HOW GPT LEARNS LAYER BY LAYER

Jason Du (3040815573), Kelly Hong (3037818786), Alishba Imran (3037052137),
Erfan Jahanparast (3036109049), Mehdi Khfifi (3039377109), Kaichun Qiao (3040815534)*
Fundamentals Track
4 Units
Group Name: Interpretability Kingdom

ABSTRACT

Large Language Models (LLMs) have been shown to excel at tasks like natural language processing, strategic gameplay, and complex reasoning. However, their ability to develop robust and generalizable internal representations, essential for reasoning and decision-making in agentic systems, remains an open question. For agents to effectively navigate complex environments, they must construct reliable world models. While LLMs often perform well on task-specific benchmarks, they frequently fall short in terms of generalization, failing to fully encode the compositional structure of their underlying domains. This limitation can lead to brittle representations that degrade their effectiveness in real-world applications. Investigating how LLMs construct these internal world models is critical to advancing the development of agents capable of maintaining coherent, adaptive, and strategic behaviors across diverse tasks.

In this work, we investigate the progression of learned features in OthelloGPT, a GPT-based model trained on the game of Othello, using it as an interpretability testbed to analyze layer-wise representations. We chose OthelloGPT as a controlled environment to study the evolution of representations. Although it is trained solely on next-token prediction with randomly generated valid moves, not explicitly on strategy or game rules, we still observe meaningful patterns across layers, revealing how the model develops an understanding of the board state and gameplay. We uncover a hierarchical progression in the learned features, where specific layers encode different attributes, such as the edges of the board, board shape, and others are potentially capturing more dynamic changes in the tiles as the game progresses and tiles flip. To better understand these representations, we compare Sparse Autoencoders (SAEs) and traditional linear probes as interpretability tools. Our experiments reveal that SAEs provide a more robust and disentangled decoding of the features the model is learning, particularly for compositional attributes, whereas linear probes primarily capture features that are strong predictors for classification tasks.

Key experiments we run include decoding features related to tile color and tile stability, a previously unexamined feature that reflects complex gameplay concepts like board control and long-term planning. We study the progression of linear probe accuracy and tile color using both SAE's and linear probes to compare their effectiveness at capturing what the model is learning. Although we begin with a smaller language model, OthelloGPT, this study establishes a framework for understanding the internal representations learned by GPT models, transformers, and LLMs more broadly. Our findings highlight the importance of robust and interpretable world models in enabling agent-like behavior, as such models allow agents to maintain consistent, adaptive, and strategic actions over time.

Project links: Interactive Demo, Code, Presentation, and Video.

[*] In alphabetical order.

1 Introduction

Large language models (LLMs) exhibit remarkable capabilities across tasks like natural language processing, strategic games, and reasoning. However, their demonstrated proficiency in performing complex reasoning tasks raises an open question: Do LLMs construct accurate internal representations of the structures, rules, and patterns that underlie the data they are trained on, or are these representations incomplete and brittle? A recent survey Chang et al. (2024) on LLM evaluation highlights the need for multidimensional assessment, emphasizing that current models often succeed in task-specific metrics but lack deeper generalization. Similarly, studies on co-temporal reasoning Su et al. (2024) reveal significant gaps in how LLMs handle concurrent or overlapping temporal events, further showing the inconsistency of their internal representations. For example, the Othello-GPT model Li et al. (2023) predicts legal moves and reconstructs game board states from sequence data which shows how LLMs can infer hidden states purely from sequential patterns. However, studies by Toshniwal et al. (2022) and Vafa et al. (2024) demonstrate that such models often fail to recover the full compositional structure of their domains. This limitation is particularly evident in navigation and deterministic finite automata tasks, where models exhibit brittleness under dynamic or adversarial conditions. In games like Othello or navigation tasks, while these models excel in next-token prediction, they fail to construct coherent and generalizable internal representations, reducing their reliability in downstream applications. The key motivating question becomes: How do GPT models construct their world models during training? Beyond flawed representations, LLMs often exhibit peculiar behaviors. For instance, Vafa et al. (2024) describes how models trained on synthetic data, such as random walks, sometimes build better world models than those trained on real-world data. This anomaly highlights how training conditions and evaluation methods heavily influence the quality of implicit representations. Understanding these implicit models requires dissecting the features learned during training and analyzing their role in shaping internal representations.

In this paper, we address these gaps using OthelloGPT as an interpretability testbed. By tracing the features learned at each layer of OthelloGPT, we aim to uncover how GPT-based models construct their world models during training. Our main contributions include:

- **Comparison of Interpretability Methods:** We compare Sparse Autoencoders (SAEs) and linear probes to analyze the learned representations. Our experiments show that SAEs uncover more distinctive and disentangled features, particularly for compositional attributes, whereas linear probes primarily identify features that act as strong correlators to classification accuracy.
- **Layer-wise Feature Analysis:** We uncover a hierarchical progression in OthelloGPT's learned features, where some layers encode general attributes like board shape and edges, while others shift to potentially capturing more dynamic aspects of gameplay, such as tile flips and changing board states towards the center tiles.

