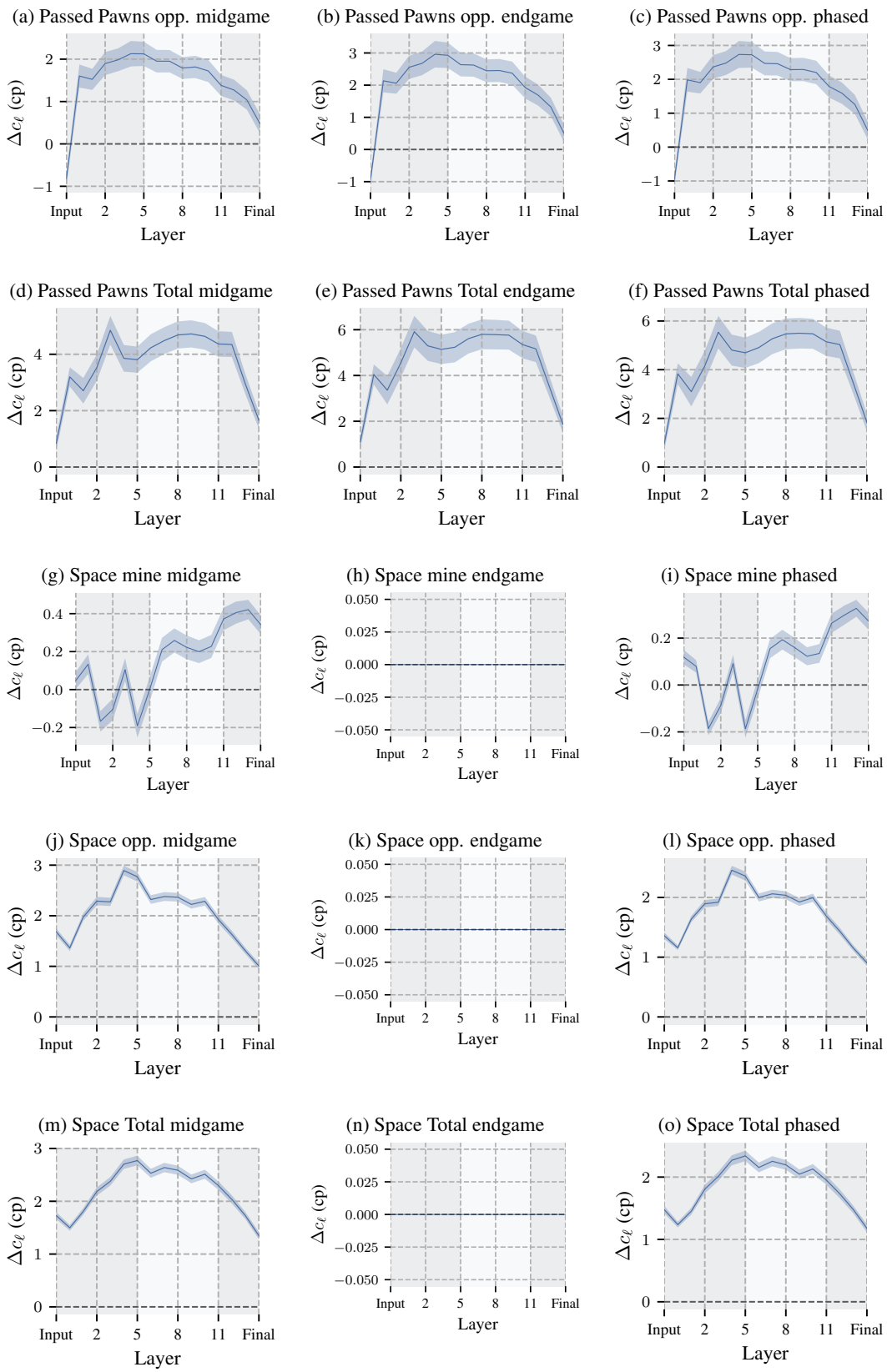


Figure 29: Stockfish evaluation concepts across layers (Part 6 of 7). Lines show mean probability-weighted concept delta across positions; shaded regions show 95% confidence intervals. All concepts are evaluated from the current player's perspective.

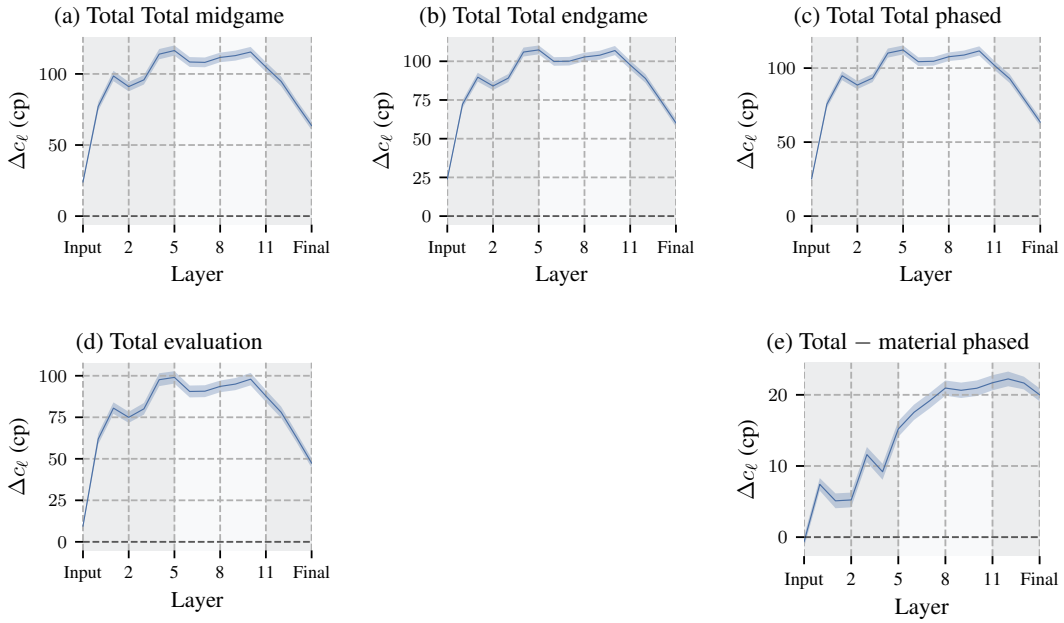


Figure 30: Stockfish evaluation concepts across layers (Part 7 of 7). Panel (d) shows Stockfish's final static evaluation, which differs from the phased total in (c) by scaling down drawish endgames and adding a bonus for the side to move. Panel (e) shows the phased total in (c) without the material term, which dominates the total. Lines show mean probability-weighted concept delta across positions; shaded regions show 95% confidence intervals. All concepts are evaluated from the current player's perspective.

