Decoding Chess Mastery: A Mechanistic Analysis of a Chess Language Transformer Model

Austin L. Davis[0000−0003−0603−0975] and Gita Sukthankar[0000−0002−6863−6609]

University of Central Florida, Orlando, FL, US
{austindavis,gitasukthankar}@ucf.edu

Abstract. Mechanistic interpretability (MI) studies aim to identify the specific neural pathways that underlie decision-making in neural networks. Here we analyze both the horizontal and vertical information flows of a chess-playing transformer. This paper introduces a new taxonomy of chessboard attention patterns that synchronize to guide move selection. Our findings show that the early layers of the chess transformer correctly identify moves that are highly ranked by the final layer. Experiments conducted on human chess players laid the foundation for much of our current understanding of human problem-solving, cognition, and visual memory. We believe that the study of chess language transformers may be an equally fruitful research area for AGI systems.

Keywords: chess cognition · mechanistic interpretability · transformers

1 Introduction

This paper presents a mechanistic analysis of the operation of a chess language transformer model. In contrast to other surface-level forms of interpretability that simply correlate inputs with outputs or highlight which features were important for a decision [13], the mechanistic interpretability community [14] seeks to uncover the computational processes that occur within a neural network, literally reverse-engineering the algorithms to reveal how the model arrived at its conclusions. This approach moves beyond treating the model as an opaque "black box" and instead examines the model's choices through the lens of its internal operations. Reverse-engineering transformer models has become an important area of research interest for two reasons:

1. transformers are a key building block for many of the SOTA vision, language, and multimodal visual-language systems (e.g., ViT [7]), ChatGPT, Gemini, and CLIP [17]);
2. the architectural elegance of the transformer makes it more amenable to certain types of circuit analysis. Elhage et al. [8] note that if MLP layers are removed, the residual stream of a transformer is composed of linear and additive operations, in which the attention heads operate independently.

This paper specifically examines the operation of the attention heads in a chess language transformer and proposes a taxonomy of chess attention mechanisms.

Although there are several chess language transformer models, we selected Toshniwal et al.'s Learning Chess Blindfold (LCB) model [20] for our analysis, due to its manageable size. LCB uses the GPT-2 small architecture [18] and is trained on sequences from human chess games. This paper explores the distribution of attention patterns per layer. Our results show that early layers of the LCB are very successful at predicting moves that are ultimately highly ranked by the final layer.

2 Related Work

2.1 Chess and Cognition

There is a long tradition of using chess as a benchmark for artificial intelligence systems ranging from specialized parallel search algorithms like Deep Blue [3] to more generalized machine learning approaches such as AlphaZero [11]. However, our work is inspired by research from the cognitive science community that endeavors to model chess mastery in humans. Charness [4] quantified the impact of chess on cognitive science by doing a literature review of published studies referencing the seminal article by Simon and Chase [5] that proposed the usage of chess for cognitive science research. Neural imaging studies [9] on human chess players have shown that expert chess players exhibit some differences in grey matter within the occipitotemporal junction (OTJ), a region linked to various perceptual processes. FMRI studies of chess [2] show increased activity in brain regions associated with spatial reasoning (parietal areas) with less activity in the frontal lobe. These findings suggest that spatial processing may play a more prominent role than decision-making in human chess cognition. Our study of chess attention patterns in transformers parallels these human fMRI studies. However most of the human fMRI studies are conducted by simply showing the subjects images of chess boards, whereas we examine average attention patterns across many games.

2.2 Mechanistic Interpretability (MI)

In this paper, we apply several techniques from the MI community to a chess-playing language model. Mechanistic interpretability is a newer area of explainable AI in which the aim is to reverse engineer neural circuits [14]. This can be accomplished through a variety of techniques including linear probes, ablation studies, and activation patching [19]. However, all these methods rely heavily on human inspection. To ameliorate this limitation, Conmy et al. [6] recently proposed a method for automating the circuit detection step that successfully rediscovered circuits in GPT-2, identified by prior work.

Many researchers believe that large language models obey the *linear representation hypothesis* [16]. This theory suggests important concepts that the model learns are encoded in a linear way within its hidden layers, facilitating the probe training process. Linear probes have been used in several of the prior

studies on game play language models to determine how the transformer encodes game state [10, 15, 20]. In this paper, we ignore the question of world state and simply examine the horizontal and vertical information flows.