1.1 Impact on Agents

LLM-based agents depend heavily on their internal world models to infer the latent structures necessary for tasks like compositional reasoning and long-term planning. For instance, Rothkopf et al. (2024) highlights that agents require consistent internal representations to maintain procedural adherence and interpretability over extended interactions. However, studies like Lopez Latouche et al. (2023) demonstrate that LLMs often fail in long-term planning, leading to behavior that becomes inconsistent with prior states, as observed in video game character dialogues where maintaining style and narrative coherence is crucial. Similarly, Binz & Schulz (2023) show that LLMs struggle with compositional reasoning, failing to synthesize structured responses in complex scenarios like causal inference tasks where even minor perturbations cause substantial deviations from correct or human-like reasoning. These limitations emphasize the importance of coherent and generalizable world models that go beyond next-token prediction accuracy, which remains a primary metric in most evaluations. Agent interpretability can be examined in multiple ways: 1. Point-in-time interpretability: Focuses on how individual tokens in a prompt or response influence an agent's immediate decisions. 2. Procedural interpretability: Analyze how sequences of inputs shape an agent's long-term behavior. 3. Mechanistic interpretability: Investigates the underlying mechanisms and representations within the model that relate to agents' outputs. In our work, we focus on mechanistic interpretability as a fundamental approach to understanding how internal model structures translate into observable agent behaviors. Othello serves as an excellent testbed for studying neural networks due to its compositional reasoning requirements and layered decision-making processes. By using Othello-GPT, we investigate the features the model learns layer-by-layer to understand how Transformers Vaswani (2017) and GPT-like models Brown (2020) construct their implicit world models. While this research aims to expand to LLMs, SLMs like Othello-GPT are valid test beds due to their similar architecture and lower computational cost. By studying the model behavior and interpretability techniques evaluation like linear probes and SAEs on these smaller models, we can devise frameworks transferable to larger models.

2 Related Works

2.1 Sparse Autoencoders (SAEs)

Sparse Autoencoders (SAEs) offer a way forward by disentangling learned features into sparse, interpretable bases Bricken et al. (2023). SAEs transform high-dimensional neural activations into sparse, interpretable representations by minimizing reconstruction error. The objective function for an SAE is:

$$L(x, \hat{x}) = \|x - \hat{x}\|^2 + \lambda \|h\|_1 \tag{1}$$

Here, x represents the input, $\hat{x}$ the reconstructed input, h the latent encoding, and $\lambda \|h\|_1$ enforces sparsity by penalizing the $L1$ norm of h. Unlike linear probes, which fit a classifier $z = Wh + b$ to activations h and identify features accessible via linear separability, SAEs uncover more granular and disentangled representations which can be particularly useful for compositional or overlapping features. By penalizing activation magnitude, SAEs force the model to distribute its representations sparsely across the basis vectors which can make features more interpretable.

More recent studies Quaisley (2024), Aizi (2024) and Girit & Rezaei (2023) applied SAEs to Othello-GPT and demonstrated their ability to disentangle sparse features linked to board states and moves. However, prior work did not analyze features layer by layer or compare SAEs directly to linear probes across layers.

2.2 Linear Probes

Linear probes are a widely used tool for analyzing neural networks(Alain (2016), Tenney (2019) and Belinkov (2021)). Linear probes effectively measure feature accessibility but do not disentangle overlapping or compositional features. Prior studies on OthelloGPT (Hazineh et al. (2023)) revealed that deeper layers encode increasingly accurate board representations as they are linearly accessible for tasks like board state classification. However, these studies did not compare the interpretability of features learned via linear probes with those disentangled by SAEs.

The original Othello-GPT model was trained to predict legal moves from sequential game data and could reconstruct board states using nonlinear probes (Li et al. (2023)). Neel Nanda later showed that a linear probe could identify a "world representation" of the board state, shifting from traditional representations (e.g., "black's turn" vs. "white's turn") to a model-derived interpretation ("my turn" vs. "their turn") (Nanda (2024)). While effective at the classification task, linear probes do not disentangle features or capture compositional patterns, limiting their interpretability.

2.3 Layer by Layer Analysis

Yoshua Bengio's work on intermediate layers (Alain (2016)) introduced the use of linear classifier probes to measure the representations encoded at each layer of a neural network. This analysis revealed that deeper layers progressively improve the linear separability of features as representations become more abstract and aligned with the model's task objective. Subsequent research on GPT models (Tenney (2019), Belinkov (2021)) extended these insights to transformers. For instance, studies on BERT showed that earlier layers encode syntactic information, while later layers capture semantic roles like coreference and entity types. This revealed a progression of linguistic abstraction across layers. However, probing classifiers, particularly linear probes, often struggle to disentangle overlapping or compositional features within the model's activations.

Our work extends Bengio's findings by comparing the features detected by SAEs to those identified by linear probes, revealing how disentangled representations evolve with depth. Inspired by studies on concept depth (Rothkopf et al. (2024), Jin et al. (2024)), we explore how abstract and compositional features emerge layer by layer. By bridging interpretability with functionality, our approach provides a nuanced perspective on how OthelloGPT's and generally LLMs internal world model supports strategic decision-making.