O Concept steering details

This appendix provides complete details for the gradient-based concept steering experiments described in the main text. We group threats and king safety as safety-related concepts based on their intuitive connection to conservative play; this categorization is ours rather than intrinsic to Stockfish's evaluation terms. The consistency of the effect across four independently defined safety-related concepts suggests the result reflects a genuine property of the model rather than an artifact of any single proxy. All steering experiments use the original model with position history.

Gradient computation For each concept c and layer ℓ, we compute the gradient of the concept preference of the complete model Δc_L (Equation 1) with respect to the residual stream activations after layer ℓ:

$$\hat{\mathbf{v}}_{c,\ell} = \frac{\mathbb{E}[\nabla_{\mathbf{h}_\ell} \Delta c_L]}{\|\mathbb{E}[\nabla_{\mathbf{h}_\ell} \Delta c_L]\|} \tag{18}$$

Gradients are computed over 10,000 positions sampled from the CCRL (Leela Chess Zero team, 2018) dataset using the same hierarchical sampling procedure described in Appendix N. Gradients are averaged across all 64 squares and positions to obtain a single 768-dimensional steering vector $\hat{\mathbf{v}}_{c,\ell}$ per concept per layer based on 640,000 activations.

Steering strength The steering strengths $\alpha \in \{\pm 0.25, \pm 0.5, \ldots, \pm 1.5\}$ reported in the main text are scaled by the per-layer mean activation norm $\mathbb{E}[\|\mathbf{h}_\ell\|_2]$, computed over the CCRL gradient positions, to ensure comparable intervention magnitude across layers, which have different activation scales. The applied perturbation is:

$$\mathbf{h}'_\ell = \mathbf{h}_\ell + \alpha \cdot \mathbb{E}[\|\mathbf{h}_\ell\|_2] \cdot \hat{\mathbf{v}}_{c,\ell} \tag{19}$$

where $\hat{\mathbf{v}}_{c,\ell}$ is the normalized gradient direction. Positive α steers toward increasing the concept preference; negative α steers toward decreasing it. The steering vector is applied uniformly to all 64 square positions. We sweep over all strengths and all layers, evaluating each configuration on the full 10,000 general puzzle dataset using full principal variation correctness.

Held-out evaluation Selecting a setting and reporting its solve rate on the same puzzles overstates the effect of steering, so we use Monte Carlo cross-validation. We split the 10,000 puzzles randomly into two halves, select the setting on one half and measure the solve rate on the other, and repeat this for 200 splits with a fixed seed. For the per-layer curves in Figure 6 and Figures 31–37, the strength is selected separately for each concept and layer. Lines show the mean over splits and shaded areas one standard deviation, which describes the variation across splits rather than a confidence interval.

The highest curve in Figure 6, **opponent threats** at layer 13, reaches a held-out solve rate of $92.43 \pm 0.25\%$, the value reported in the main text. Picking the highest curve is itself a selection over concepts and layers, so we also select concept, layer and strength jointly by cross-validation, among all concept terms. This gives nearly the same held-out solve rate of $92.31 \pm 0.29\%$, with both values given as mean and standard deviation over splits. In every split, the selection lands on a safety-related concept, and on **opponent threats** at layer 13 in 180 of the 200 splits.

Fixed setting and transfer The counts of gained and lost puzzles in the main text and the transfer test use one fixed setting, **opponent threats** at layer 13 with $\alpha = -0.5$, the best strength on all 10,000 puzzles. It reaches 92.46% there, only 0.03 points above its held-out value. Because this setting is selected on the general dataset only, the forgotten dataset, which played no part in the selection, serves as a held-out test set. After removing the 53 puzzles it shares with the general dataset, steering recovers 71.7% of the remaining 8,595 puzzles.

Random baseline To establish that the steering effects are concept-specific rather than artifacts of generic activation perturbation, we compare against 50 random steering vectors drawn from an isotropic Gaussian distribution, each normalized to unit length. Each random vector is evaluated like a concept, with the strength selected per layer by the same cross-validation. The **random baseline** shows the mean held-out solve rate across the 50 vectors, with 95% confidence intervals across vectors.

Per-concept steering results Figures 31–37 show the held-out steering results for all Stockfish concepts. Each plot displays the held-out solve rate across layers with one standard deviation over splits, with point color indicating the most frequently selected strength. The **unsteered baseline** and **random baseline** are shown for reference. Three of the four safety-related concepts—**opponent threats**, **own threats** and **opponent king vulnerability**—consistently produce the largest improvements, reaching 91.8 to 92.4% at their best layers, while **own king safety** improves the solve rate only in the last layers, reaching 90.9% at layer 13. While other concepts also improve performance above the random baseline, likely reflecting correlations between piece-specific concepts and tactical move types common in puzzles, the effects are smaller and less consistent across layers. Steering in middle layers is typically least effective, again mirroring the computational phases identified in the main text. The most directly safety-related concepts yield the largest effects, supporting the interpretation that learned safety priors are a primary cause of the forgotten puzzle phenomenon.

Concepts versus move types Safe play, as we use the term, includes preferences over move types such as avoiding sacrifices, preserving material and preferring quiet moves, so safety-related concepts are expected to correlate with move types. A stronger concern is that a concept merely marks specific move types, such as captures or checks, so that steering it amounts to a preference for these moves regardless of their underlying risk. In 98.3% of forgotten puzzles, the first move of the solution is a capture, check or sacrifice, while the model replaces it with a quiet move in 68.3%. For each concept and each of these four move types, we measure as an AUC how well the concept's change Δc separates moves of that type from the other candidates in the same position, namely the solution's first move and the model's eight most probable moves, and take the largest deviation from 0.5 as the concept's encoding strength. As a positive control, material separates captures almost perfectly. We use each of the 31 concepts in its phase-interpolated form, so that each concept counts once. Consistent with this expectation, stronger move-type encoding goes with moderately stronger steering, with Spearman correlations of 0.45 for forgotten recovery at layer 13 and 0.37 for the peak held-out solve rate. Move-type encoding alone does not explain the effect, however. To test whether the safety-related concepts steer better than their move-type encoding predicts, we fit a linear relation between encoding strength and steering efficacy across the 27 other concepts. **Opponent threats**, **opponent king vulnerability** and **own threats** steer better than this fit predicts by 1.7 to 3.2 times the typical deviation of the other concepts from it. **Opponent threats** is the largest outlier of all 31 concepts, while **own king safety**, which is effective only in the last layers, lies only slightly above the fit.

