Emergent Linear Representations in World Models of Self-Supervised Sequence Models

Neel Nanda[*]
Independent

Andrew Lee[*]
University of Michigan

Martin Wattenberg
Harvard University

Abstract

How do sequence models represent their decision-making process? Prior work suggests that Othello-playing neural network learned nonlinear models of the board state (Li et al., 2023a). In this work, we provide evidence of a closely related *linear* representation of the board. In particular, we show that probing for "my colour" vs. "opponent's colour" may be a simple yet powerful way to interpret the model's internal state. This precise understanding of the internal representations allows us to control the model's behaviour with simple vector arithmetic. Linear representations enable significant interpretability progress, which we demonstrate with further exploration of how the world model is computed.[1]

1 Introduction

How do sequence models represent their decision-making process? Large language models are capable of unprecedented feats, yet largely remain inscrutable black boxes. Yet evidence has accumulated that models extract features – articulable properties of the input[2] – and represent them in its internal activations (Geva et al., 2021; Bau et al., 2020; Gurnee et al., 2023; Belinkov, 2022; Burns et al., 2022; Goh et al., 2021; Elhage et al., 2022a). A key first step for interpreting them is understanding how these features are represented. Mikolov et al. (2013c) introduce the **linear representation hypothesis**: that features are represented linearly as directions in activation space. This would be highly consequential if true, yet this remains controversial and without conclusive empirical justification. In this work, we present novel evidence of
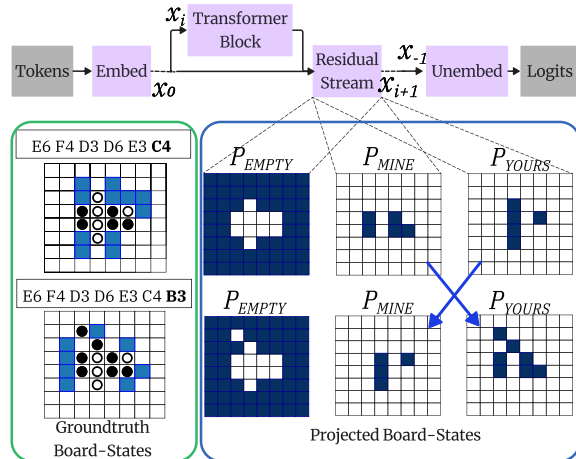

Figure 1: The emergent world models of OthelloGPT are linearly represented. We find that the board states are encoded relative to the current player's colour (MINE vs. YOURS) as opposed to absolute colours (BLACK vs. WHITE).

linear representations, and show that this hypothesis has real predictive power.

We build on the work of Li et al. (2023a), who demonstrate the emergence of a *world model* in sequence models. Namely, the authors train OthelloGPT, an autoregressive transformer model, to predict legal moves in a game of Othello given a sequence of prior moves (Section 2.2). They show that the model spontaneously learns to track the correct board state, recovered using *non-linear* probes, despite never being told that the board exists. They further show a causal relationship between the model's inner board state and its move predictions using model edits. Namely, they show that the edited network plays moves that are legal in the edited board state even if illegal in the original board, and even if the edited board state is unreachable by legal play (i.e., out of distribution).

Critically, the original authors claim that OthelloGPT uses *non-linear* representations to encode the board state, by achieving high accuracy with non-linear probes, but failing to do so using linear

---

[*]Equal contribution. neelnanda27@gmail.com, ajyl@umich.edu

[1]Code available at https://github.com/ajyl/mech_int_othelloGPT

[2]Note that our use of the term refers to a higher-level notion than its more common use in deep learning terminology, i.e., an individual neuron.

probes. In our work, we demonstrate that a closely related world model is actually *linearly* encoded. Our key insight is that rather than encoding the *colours* of the board (BLACK, WHITE, EMPTY), the sequence model encodes the board *relative* to the current player of each timestep (MINE, YOURS, EMPTY). In other words, for odd timesteps, the model considers BLACK tiles as MINE and WHITE tiles as YOURS, and vice versa for even timesteps (Section 3). Using this insight, we demonstrate that a *linear* projection can be learned with near perfect accuracy to derive the board state.

We further demonstrate that we can steer the sequence model's predictions by simply conducting vectoral arithmetics using our linear vectors (Section 4). Put differently, by pushing the model's activations in the directions of MINE, YOURS, or EMPTY, we can alter the model's belief state of the board, and change its predictions accordingly. Our intervention method is much simpler and interpretable than that of Li et al. (2023a), which rely on gradients to update the model's activations (Section 4.1). Our results confirm that our interpretation of each probe direction is correct, but also demonstrates that a mechanistic understanding of model representations can lead to better control. Our results do not contradict that of Li et al. (2023a), but add to our understanding of emergent world models.

We provide additional interpretations of the sequence model using linear operations. For example, we provide empirical evidence of how the model derives empty tiles of the board, and find additional linear representations, such as tiles being FLIPPED at each timestep.

Finally, we provide a short discussion of our thoughts. How should we think of linear versus non-linear representations? Perhaps most interestingly, why do linear representations emerge?

2 Preliminaries

In this section we briefly describe Othello, OthelloGPT, and our notations.

2.1 Othello

Othello is a two player game played on a 8x8 grid. Players take turns playing black or white discs on the board, and the objective is to have the majority of one's coloured discs by the end of the game.

The board always starts with the middle 4 tiles filled with black and white tiles. At each turn, when a tile is played, all of the opponent's discs that are enclosed in a horizontal, vertical, or diagonal row between two discs of the current player are flipped. The game ends when there are no more valid moves for both players.