3 METHOD

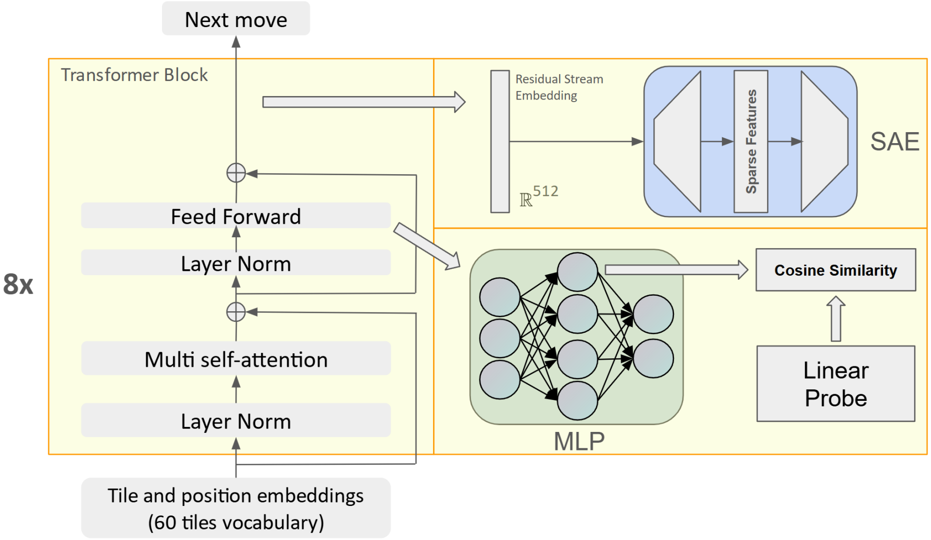


Figure 1: **Our work is divided into three parts.** The left side of the figure illustrates the architecture of OthelloGPT, designed to predict the next legal move in the game of Othello. The upper-right section shows how the Residual Stream from OthelloGPT is used as input to a SAE, enabling feature analysis through its sparse representations. The lower-right section presents a cosine similarity analysis between the parameters of individual neurons in the MLP layers of OthelloGPT and the linear probes we trained.

3.1 OTHELLOGPT

We trained an 8-layer, decoder-only transformer model, Othello-GPT (as shown in Figure 1), with an 8-head attention mechanism and a residual stream dimension $d = 512$, just like Aizi (2024). The model predicts the next token in random Othello game transcripts, treating games as sequences tokenized with a 66-word vocabulary (representing 64 board tiles, with a padding token and an end-of-sequence token). The OthelloGPT was trained on the Synthetic dataset mentioned in Li et al. (2023). It contains out-of-distribution steps that are legal but sub-optimal, which conveys that our OthelloGPT training process has no long-term strategy involved. Formally, let the sequence of tokens for a game be $\mathbf{x} = (x_1, x_2, \ldots, x_T)$, where x_t is the token at timestep t. The model is trained to minimize the autoregressive next-token prediction loss: $\mathcal{L}_{\text{token}} = -\frac{1}{T}\sum_{t=1}^{T} \log p(x_t \mid x_{<t})$, where $p(x_t \mid x_{<t}) = \text{softmax}(\mathbf{h}_t \mathbf{W}_{\text{output}})$, $\mathbf{h}_t$ is the hidden state of the model at timestep t, and $\mathbf{W}_{\text{output}}$ is the output projection matrix. Additionally, for our Sparse Autoencoder (SAE) experiments, we extracted residual stream embeddings from the intermediate layers of OthelloGPT and used these embeddings as inputs to the SAE.

3.2 LINEAR PROBE

We trained linear probes across all layers of the 8-layer. Linear probes aim to extract specific semantic features or predict attributes from the model's hidden states. Let the hidden states at layer

l of a decoder-only Transformer be $\mathbf{H}^{(l)} = [\mathbf{h}_1^{(l)}, \mathbf{h}_2^{(l)}, \ldots, \mathbf{h}_n^{(l)}] \in \mathbb{R}^{n \times d}$, where $\mathbf{h}_i^{(l)} \in \mathbb{R}^d$ is the hidden representation for the i-th token, and n is the sequence length. The linear probe is defined as a classifier: $g_\phi(\mathbf{h}) = \mathbf{W}^T\mathbf{h}$, where $\phi = \mathbf{W} \in \mathbb{R}^{d \times k}$ are the learnable parameters of the probe, and k is the number of target classes. In our case, k is 64 (similar to Nanda (2024)), which represents all the locations on the board. We also have trained three different probes under three modes respectively: empty, which means there are no game pieces on that location on the board; own, which means the game piece is my color on the board on that location; and enemy, which means the game piece is the opponent's color on the board on that location. For a given hidden state $\mathbf{h}$, the predicted probability distribution over the classes is: $\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}^T\mathbf{h})$, and the predicted class is the one with the highest probability: $\hat{y} = \arg\max_j \hat{y}_j$. The linear probe minimizes the cross-entropy loss function as follows:

$$\mathcal{L}(\phi) = \frac{1}{N}\sum_{i=1}^{N} \ell_{CE}(g_\phi(\mathbf{h}_i), y_i) \tag{2}$$

where N is the number of samples, $\mathbf{h}_i = f_\theta(\mathbf{x}_i)$ is the frozen hidden representation extracted by the Transformer f_θ, y_i is the corresponding target label, and the cross-entropy loss is defined as $\ell_{\text{CE}}(\hat{\mathbf{y}}, y) = -\log \hat{y}_y$, where $\hat{y}_y$ is the predicted probability for the correct class y. By training linear probes on hidden states from all layers, we evaluate the layer-wise encoding of semantic information in OthelloGPT.

3.3 Tile color

We devised two methods, using linear probe and SAE, to analyze tile color and discover the robustness of features learned by OthelloGPT.