Early-layer steering Steering toward aggression in early layers is nearly as effective as the best late setting. On all puzzles, steering **opponent king vulnerability** at layer 2 reaches 92.1%, compared with 92.5% for **opponent threats** at layer 13, but it also loses more previously solved puzzles, 186 against 129. To see how an intervention this early recovers forgotten puzzles, we read out the probability of the correct move with the logit lens at every layer after the steering point. For the 240 forgotten puzzles that steering at layer 2 recovers, the trajectory stays close to the unsteered one through the intermediate layers, and about three quarters of the gain in log-odds of the correct move arises in the final layer, which lowers the correct move in only 32% of these puzzles with steering. Early steering therefore acts mainly by weakening the final-layer override rather than by producing a stronger solution in earlier layers.

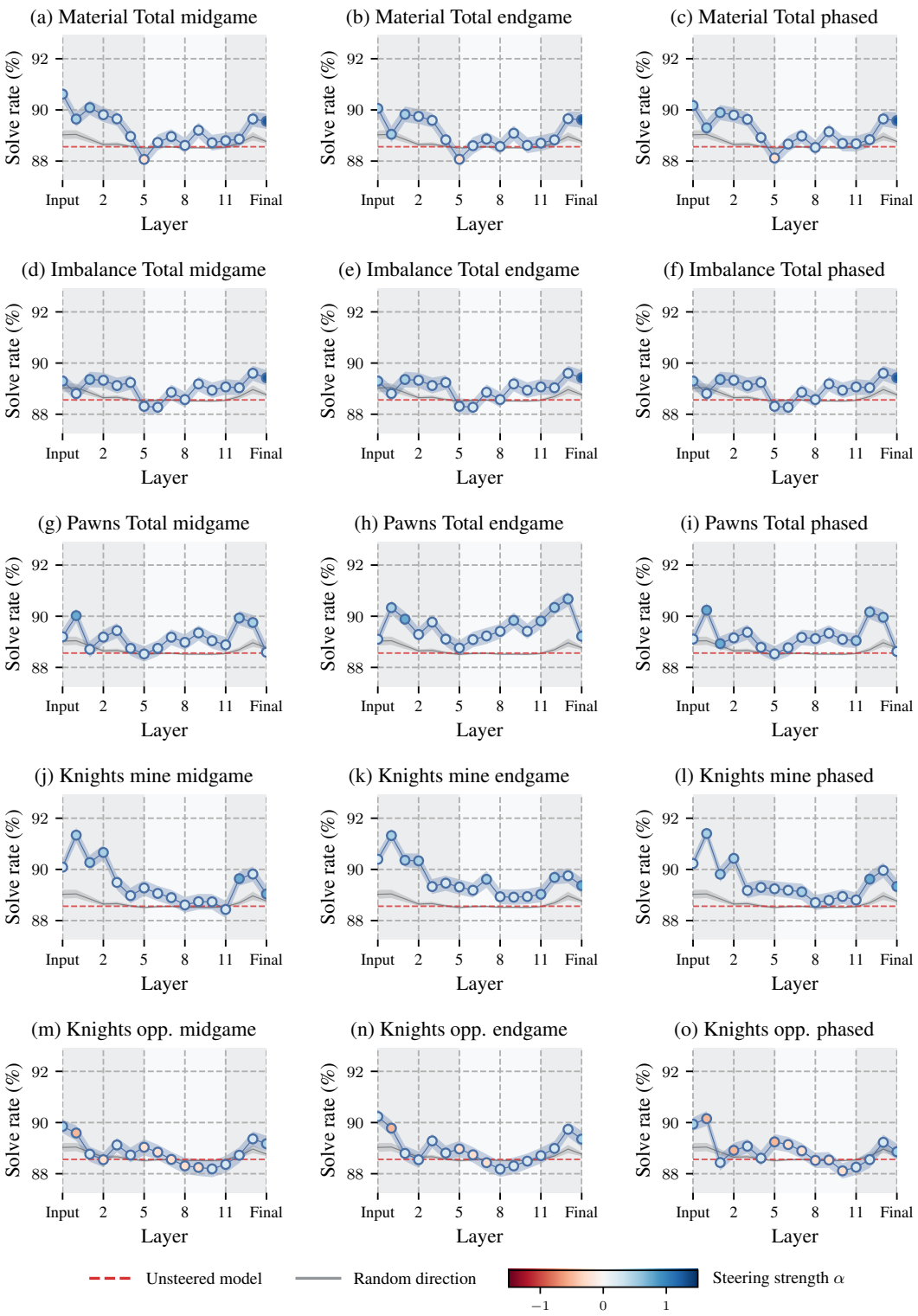


Figure 31: Concept steering effects across layers (Part 1 of 7). Each panel shows the held-out solve rate for the given concept at each intervention layer: the strength is selected on a random half of the puzzles and evaluated on the other half, averaged over 200 random splits, and the shaded band shows one standard deviation across splits. Marker colors indicate the most frequently selected strength, **blue** for positive and **red** for negative values. The **gray line and band** show the mean and 95% CI across 50 random directions under the same protocol. Panels without a curve are terms that are zero in every position, so no steering vector exists. The **red dashed line** shows the unsteered baseline.