3 Method

3.1 The Learning Chess Blindfolded (LCB) model

Toshniwal et al.'s Learning Chess Blindfolded (LCB) model [20] utilizes the GPT-2 small architecture [18] with 12 attention heads in each of its 12 transformer layers. LCB is a causal language model trained to minimize loss in predicting the next token in chess games. Its inputs are sequences of chess moves (game traces) given in Universal Chess Interface (UCI) notation. UCI notation describes chess moves by specifying the starting and ending square of a piece's movement. For example, the move of a knight from b1 to c3 is noted as "b1c3". Figure 1 shows how a game trace in UCI notation correspond to the chess board state.

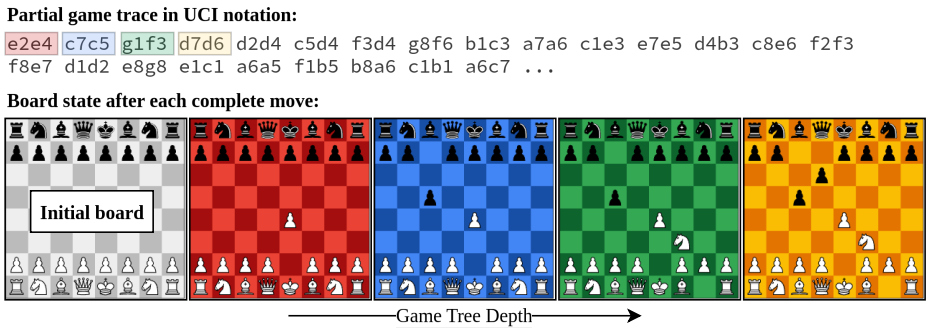

Fig. 1. UCI notation of a partial game trace and the resulting board states after each move. The game starts from the initial board position, and the first move shown in red moves the white pawn from e2 to e4. LCB treats the start and end tiles of a single move as separate tokens. Thus, each move requires two tokens.

To input a game trace into the LCB model, each UCI move is divided into two tokens: one for the starting tile and another for the destination tile, each processed separately. The model outputs a non-normalized vector over its vocabulary (see Table 1), which is converted into a probability distribution using the `softmax` operation. Predicting a chess move involves inputting a game trace prefix and prompting the model twice. In the first forward pass, the starting tile is selected from the output probability distribution, appended to the trace, and the model is prompted again to determine the destination tile. This interactive process is facilitated through a Google Colab notebook[1].

[1] See `colab.research.google.com/drive/125y4MpnSWAakoSE5My9jGMtBExbjqTpW`

Table 1. LCB Vocabulary

Type	Examples	Count
Square names	`e4`, `d1`	64
Promotion type	`q`, `r`, `b`, `n`	4
Special symbols	BOS, EOS, PAD	9
Total		77

LCB predicts legal moves with an accuracy of 97.7% [20]. In practice, we find it can regularly play thirty complete moves before it recommends an illegal move. This is impressive when considering that the model was trained with no *a priori* knowledge of the game of chess; it was trained exclusively to minimize loss when predicting the next token in UCI move sequences. Consequentially, LCB does not play to win; it simply anticipates which move is most likely to occur next. As a result, LCB plays stronger moves if the input sequence resembles high-skill gameplay and weaker moves when the input sequence resembles low-skill gameplay.

Our intent was to localize the internal mechanisms which allow it to generate legal moves with such high accuracy. Importantly, our use of the LCB model is strictly out-of-the-box, meaning we have not modified its configuration or parameterization. In a similar way that a neural scientist might localize a cognitive function by stimulating a patient with sound or images during an MRI scan, we stimulate the model by performing forward passes on a large number of chess games and record the values of the model's hidden states.

3.2 Dataset

We utilize a dataset of 1000 chess games sourced from the lichess.org open database. The games were converted into UCI notation and deduplicated. Each game in the dataset is truncated to the first 20 tokens (representing the first 10 moves) to center analysis on opening move sequences, provide consistency when computing metrics, and focus on games without pawn promotions since promotion is handled differently by the model and tokenizer.