3.3.1 Linear Probe and Cosine Similarity

To analyze the internal behavior of GPT, we employ the cosine similarity method for network analysis. Given two distinct feature $\mathbf{a}$ and $\mathbf{b}$, the cosine similarity function is: $similarity(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{ab}}{\|\mathbf{a}\|\|\mathbf{b}\|}$. We focus on analyzing individual neurons in every layer. With a pre-trained linear probe for three modes, we calculate the cosine similarity between the MLP neurons and the probe, assessing each neuron's contribution to classification for specific tiles. We present findings primarily from the Encoding layer after observing similar results on the Encoding and Projection layers, where we compute the cosine similarity between MLP parameters and the layer-specific linear probe to quantify each neuron's contribution to encoding tile color. Neurons exceeding a similarity threshold of 0.2 are counted for each tile, focusing on the "my color" probe.

3.3.2 SAE

To ensure the robustness of the features learned by Othello-GPT, we compared the top-performing features across 10 random initialization seeds of the SAEs. We extract sparse features from the residual stream embedding with shape $\mathbf{R}^{512}$ after feed forward in each transformer block. This validation ensures that the model is learning tile colors rather than artifacts from specific random states. For each seed, we identify the top 50 features with AUROC > 0.7, which measure a feature's ability to classify tile states (empty, player's piece, or opponent's piece) by balancing the true positive and false positive rates. A higher AUROC value indicates stronger discriminative power. Aggregating results across all seeds, we tally the frequency of each board position appearing among these top features, as visualized in Figure 3.

3.4 Tile Stability

A tile is defined as stable if it cannot be flipped for the remainder of the game. For instance, corner tiles are inherently stable once placed. Stable tiles include corner tiles, edge tiles anchored to stable tiles, and interior tiles surrounded by stable tiles. Given a set of 104,000 board states (2,000 games x 52 board states per game), we computed binary stability maps for each state. Each board state consisted of 64 tiles, each encoded by its color (0 for empty, 1 for black, 2 for white). Occupied tiles (indicated by 1 or 2), were marked as stable if it was 1) a corner tile ((0,0), (0,7), (7,0), or (7,7)) an

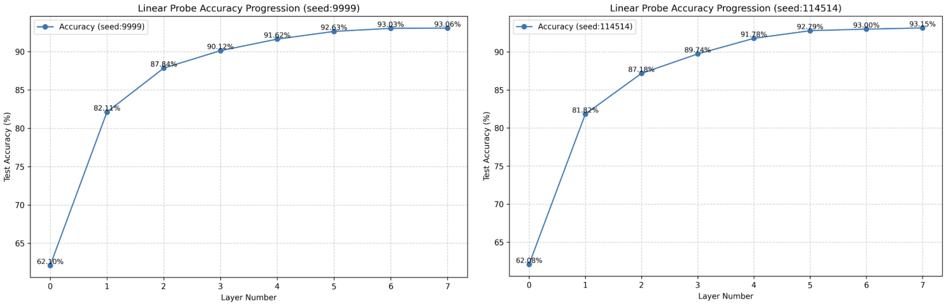


Figure 2: Linear probe accuracy for two seeds. The results demonstrate that linear probes effectively capture features that are good predictors of classification accuracy which increases over layers.

edge tile directly adjacent to a stable tile or 3) an interior tile surrounded by 8 stable tiles (including top, bottom, left, right, and diagonally adjacent neighbors). This process yielded a stability map for each board state, where each tile was assigned a binary value: 1 if stable and 0 otherwise.

Stability Feature Activations. We analyzed feature activations across all 8 layers of our Othello-GPT model with a binary classification framework. Each board state was paired with its corresponding stability map to evaluate whether individual features reliably predicted tile stability. Features were considered active if their activation strength was > 0. For each feature and tile, we computed: true positives (active feature and stable tile), false positives (active feature, non-stable tile), true negatives (inactive feature, non-stable tile), and false negatives (inactive feature, stable tile). From these values, we calculated the F1-score and AUROC using standard metrics.

Feature Analysis using AUROC Thresholds. To determine whether dominant features consistently encode stability across layers, we analyze feature activations with AUROC scores > 0.8. We did the following analysis for each layer: 1. Tile-level: For each tile, we computed the frequency of feature activations with AUROC scores exceeding the threshold. 2. Feature-level: For each feature, we computed the frequency of its activations exceeding the threshold for that layer without regard to specific tile positions. We repeated this analysis for 2 different seeds to confirm our results.

4 Experiments

4.1 Comparing SAEs vs Linear Probes

We show that linear probe accuracy increases across layers (Figure 2), suggesting the model learns stronger predictors for classification tasks. However, they fail to reveal distinct or compositional features per layer. SAEs address this by disentangling activations into sparse, interpretable bases, providing deeper insights into the features learned at each layer.

4.2 Tile Color Across Layers

Figure 3 shows the features extracted by the SAEs, computed by identifying the most discriminative features using AUROC scores across multiple random seeds. The resulting heatmaps highlight positions with consistently high importance, such as edges and central tiles. In contrast, Figure 4 visualizes the contributions of individual MLP neurons from linear probes to tile classifications, measured via cosine similarity.