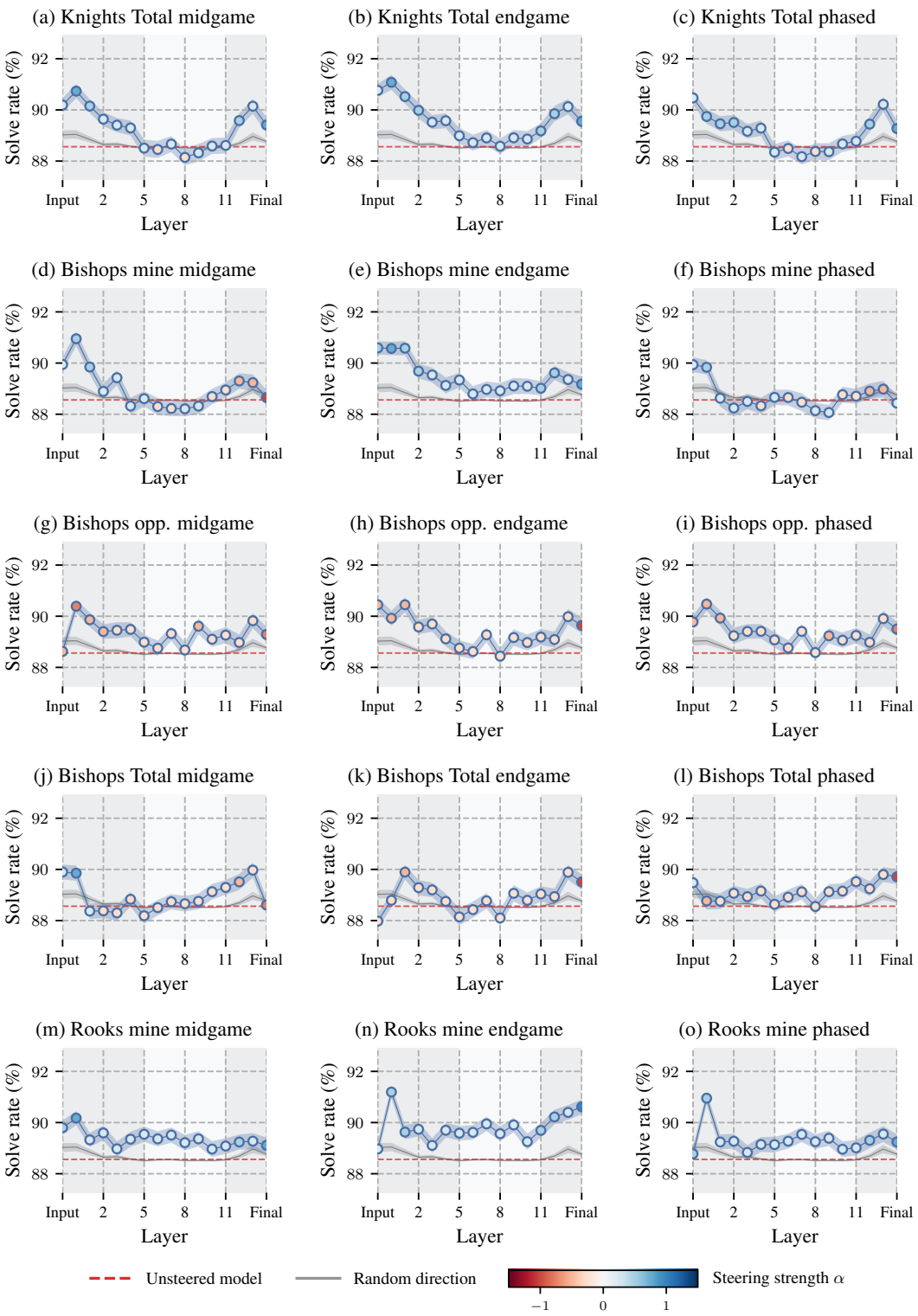


Figure 32: Concept steering effects across layers (Part 2 of 7). Each panel shows the held-out solve rate for the given concept at each intervention layer: the strength is selected on a random half of the puzzles and evaluated on the other half, averaged over 200 random splits, and the shaded band shows one standard deviation across splits. Marker colors indicate the most frequently selected strength, **blue** for positive and **red** for negative values. The **gray line and band** show the mean and 95% CI across 50 random directions under the same protocol. Panels without a curve are terms that are zero in every position, so no steering vector exists. The **red dashed line** shows the unsteered baseline.