3.3 Activation Caching and Direct Logit Attribution (DLA)

During each forward pass, the hidden state $h_i^{(\ell)}$ is captured post each transformer layer in an *activation cache*. This activation cache is the set

$$\text{Activation Cache} = \{h_i^{(\ell)} : i \in [1, 12] \text{ and } \ell \in [1, \texttt{numTokens}]\}$$

We employ DLA [1] to assess the impact of each layer on the decision-making process of LCB. By applying the GPT's `unembed` matrix directly to the captured hidden states, we can back-translate these intermediate representations from the latent embedded space to the output vocabulary. By analyzing the evolution of these intermediate states, we can uncover how each layer influences the final move prediction. This process is illustrated in Figures 2 and 3.

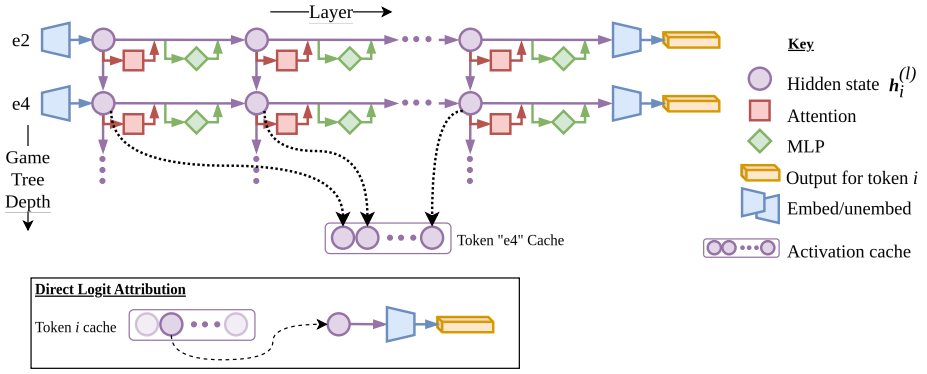

Fig. 2. Caching hidden states within the GPT-2 architecture. DLA applies the `unembed` matrix to a vector in the activation cache to transform latent hidden state vectors into a vector in the vocabulary (output) space. By comparing DLA vectors across layers we can see how each layer contributes to the final output of the LCB model. The unfolded representation of the GPT-2 architecture was inspired by [12].

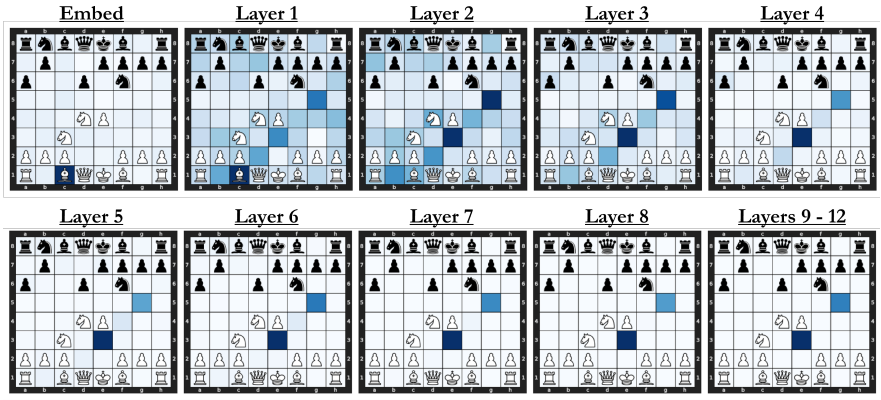

Fig. 3. DLA visualization. The model selects the destination square for the bishop on `c1`. Boards are colored according to the softmax of the unembedded hidden state for each layer starting from the token embedding (top-left) through layer 12 (bottom-right); richer blue colors indicate stronger activations. Layer 1 activations highlight the bishop on c1 (the piece about to be moved), and intermediate layers process information in the previous hidden states to select a destination for that bishop. The model's final (actual) output, `e3`, corresponds to the darkest tile on the bottom-right board. The model's preference for `e3` is evident as early as layer 2.

4 Results

We performed a forward pass on 1,000 unique game traces with at least 20 tokens, and cached the attention head activations for every head, layer, and game trace. We averaged these activations over the 1,000 games to obtain a 20×20 activation pattern for all 144 ($= 12 \times 12$) attention heads. Inspecting these attention activations, we observed several, strong structural patterns shown in Figure 4. Each pattern emerges from the training process and implies these heads are selecting information in a very predictable way that corresponds with the regular structure of UCI game traces.