Figure 3 and Figure 4 reveal distinct differences in how SAEs and linear probes learn features from the board. The SAE visualizations highlight clear and structured patterns, such as strong activations at corner and edge tiles in layer 1, indicating that the model captures the board's shape early on. As we progress to Layers 2 and 4, SAEs show more dynamic changes, with activations concentrated in the central tiles and along the edges. This suggests the SAEs are not only learning positional importance but could also capturing the evolving dynamics of central tiles, which tend to flip frequently as the game progresses. Importantly, these results are aggregated across 10 random seeds, demonstrating the robustness and consistency of SAEs in identifying meaningful features. In contrast, the linear probe visualizations show more dispersed activations across the board. While individual tiles are well-classified, the activations lack the clear structural patterns seen in SAEs.

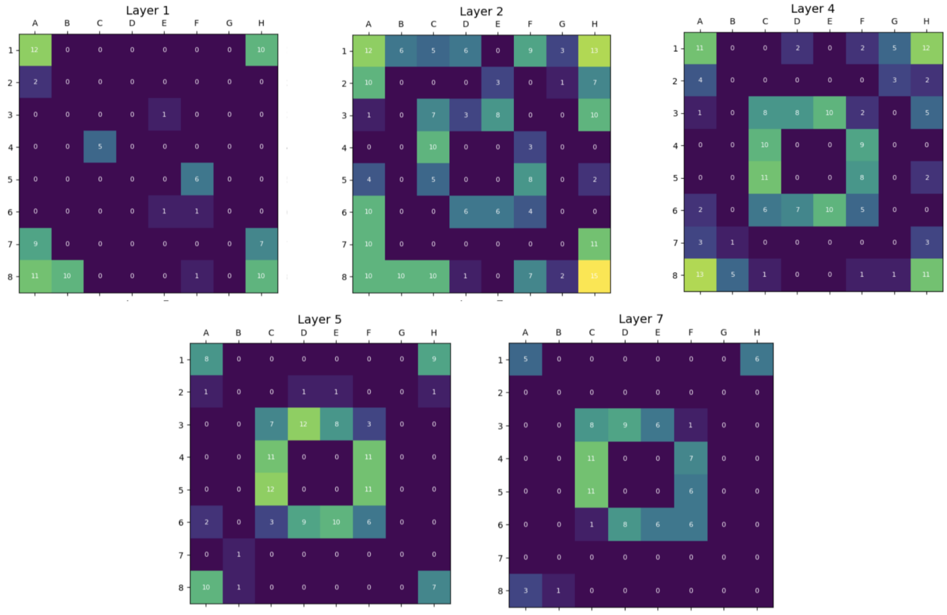


Figure 3: **SAE Tile color activation maps.** Showing frequency of tile color activations measured across 10 different seeds, as described in Section 3.3.