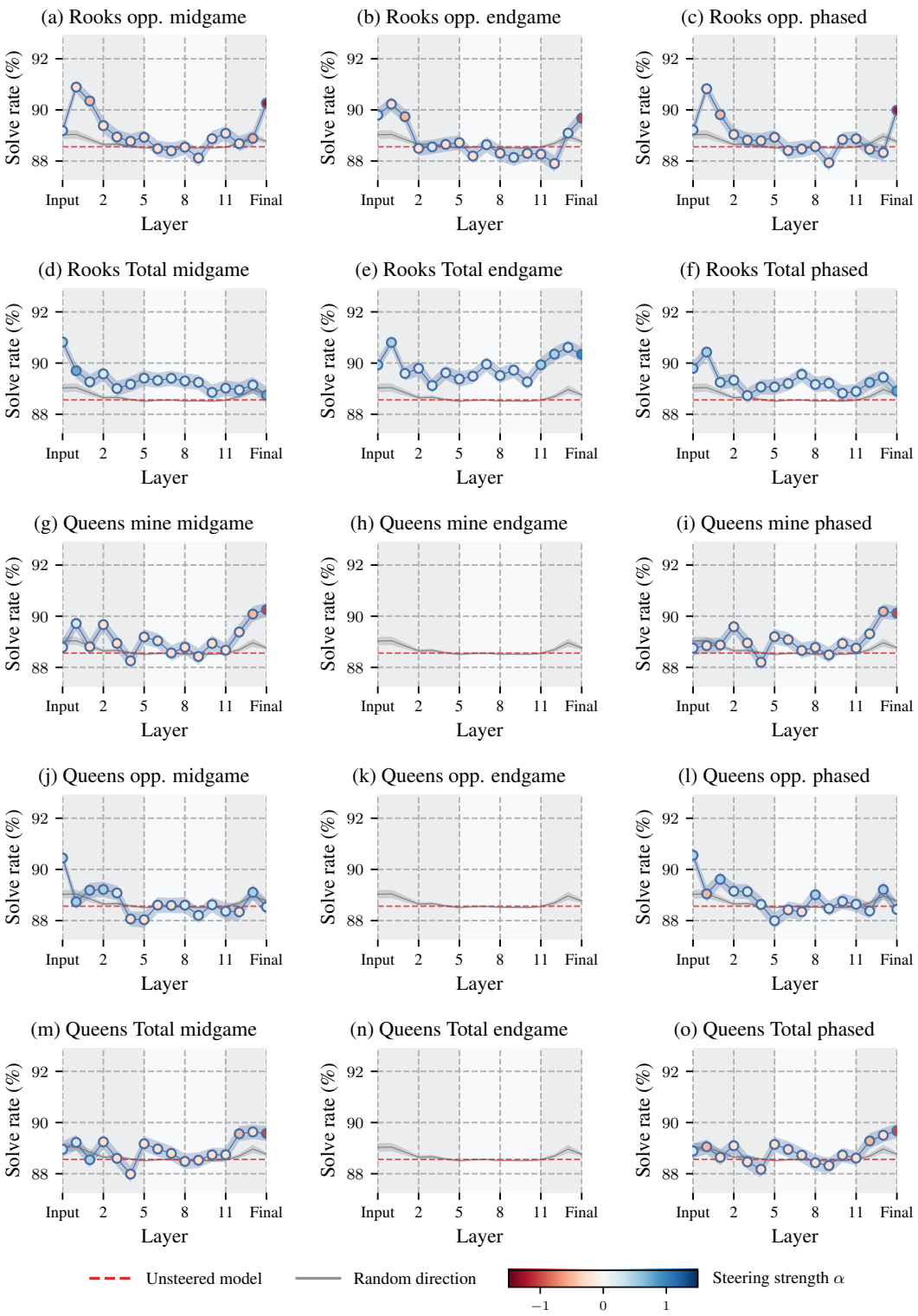


Figure 33: Concept steering effects across layers (Part 3 of 7). Each panel shows the held-out solve rate for the given concept at each intervention layer: the strength is selected on a random half of the puzzles and evaluated on the other half, averaged over 200 random splits, and the shaded band shows one standard deviation across splits. Marker colors indicate the most frequently selected strength, **blue** for positive and **red** for negative values. The **gray line and band** show the mean and 95% CI across 50 random directions under the same protocol. Panels without a curve are terms that are zero in every position, so no steering vector exists. The **red dashed line** shows the unsteered baseline.

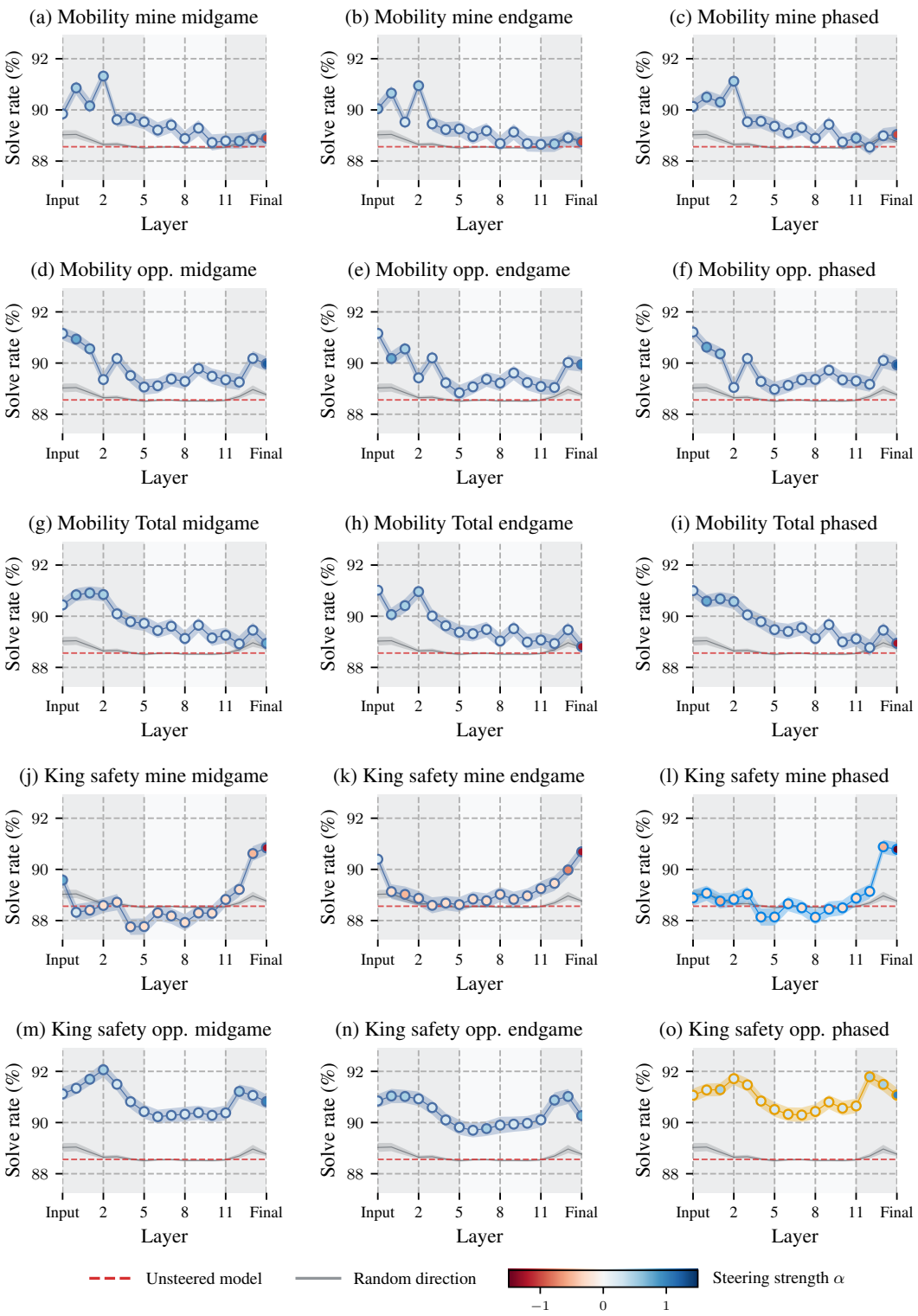


Figure 34: Concept steering effects across layers (Part 4 of 7). Each panel shows the held-out solve rate for the given concept at each intervention layer: the strength is selected on a random half of the puzzles and evaluated on the other half, averaged over 200 random splits, and the shaded band shows one standard deviation across splits. Marker colors indicate the most frequently selected strength, **blue** for positive and **red** for negative values. The **gray line and band** show the mean and 95% CI across 50 random directions under the same protocol. Panels without a curve are terms that are zero in every position, so no steering vector exists. The **red dashed line** shows the unsteered baseline.