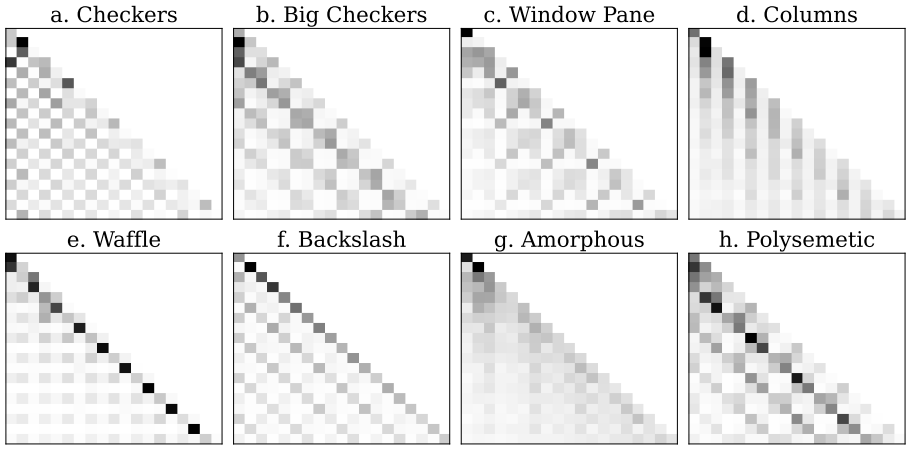

Fig. 4. Structural patterns in LCB attention heads. Each pattern (a-h) is a prototypical example. Values are averages over 1K game traces which range from 0 (white) to 1 (black) with $\sigma = 0.133$. The top row includes the most frequently observed patterns. The bottom row includes additional noteworthy patterns.

We interpret these attention patterns from the perspective of UCI game traces. (Note: a full move is processed in either 2 forward passes (for a single player) or 4 forward passes (to complete a full turn for both white and black players).

The following patterns were observed multiple times in the attention heads of the LCB model. Their frequencies are reported in Table 2:

a. **Checkers**. When selecting a source/destination tiles, these heads attend to the history of previous source/destination tiles. These heads do not appear to have a limited context window, and the full history of moves are typically attended to.
b. **Big Checkers** Similar to the checkers pattern, except rather than attending to the history of tiles, these heads attend to the history of full moves.

Depending on the horizontal offset of the grid, these heads either attend to the current player's move history or the opponent's move history.

c. **Window Pane** These heads have two modalities. In the prototypical example above, LCB attends only to white player's destination tile history exactly when selecting the destination tile for white player pieces. During the other phases of movement, the head attends approximately equally to all positions.
d. **Columns** These heads attend to the history of a single phase of movement, i.e. either destination or source tile selection depending on the horizontal offset of the columns.

The following patterns were observed at least once, and were noteworthy because of their strong structural regularity:

e. **Waffle** This pattern occurred once in layer 3 head 5 (L3H5). When selecting a destination, the head attends only to the previous source tile (because of the strong diagonal), and when selecting a source tile, the head attends to both players' previous destinations equally.
f. **Backslash** (L1H5 and L1H12) This is a special case of the checkers pattern, but instead of looking back one phase, these heads look back four tokens to the player's own source/destination history.
g. **Amorphous** (L12 H1-H3) These heads occurred exclusively in the final layer. Unlike other heads which have a high variance, these heads are diffuse, indicating the head is likely highly state-dependent.
h. **Polysemetic** Many heads can be described as a combination of multiple other patterns, encoding multiple modalities at once. The head pictured here (L4H12) is simultaneously a Big Checkers and a Backslash pattern.

Table 2. Frequency Counts by Layer for Top Four Attention Patterns

Layer	Checkers	Big Checkers	Columns	Window Pane
Layer 1				1
Layer 2	1		1	1
Layer 3	1	2	1	
Layer 4	1	3	1	
Layer 5		5	1	1
Layer 6		5		2
Layer 7		3		2
Layer 8				
Layer 9		1	3	
Layer 10		1		
Layer 11				
Layer 12				
Total	3	20	7	7

4.1 Comparing DLA Output Vectors Across Layers

A key question is when the LCB model completes its computation during the forward pass. By comparing the final output vector with a DLA vector from any layer, we can use their similarity as an indicator. A small metric value between layer ℓ and layer 12 suggests that computation is largely complete by layer ℓ.