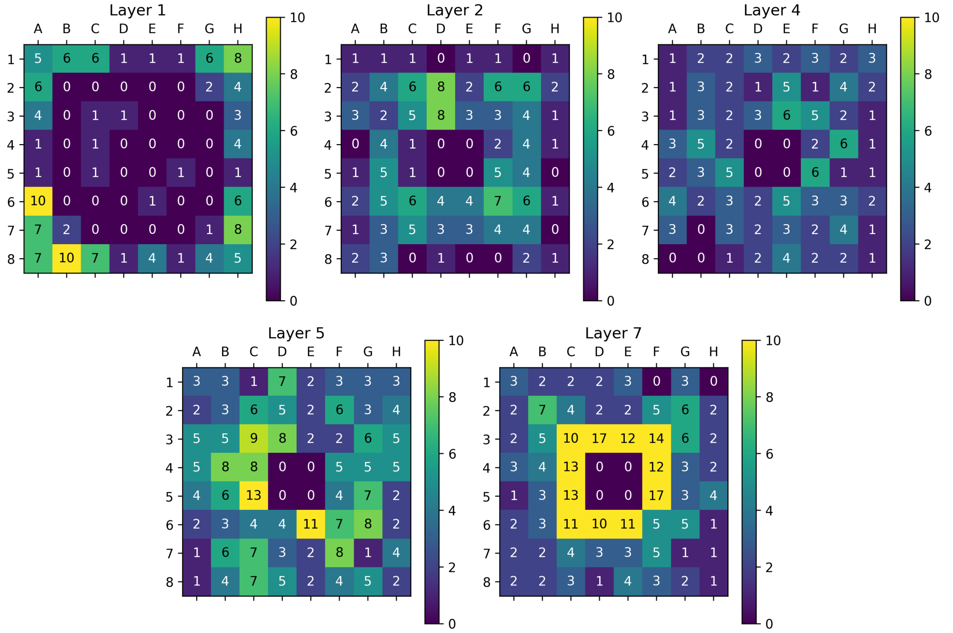


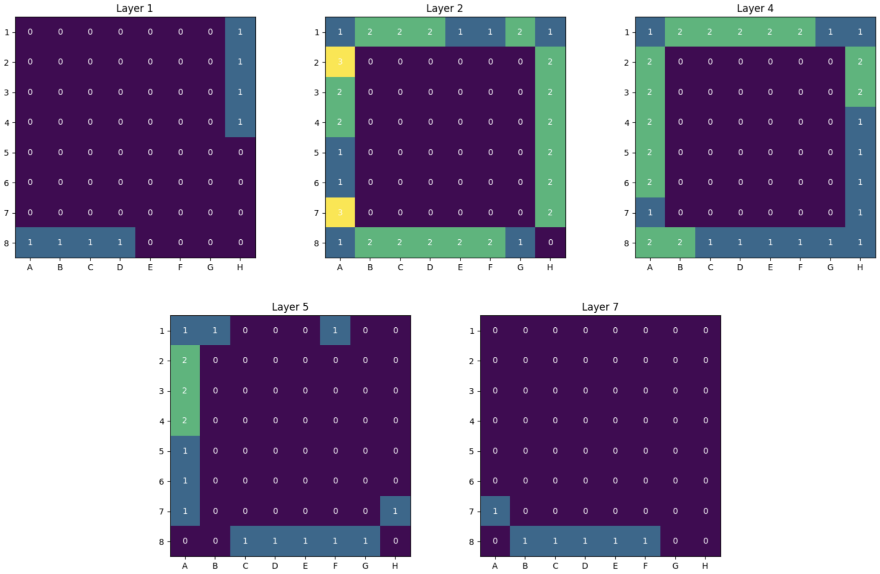
Figure 4: **Linear Probe tile color activation maps.** Showing the tile color activations measured across layers, as described in Section 3.3.

4.3 Tile Stability Across Layers

We notice the highest frequency of tile-feature activations in the intermediate layers (layers 2 through 4), which can be seen in Figure 5. Earlier (layer 1) and later layers (layers 5 through 8) do not appear to learn stability, but rather they likely dedicate their representational capacity to other aspects of the board state. This layer-specific pattern is consistent with the notion that different depths in the model are dedicated to learning different concepts.

We further support this analysis through Table 1, which reveal this same pattern across layers for distinct features. Features 349 and 108 for instance, show a strong pattern which suggests that they encode tile stability. Table 4 shows the exact AUROC scores for tile-feature pairs in layer 2, which are clearly higher relative to other layers. We're able to disentangle and trace these feature patterns across layers through using SAEs, which is an advantage over using linear probes.

However, we must acknowledge that there is some variability across seeds as we can see in Table 1 and Table 2. This variability raises the possibility that the features we interpret as "stability" may be a composition of related but more granular features, such as the presence of edge or corner tile configurations. These properties may jointly give rise to the notion of stability, without representing stability itself in isolation. Future work with additional seeds is necessary to fully investigate these distinctions.



Layer 1

	A	B	C	D	E	F	G	H
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	1	1	1	1	0	0	0	0

Layer 2

	A	B	C	D	E	F	G	H
1	1	2	2	2	1	1	2	1
2	3	0	0	0	0	0	0	2
3	2	0	0	0	0	0	0	2
4	2	0	0	0	0	0	0	2
5	1	0	0	0	0	0	0	2
6	1	0	0	0	0	0	0	2
7	3	0	0	0	0	0	0	2
8	1	2	2	2	2	2	1	0

Layer 4

	A	B	C	D	E	F	G	H
1	1	2	2	2	2	2	1	1
2	2	0	0	0	0	0	0	2
3	2	0	0	0	0	0	0	2
4	2	0	0	0	0	0	0	1
5	2	0	0	0	0	0	0	1
6	2	0	0	0	0	0	0	1
7	1	0	0	0	0	0	0	1
8	2	2	1	1	1	1	1	1

Layer 5

	A	B	C	D	E	F	G	H
1	1	1	0	0	0	1	0	0
2	2	0	0	0	0	0	0	0
3	2	0	0	0	0	0	0	0
4	2	0	0	0	0	0	0	0
5	1	0	0	0	0	0	0	0
6	1	0	0	0	0	0	0	0
7	1	0	0	0	0	0	0	1
8	0	0	1	1	1	1	1	0

Layer 7

	A	B	C	D	E	F	G	H
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0
8	0	1	1	1	1	1	0	0

Figure 5: **Stability activation maps.** Computed for seed 1, as described in Section 3.4. We provide results for another OthelloGPT model in Appendix Figure 1 which shows a similar trend.

5 Conclusion

In this work, we analyzed the progression of learned features in OthelloGPT, uncovering a hierarchical structure where earlier layers capture general attributes like board edges and shape, while deeper layers focus on dynamic gameplay aspects, such as tile flips and evolving board states. By comparing SAEs and linear probes, we demonstrated that SAEs uncover more distinctive and disentangled features, particularly for compositional attributes, while linear probes primarily identify strong correlators for classification tasks. Through key experiments, we decoded features related to tile color and stability, establishing a framework for understanding how GPT-based models and transformers construct internal representations. Future work will use attribution analysis and automated interpretability methods Bills et al. (2023) to identify causal features influencing moves and better understand neuron-level representations.

REFERENCES

Robert Aizi. Research report: Sparse autoencoders find only 9/180 board state features in othellogpt, 2024. URL https://www.lesswrong.com/posts/BduCMgmjJnCtc7jKc/research-report-sparse-autoencoders-find-only-9-180-board.

Guillaume Alain. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances, 2021. URL https://arxiv.org/abs/2102.12452.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. *URL https://openaipublic. blob. core. windows. net/neuron-explainer/paper/index. html.(Date accessed: 14.05. 2023)*, 2, 2023.

Marcel Binz and Eric Schulz. Using cognitive psychology to understand gpt-3. *Proceedings of the National Academy of Sciences*, 120(6):e2218523120, 2023. doi: 10.1073/pnas.2218523120. URL https://www.pnas.org/doi/abs/10.1073/pnas.2218523120.