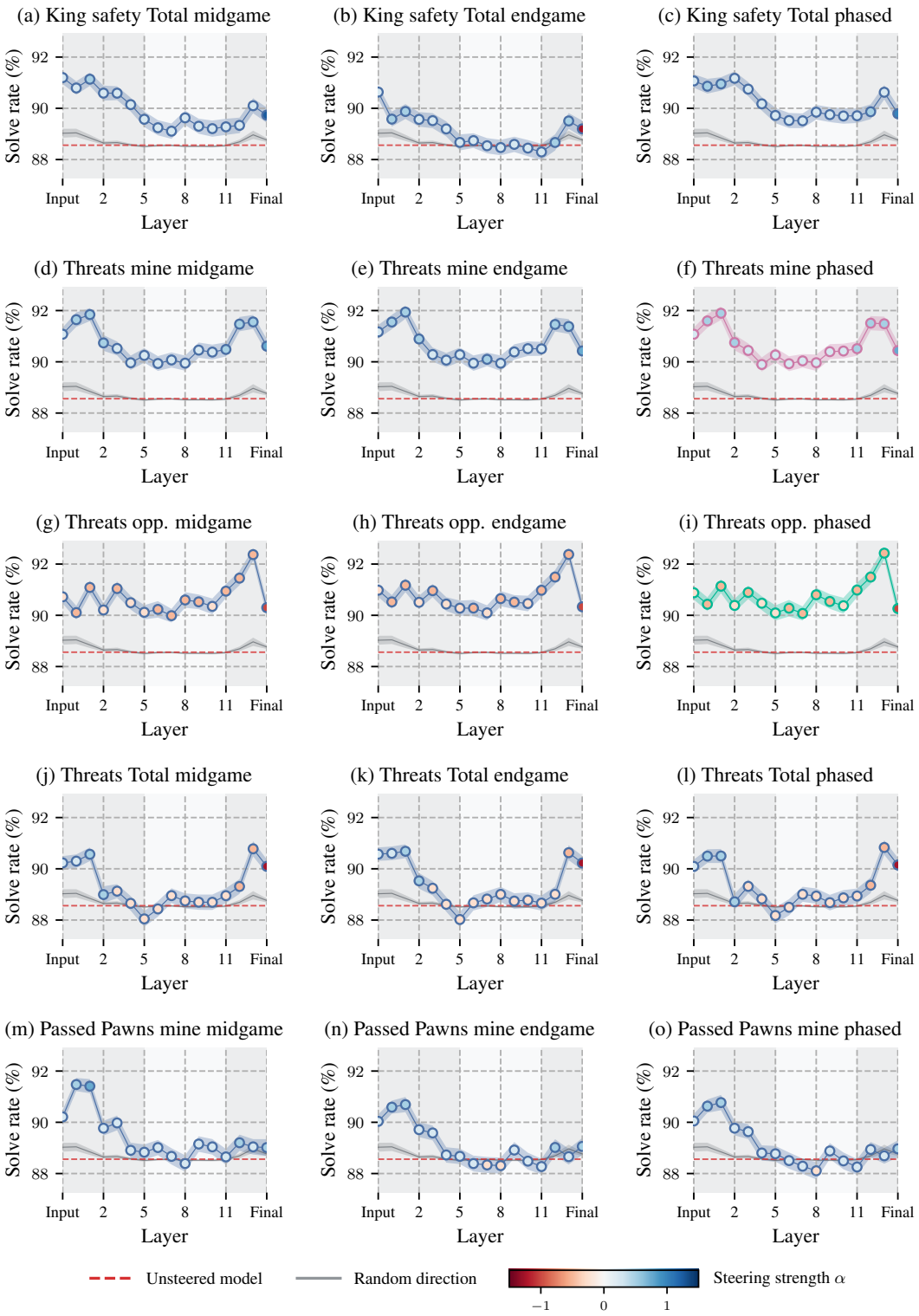


Figure 35: Concept steering effects across layers (Part 5 of 7). Each panel shows the held-out solve rate for the given concept at each intervention layer: the strength is selected on a random half of the puzzles and evaluated on the other half, averaged over 200 random splits, and the shaded band shows one standard deviation across splits. Marker colors indicate the most frequently selected strength, **blue** for positive and **red** for negative values. The **gray line and band** show the mean and 95% CI across 50 random directions under the same protocol. Panels without a curve are terms that are zero in every position, so no steering vector exists. The **red dashed line** shows the unsteered baseline.