We employ three metrics to evaluate similarity between DLA vectors and the final output. **Cosine similarity** measures vector alignment, with higher values indicating greater similarity. **Precision at K (P@K)** evaluates how well the top K predictions from earlier layers match the final prediction, at levels $K = 1$, $K = 2$, and $K = 5$. To address gaps left by these metrics, we introduce **Magnitude Precision at K (M-P@K)**, which considers vector magnitudes.

Let S be the scores tensor and T be the target tensor (i.e. the unnormalized output of layer 12). Define T' as the softmax-normalized version of T. Select the top k entries from S and T', denoted as S_k and T'_k respectively. Let idx_k be the indices corresponding to the top k entries from S. The corresponding targets from T' at these indices are $T'[\text{idx}_k]$. Compute the following magnitudes:

$$\text{Mag}_{\text{sel}} = \sum T'[\text{idx}_k], \quad \text{and} \quad \text{Mag}_k = \sum T'_k$$

Then the Magnitude Precision at K (MP@K) is the ratio of these magnitudes: $\text{MP@K} = \text{Mag}_{\text{sel}}/\text{Mag}_k$. Figure 5 and Figure 6 show the values of these metrics over the 1000 game traces used to perform the attention head analysis, the mean is taken over token positions $i = 1, ...20$.

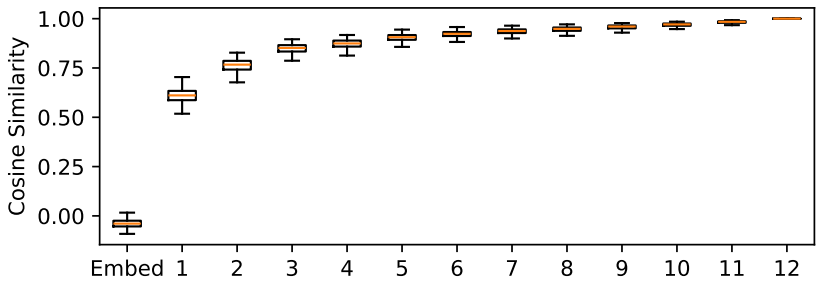

Fig. 5. Cosine similarity for DLA of h_i^ℓ versus output at layer 12. *Embed* is the hidden state vector prior to layer 1, just after `embed()` on the token id.

Figure 5 reveals significant insights. The `embed()` function's vector is nearly perpendicular to the final output vector, aligning with observations from Figure 3 that the Embed layer indicates the moved piece based on the latest input token `c1`, rather than its destination. By layer $\ell = 9$, much of the computation is already complete as the median cosine similarity reaches 92%.

In interpreting Figure 6, note the presence of multiple viable chess moves per position, affecting the stability of rank-based metrics like P@K and M-P@K. The

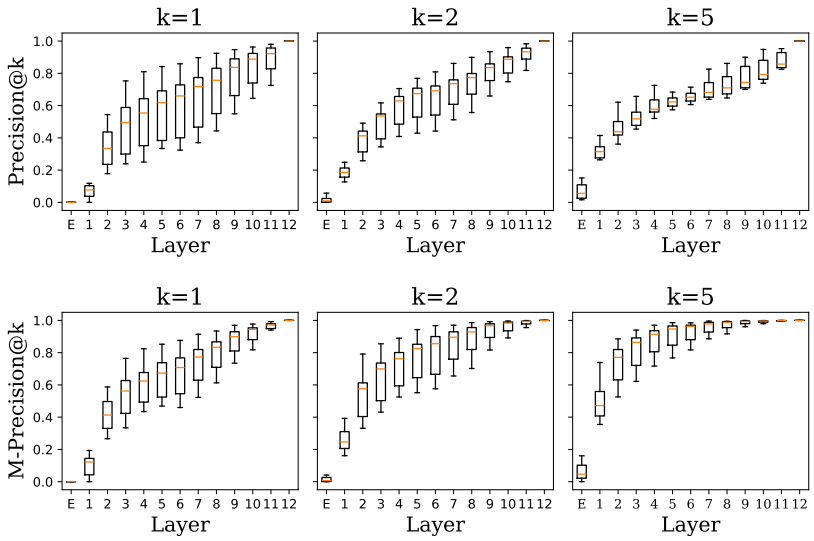

Fig. 6. Precision at k and Magnitude (M-)Precision at k by layer. E=*Embed*

large inner quartile range of $P@1$ across all layers, which narrows significantly for $k > 1$, suggests that while the model often identifies several plausible moves early, the optimal move typically emerges in the last three layers. Additionally, the relative stability of the M-P@K metric from $k = 1$ to $k = 2$ indicates that even if the top-ranked move at earlier layers differs from the final layer, the quality of alternative choices remains high.