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

Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.*, 15(3), March 2024. ISSN 2157-6904. doi: 10.1145/3641289. URL https://doi.org/10.1145/3641289.

Uzay Girit and Tara Rezaei. Studying the benefits and limitations of sparse auto-encoders for compositional reasoning tasks, December 2023. URL https://deep-learning-mit.github.io/staging/blog/2023/sparse-autoencoders-for-othello/.

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

Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, et al. Exploring concept depth: How large language models acquire knowledge at different layers? *arXiv preprint arXiv:2404.07066*, 2024.

Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=DeG07_TcZvT.

Gaetan Lopez Latouche, Laurence Marcotte, and Ben Swanson. Generating video game scripts with style. In Yun-Nung Chen and Abhinav Rastogi (eds.), *Proceedings of the 5th Workshop on NLP for Conversational AI (NLP4ConvAI 2023)*, pp. 129–139, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.nlp4convai-1.11. URL https://aclanthology.org/2023.nlp4convai-1.11.

Neel Nanda. Actually, othello-gpt has a linear emergent world representation, 2024. URL https://www.neelnanda.io/mechanistic-interpretability/othello.

Andrew Quaisley. Research report: Alternative sparsity methods for sparse autoencoders with othellogpt, June 2024. URL https://www.lesswrong.com/posts/ignCBxbqWWPYCdCCx/research-report-alternative-sparsity-methods-for-sparse.

Raven Rothkopf, Hannah Tongxin Zeng, and Mark Santolucito. Procedural adherence and interpretability through neuro-symbolic generative agents. 2024. URL https://arxiv.org/abs/2402.16905.

Zhaochen Su, Juntao Li, Jun Zhang, Tong Zhu, Xiaoye Qu, Pan Zhou, Yan Bowen, Yu Cheng, et al. Living in the moment: Can large language models grasp co-temporal reasoning? *arXiv preprint arXiv:2406.09072*, 2024.

I Tenney. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 2019.

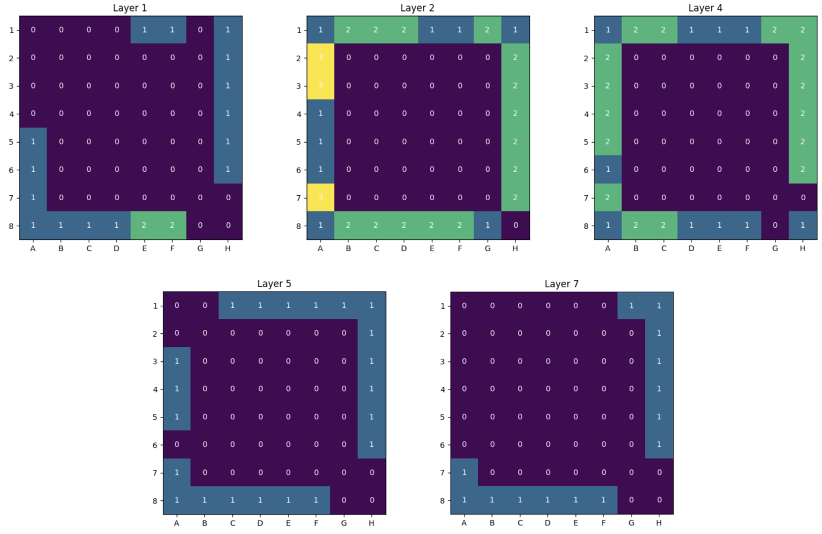
Shubham Toshniwal, Sam Wiseman, Karen Livescu, and Kevin Gimpel. Chess as a testbed for language model state tracking. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36 (10):11385–11393, Jun. 2022. doi: 10.1609/aaai.v36i10.21390. URL https://ojs.aaai.org/index.php/AAAI/article/view/21390.

Keyon Vafa, Justin Y Chen, Ashesh Rambachan, Jon Kleinberg, and Sendhil Mullainathan. Evaluating the world model implicit in a generative model. In *Neural Information Processing Systems*, 2024.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

A Appendix

A.1 Features Activated for Stability



Layer 1

	A	B	C	D	E	F	G	H
1	0	0	0	0	1	1	0	1
2	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	1
5	1	0	0	0	0	0	0	1
6	1	0	0	0	0	0	0	1
7	1	0	0	0	0	0	0	0
8	1	1	1	1	2	2	0	0

Layer 2

	A	B	C	D	E	F	G	H
1	1	2	2	2	1	1	2	1
2	3	0	0	0	0	0	0	2
3	3	0	0	0	0	0	0	2
4	1	0	0	0	0	0	0	2
5	1	0	0	0	0	0	0	2
6	1	0	0	0	0	0	0	2
7	3	0	0	0	0	0	0	2
8	1	2	2	2	2	2	1	0

Layer 4

	A	B	C	D	E	F	G	H
1	1	2	2	1	1	1	2	2
2	2	0	0	0	0	0	0	2
3	2	0	0	0	0	0	0	2
4	2	0	0	0	0	0	0	2
5	2	0	0	0	0	0	0	2
6	1	0	0	0	0	0	0	2
7	2	0	0	0	0	0	0	0
8	1	2	2	1	1	1	0	1

Layer 5

	A	B	C	D	E	F	G	H
1	0	0	1	1	1	1	1	1
2	0	0	0	0	0	0	0	1
3	1	0	0	0	0	0	0	1
4	1	0	0	0	0	0	0	1
5	1	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	1
7	1	0	0	0	0	0	0	0
8	1	1	1	1	1	1	0	0

Layer 7

	A	B	C	D	E	F	G	H
1	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	1
7	1	0	0	0	0	0	0	0
8	1	1	1	1	1	1	0	0

Figure 1: Tile Stability activation map for seed 2.