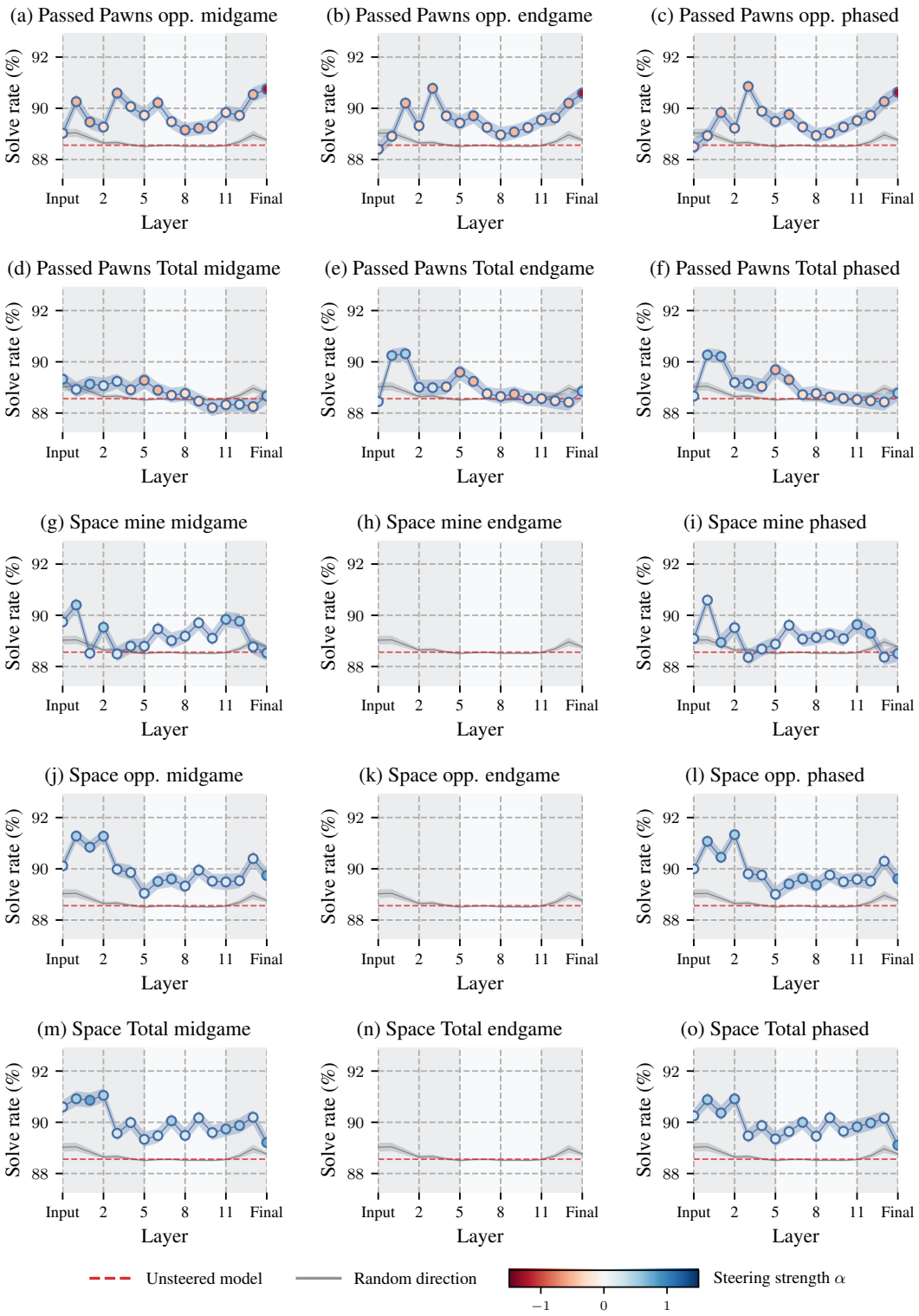


Figure 36: Concept steering effects across layers (Part 6 of 7). Each panel shows the held-out solve rate for the given concept at each intervention layer: the strength is selected on a random half of the puzzles and evaluated on the other half, averaged over 200 random splits, and the shaded band shows one standard deviation across splits. Marker colors indicate the most frequently selected strength, **blue** for positive and **red** for negative values. The **gray line and band** show the mean and 95% CI across 50 random directions under the same protocol. Panels without a curve are terms that are zero in every position, so no steering vector exists. The **red dashed line** shows the unsteered baseline.

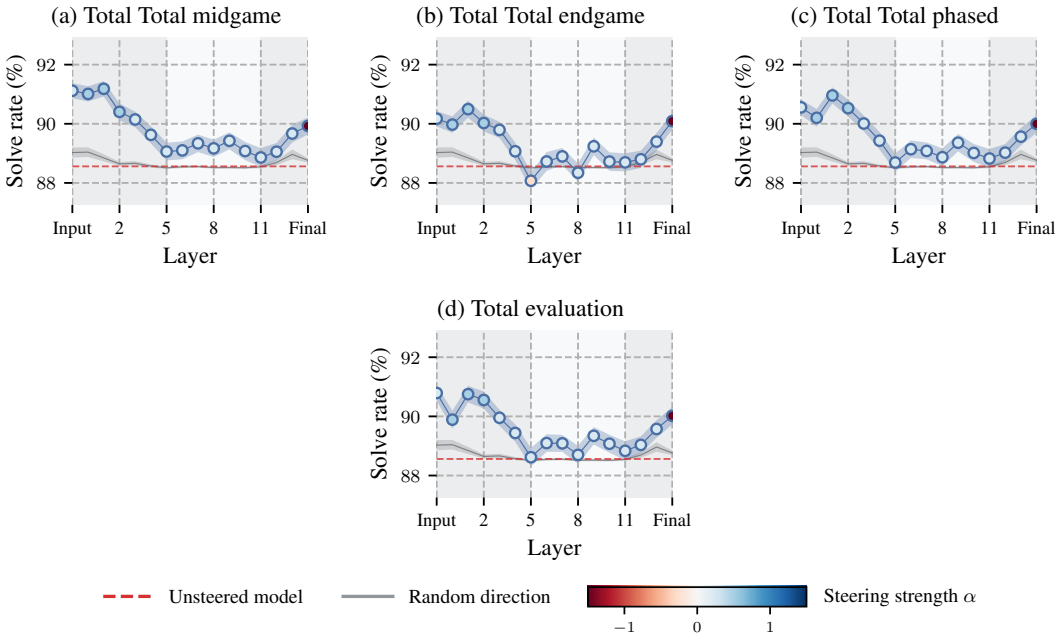


Figure 37: Concept steering effects across layers (Part 7 of 7). Each panel shows the held-out solve rate for the given concept at each intervention layer: the strength is selected on a random half of the puzzles and evaluated on the other half, averaged over 200 random splits, and the shaded band shows one standard deviation across splits. Marker colors indicate the most frequently selected strength, **blue** for positive and **red** for negative values. The **gray line and band** show the mean and 95% CI across 50 random directions under the same protocol. Panels without a curve are terms that are zero in every position, so no steering vector exists. The **red dashed line** shows the unsteered baseline.

P Search experiments

Setup We run both experiments of Section 3.4 on the 8,648 forgotten and 23,601 solved puzzles, starting each search from the puzzle position with the full move history of its game, as in all other analyses, and scoring only the first move of the solution. We compare two networks: the final-layer network is the unmodified network analyzed throughout the paper, and the penultimate-layer network is identical except that its policy head reads the logit lens at the penultimate layer, while its value and moves-left heads still use the full model. All searches run in Leela's engine (Leela Chess Zero team), version 0.33 at commit d8ce482, with the final-layer network on the CUDA backend and the penultimate-layer network on the ONNX backend. To check that the engine uses the policy we analyze, we compare its prior at a single node, with temperature one and smart pruning off, against our own implementation. The top move agrees in at least 99.58% of puzzles for each network and dataset, and move probabilities differ by at most 0.011. The remaining differences are numerical, since the engine's two backends and our implementation compute the same network with different kernels. Every disagreement on the top move is a near-tie, in which the two most likely moves differ by at most 0.0034 in our implementation.

Training target For the training target, we run the self-play search described in Appendix D.4 with the settings of the Leela run that generated the network's training data, listed in Table 39, to obtain the target the training procedure would assign to each puzzle. Each puzzle is searched once until the KL-divergence stop fires, which happens after a median of 861 visits on forgotten and 809 on solved puzzles, and always before the cap of 10,000. Since the engine has no seed for the root noise, each puzzle gets a single noise draw. Because unvisited root moves enter the selection rule with the value of a certain win, every root move is evaluated at least once in practice.

Deployment For deployment, we use the engine's default settings and limit each search to 1, 8, 32, 128, 800 or 2,048 nodes, running each budget separately. Smart pruning is on, as in normal use, so a search can stop before reaching its budget. Here we count a puzzle as solved if the engine plays the first move of the solution.

Table 39: Search settings for the training target, taken from the self-play run that generated the network's training data, and for deployment, which uses the engine's defaults. In both profiles, c_{puct} grows with the visit count with the engine's default factor 3.894 and base 38,739, and a twofold repetition counts as a draw within the search.