5 Conclusion

This paper presents a mechanistic analysis of the LCB chess-playing transformer. We introduce a new taxonomy of chessboard attention patterns and describe how they can be used to guide chess play. The first seven layers of the transformer use our attention patterns frequently. According to our ranked move analysis, these are also the layers that contribute the most to move selection. High metric values for layers $\ell \geq 8$, combined with the distribution of attention head patterns in Table 2 and manual reviews of numerous DLA sequences such as those shown in Figure 3, suggest a significant interaction between structured attention heads and move selection in the LCB model. However, the function of the final five layers remains inconclusive. They likely play a role in long-term strategy, which is difficult to analyze using current techniques.

Acknowledgments. The conclusions and opinions expressed in this research paper are those of the authors and do not necessarily reflect the official policy or position of the U.S. Government or Department of Defense.

References

1. Interpreting GPT: The Logit Lens, https://www.lesswrong.com/posts/interpreting-gpt-the-logit-lens
2. Atherton, M., Zhuang, J., Bart, W.M., Hu, X., He, S.: A functional MRI study of high-level cognition. I. the game of chess. Cognitive Brain Research **16**(1), 26–31 (2003)
3. Campbell, M., Hoane Jr, A.J., Hsu, F.h.: Deep Blue. Artificial intelligence **134**(1-2), 57–83 (2002)
4. Charness, N.: The impact of chess research on cognitive science. Psychological Research **54**, 4–9 (1992)
5. Chase, W.G., Simon, H.A.: The mind's eye in chess. In: Visual Information Processing, pp. 215–281. Elsevier (1973)
6. Conmy, A., Mavor-Parker, A., Lynch, A., Heimersheim, S., Garriga-Alonso, A.: Towards automated circuit discovery for mechanistic interpretability. Advances in Neural Information Processing Systems **36**, 16318–16352 (2023)
7. Dosovitskiy, A., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
8. Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., et al.: A mathematical framework for transformer circuits. Transformer Circuits Thread **1**, 1 (2021)
9. Hänggi, J., Brütsch, K., Siegel, A.M., Jäncke, L.: The architecture of the chess player's brain. Neuropsychologia **62**, 152–162 (2014)
10. Li, K., Hopkins, A.K., Bau, D., Viégas, F., Pfister, H., Wattenberg, M.: Emergent world representations: Exploring a sequence model trained on a synthetic task. In: International Conference on Learning Representations (2023)
11. McGrath, T., Kapishnikov, A., Tomašev, N., Pearce, A., Wattenberg, M., Hassabis, D., Kim, B., Paquet, U., Kramnik, V.: Acquisition of chess knowledge in AlphaZero. Proceedings of the National Academy of Sciences **119**(47), e2206625119 (2022)
12. Meng, K., Bau, D., Andonian, A., Belinkov, Y.: Locating and editing factual associations in gpt. NIPS **35**, 17359–17372 (2022)
13. Minh, D., Wang, H.X., Li, Y.F., Nguyen, T.N.: Explainable artificial intelligence: a comprehensive review. Artificial Intelligence Review pp. 1–66 (2022)
14. Nanda, N.: A comprehensive mechanistic interpretability explainer and glossary, https://www.neelnanda.io/mechanistic-interpretability/glossary
15. Nanda, N., Lee, A., Wattenberg, M.: Emergent linear representations in world models of self-supervised sequence models (2023)
16. Park, K., Choe, Y.J., Veitch, V.: The linear representation hypothesis and the geometry of large language models (2023)
17. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning. pp. 8748–8763. PMLR (2021)
18. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2019)
19. Räuker, T., Ho, A., Casper, S., Hadfield-Menell, D.: Toward transparent AI: A survey on interpreting the inner structures of deep neural networks (2023)
20. Toshniwal, S., Wiseman, S., Livescu, K., Gimpel, K.: Chess as a testbed for language model state tracking. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 11385–11393 (2022)