Feature	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7	Layer 8	Total Count
Feature 349	0	0	0	26	13	9	0	0	48
Feature 108	0	24	23	0	0	0	0	0	47
Feature 687	0	0	0	0	0	5	6	6	17
Feature 64	0	7	9	0	0	0	0	0	16
Feature 947	0	0	0	11	5	0	0	0	16
Feature 850	0	7	7	0	0	0	0	0	14
Feature 917	0	7	0	0	0	0	0	0	7
Feature 121	0	0	0	7	0	0	0	0	7
Feature 214	4	0	0	0	0	0	0	0	4
Feature 311	4	0	0	0	0	0	0	0	4
Feature 629	0	0	3	0	0	0	0	0	3
Feature 706	0	0	0	3	0	0	0	0	3
Feature 689	0	0	0	0	0	2	0	0	2
Feature 921	0	0	1	0	0	0	0	0	1
Feature 20	0	0	1	0	0	0	0	0	1
Feature 423	0	0	1	0	0	0	0	0	1
Feature 385	0	0	1	0	0	0	0	0	1
Feature 690	0	0	0	0	0	0	0	0	1
Feature 691	0	0	0	0	0	0	0	0	1
Feature 678	0	0	0	0	0	0	0	0	1

Table 1: Features activated per layer for stability (seed 1)

Feature	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7	Layer 8	Total Count
Feature 90	0	0	11	20	14	0	0	0	45
Feature 67	0	0	0	7	7	7	7	6	34
Feature 403	0	24	0	0	0	0	0	0	24
Feature 727	0	7	7	7	0	0	0	0	21
Feature 369	0	0	0	0	5	7	7	7	19
Feature 50	0	0	0	7	0	0	0	0	14
Feature 629	8	0	0	0	0	0	0	0	8
Feature 76	7	0	0	0	0	0	0	0	7
Feature 1016	0	0	7	0	0	0	0	0	7
Feature 373	0	0	6	0	0	0	0	0	6
Feature 412	4	0	0	0	0	0	0	0	4
Feature 130	0	2	0	0	0	0	0	0	2
Feature 196	0	0	0	1	0	0	0	0	1
Feature 233	0	0	0	1	0	0	0	0	1
Feature 158	0	0	0	1	0	0	0	0	1
Feature 496	0	0	1	0	0	0	0	0	1
Feature 647	0	0	0	0	0	0	0	1	1
Feature 270	0	0	0	0	0	0	0	1	1
Feature 263	0	0	0	0	0	0	0	1	1
Feature 514	0	0	0	0	0	0	0	1	1

Table 2: Features activated per layer for stability (seed 2)

Feature	Tile Number	AUROC
214	7	0.8814
311	56	0.8731
214	15	0.8412
311	57	0.8344
214	23	0.8246
311	58	0.8174
214	31	0.8105
311	59	0.8020

Table 3: Top Feature-Tile AUROC Scores (Layer 1, Seed 1)

Feature	Tile Number	AUROC
850	56	0.9504
917	7	0.9476
917	15	0.9074
850	57	0.9052
64	0	0.8936
917	23	0.8907
850	58	0.8859
917	31	0.8773
850	59	0.8696
917	39	0.8642
108	61	0.8595
917	6	0.8592
385	55	0.8584
108	40	0.8569
850	60	0.8561
850	48	0.8551
108	32	0.8545
108	60	0.8535
108	5	0.8529
917	47	0.8506

Table 4: Top Feature-Tile AUROC Scores (Layer 2, Seed 1)

Feature	Tile Number	AUROC
947	0	0.8999
947	8	0.8767
947	1	0.8682
947	16	0.8568
349	55	0.8510
349	48	0.8446
349	61	0.8444
349	32	0.8443
349	59	0.8437
947	2	0.8437
947	24	0.8435
706	7	0.8427
349	60	0.8419
349	40	0.8416
349	62	0.8354
349	24	0.8347
947	3	0.8307
349	58	0.8303
349	5	0.8300
349	47	0.8296

Table 5: Top Feature-Tile AUROC Scores (Layer 4, Seed 1)

Feature	**Tile Number**	**AUROC**
349	32	0.8532
349	40	0.8524
947	0	0.8483
349	61	0.8368
947	8	0.8367
349	48	0.8351
349	24	0.8332
349	59	0.8291
349	60	0.8288
349	16	0.8233
947	16	0.8227
947	1	0.8189
947	24	0.8094
349	55	0.8067
349	58	0.8056
349	62	0.8054
349	8	0.8047
349	5	0.8042

Table 6: Top Feature-Tile AUROC Scores (Layer 5, Seed 1)

Feature	**Tile Number**	**AUROC**
687	60	0.8495
687	59	0.8488
687	61	0.8439
687	48	0.8438
687	58	0.8366
687	57	0.8247

Table 7: Top Feature-Tile AUROC Scores (Layer 7, Seed 1)