Setting	Training target	Deployment
Policy temperature	1.45	1.359
c_{puct}, tree / root	1.32 / 2.0	1.745 / 1.745
First-play urgency, tree / root	reduction 0.26 / absolute 1.0	reduction 0.33 / reduction 0.33
Moves-left utility	max. effect 0.2, threshold 0, slope 0.007, quadratic 0.85, scaled 0.15, constant 0	max. effect 0.0345, threshold 0.8, slope 0.0027, quadratic −0.6521, scaled 1.6521, constant 0
Root noise	Dirichlet, $\varepsilon = 0.1$, $\alpha = 0.12$	none
Smart pruning	off	factor 1.33
Stopping	KL gain below $5 \cdot 10^{-5}$ per visit, checked every 100 visits, at most 10,000 visits	budget of 1 to 2,048 nodes

Results Table 40 gives the training target and Table 41 the deployment solve rates on both datasets. Stockfish confirms that the solution wins in all 49 forgotten puzzles, 0.6%, in which the training target misses it. In most of them, the value head rates the position after the solution as losing, so the search spends few visits on it. Even at 2,048 nodes, the penultimate-layer network solves slightly more forgotten puzzles, 99.3% against 99.1%, and only 38 forgotten puzzles are never solved by it at any budget, against 71 for the final-layer network. On solved puzzles, the penultimate-layer network catches up quickly: from 76.7% at a single node, it reaches 98.6% at 8 nodes and 99.6% at 32, so the final layer's advantage there is limited to the smallest budgets.

Table 40: Training target on forgotten and solved puzzles, compared with the policies read from the final and the penultimate layer.

	Forgotten	Solved
Puzzles	8,648	23,601
Visits when the search stops, median	861	809
Target probability on the solution, mean	0.931	0.953
Target's top move is the solution	99.4%	100.0%
Solution has the highest value at the root	99.5%	$>$ 99.99%
Final-layer probability on the solution, median	0.159	0.795
Penultimate-layer probability on the solution, median	0.262	0.469
Penultimate-layer top move agrees with target	43.9%	76.7%
Cross-entropy to target, final layer (nats)	2.223	0.523
Cross-entropy to target, penultimate layer (nats)	1.716	1.026
Penultimate layer closer to target than final layer	77.5%	7.3%

Table 41: Share of puzzles in which the engine plays the solution's first move, by search budget in nodes, with engine defaults.

Puzzles	Network	1	8	32	128	800	2,048
Forgotten	Final layer	0.0%	53.7%	78.3%	92.0%	97.8%	99.1%
	Penultimate layer	43.7%	75.3%	86.4%	94.8%	98.6%	99.3%
Solved	Final layer	100.0%	99.98%	$>$ 99.99%	$>$ 99.99%	100.0%	100.0%
	Penultimate layer	76.7%	98.6%	99.6%	99.9%	99.97%	99.99%

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state four main claims: (1) extending the logit lens to Post-LN architectures yields meaningful intermediate policies (Section 3.1), (2) forgotten puzzles exist and look-ahead functions normally in these positions (Section 3.2), (3) learned safety priors override these algorithmically computed solutions (Section 3.3), and (4) the training objective asks for the solution on these positions, so forgotten puzzles are failures of the network rather than of its objective (Section 3.4). The overarching claim that algorithmic structure does not guarantee algorithmic behavior rests on their combination.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 states the limitations: (1) we do not provide a precise mechanistic description of how the safety priors are implemented, (2) our causal analyses are conducted on a single network, although the phenomenon itself is verified across several Leela networks, and (3) Stockfish evaluation terms serve as proxies for the model's internal concepts. Appendix A discusses further limitations and directions.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper is primarily empirical. The Post-LN logit lens extension is justified through a decomposition analysis (Appendix C) based on Mickus et al. (2022), but no novel theoretical results are claimed.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 2 and Appendices L, M, N, O and P give the full experimental details, including the search settings. Our code is publicly available at `https://github.com/hartigel/leela-logit-lens`. Stockfish and Leela Chess Zero are publicly available.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Our code is publicly available at `https://github.com/hartigel/leela-logit-lens`, with instructions to reproduce the main results. All source data are publicly available: Lichess puzzles (CC0), the CCRL dataset, and the Encyclopedia of Chess Openings positions from Ruoss et al. (2024). The model is from Leela Chess Zero and the inference framework from Jenner et al. (2024). The derived forgotten and solved datasets are released with the code. Licenses are listed in Appendix B.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All experimental settings are specified in Section 2 and the appendices. Probe training hyperparameters follow Jenner et al. (2024) (Appendix L). Steering configurations are detailed in Appendix O. Tournament settings are in Section 2.3. Search settings are listed in Appendix P.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Error bars and confidence intervals are reported throughout. Tournament Elo ratings include asymmetric confidence bounds in Appendix F, and puzzle solve rates are deterministic given the model and dataset. Activation patching, head ablation, and probing error bars follow the methodology of Jenner et al. (2024). Concept preferences are reported with 95% CI. Steering results are held-out estimates from Monte Carlo cross-validation, reported as the mean and standard deviation over 200 random splits, and the steering random baseline reports 95% CI over 50 random vectors. In principle, variance over different gradient computation samples could be measured, but this is not standard practice in activation steering and would be computationally prohibitive. In the search experiments, deployment runs are deterministic up to floating-point differences, while the training-target search includes the root Dirichlet noise of the training procedure, so each puzzle's target is a single random draw; reported values are averages over all puzzles in each set.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute resources including hardware specifications and approximate runtimes for all experiments are provided in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics. We analyze an existing publicly available chess model and publicly available datasets. No human subjects, private data, or dual-use concerns are involved.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The Impact paragraph in Section 5 discusses broader implications for mechanistic interpretability and connects our findings to related phenomena in language models. This is foundational interpretability research on a chess model with no direct negative societal applications.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

Justification: The paper analyzes an existing chess model and does not release models or data with risk of misuse.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All assets are properly credited, and their licenses are listed in Appendix B: Lichess puzzles (CC0); the puzzle dataset, Encyclopedia of Chess Openings positions, and tournament code from Ruoss et al. (2024) (Apache 2.0); Leela Chess Zero, Stockfish, and the inference framework from Jenner et al. (2024) (GPLv3); lichess-bot (AGPLv3); and the CCRL dataset, which is publicly available without a stated license.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Our code is publicly released at `https://github.com/hartigel/leela-logit-lens` under GPLv3, with README documentation. The forgotten and solved datasets are filtered subsets of publicly available Lichess puzzles (CC0), constructed with the criteria in Section 2.4 and released with the code.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.

- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subjects research was conducted.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were used for writing assistance and coding support but not as a component of the core methodology. Per NeurIPS policy, declaration is not required for writing, editing, or formatting purposes.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.