2.2 OthelloGPT

OthelloGPT is a 8-layer GPT model (Radford et al., 2019), each layer consisting of 8 attention heads and a 512-dimensional hidden space. We use the model weights provided by Li et al. (2023a), denoted there as the synthetic model. The vocabulary space consists of 60 tokens,[3] each one corresponding to a playable move on the board (e.g., A4).

The model is trained in an autoregressive manner, meaning for a given sequence of moves $m_{<t}$, the model must predict the next valid move m_t.

Note that no a priori knowledge of the game nor its rules are provided to the model. Rather, the model is only given move sequences with a training objective to predict next valid moves, by randomly sampling sequences of games from a game tree. This training objective differs from that of models like AlphaZero (Silver et al., 2018), which are trained to play strategic moves to win games.

2.3 Notations

Transformers. Our transformer architecture (Vaswani et al., 2017) consists of embedding and unembedding layers Emb and $Unemb$ with a series of L transformer layers in-between. Each transformer layer l consists of H multi-head attentions and a multilayer perception (MLP) layer.

A forward pass in the model first embeds the input token at timestep t using embedding layer Emb into a high dimensional space $x_t^0 \in \mathbb{R}^D$. We refer to $x_{t \in T}^0$ as the start of the *residual stream*. Then each attention head $Att_l^h, \forall h \in H$ and MLP block at layer l add to the residual stream:

$$x_t^{l\_mid} = x_t^l + \sum_{h \in H} Att_l^h(x_t^l)$$
$$x_t^{l+1} = x_t^{l\_mid} + MLP_l(x_t^{l\_mid})$$

Each attention head Att_l^h computes value vectors by projecting the residual stream to a lower dimension using $Att_l^h.V$, linearly combines value

---

[3]The game always starts with 4 tiles in the center of the board already filled.

	x^0	x^1	x^2	x^3	x^4	x^5	x^6	x^7
Randomized	37	35.1	33.9	35.5	34.8	34.7	34.4	34.5
Probabilistic				61.8				
Linear {BLACK, WHITE, EMPTY}	62.2	74.8	74.9	75.0	75.0	74.9	74.8	74.4
Non-Linear {BLACK, WHITE, EMPTY}	63.4	88.6	93.3	96.3	97.5	98.3	98.7	98.3
Linear {MINE, YOURS, EMPTY}	**90.9**	**94.8**	**97.2**	**98.3**	**99**	**99.4**	**99.6**	**99.5**

Table 1: Probing accuracy for board states. OthelloGPT linearly encodes the board state relative to the current player at each timestep (MINE vs. YOURS, as opposed to colours BLACK or WHITE.

vectors using $Att_l^h.A$, and projects back to the residual stream using $Att_l^h.O$:

$$h(x) = (Attn_l^h.A \otimes Attn_l^h.O \cdot Attn_l^h.V) \cdot x$$

where $\otimes$ notates a tensor product. A final prediction is made by applying $Unemb$ on x^{L-1}, followed by a softmax.

Probe Models. We notate linear and non-linear probes as p^λ and p^ν. Our linear probes are simple linear projections from the residual stream: $p^\lambda(x_t^l) = \text{softmax}(Wx_t^l), W \in \mathbb{R}^{D \times 3}$. The dimension $D \times 3$ comes from doing a 3-way classification.[4] Non-linear probes are 2-layer MLP models: $p^\nu(x_t^l) = \text{softmax}(W_1\text{ReLU}(W_2x_t^l)), W_1 \in \mathbb{R}^{H \times 3}, W_2 \in \mathbb{R}^{D \times H}$. Li et al. (2023a) classify the colour at each tile (BLACK, WHITE, EMPTY). Our insight is to classify the colours *relative* to the current turn's player (MINE, YOURS, EMPTY).

3 Linearly Encoded Board States

In this section we describe our experiments to find linear board state representations.

3.1 Experiment Setup

Rather than encoding the colour of each tile (BLACK, WHITE, EMPTY), OthelloGPT encodes each tile *relative* to the player of each timestep (MINE, YOURS, EMPTY) — for *odd* timesteps, we consider BLACK to be MINE and WHITE to be YOURS, and vice versa for *even* timesteps.

In order to learn the weights of our linear probe, we train on random game sequences until a validation loss on a set of 512 games converges according to a patience value of 10. In practice, our linear probes converge after around 100,000 training samples. We test our probes on a held out set of 1,000 games.
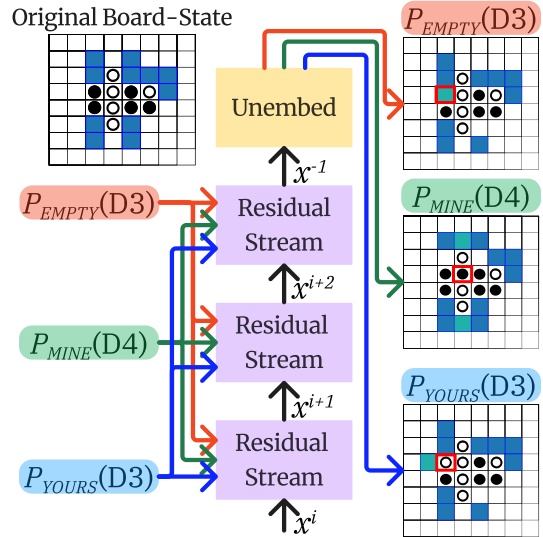

Figure 2: Intervening methodology: we intervene by adding either EMPTY, MINE, or YOURS directions into each layer of the residual stream. Red squares in each board indicate the tiles that have been intervened, teal tiles indicate new legal moves post-intervention that the model predicts.

We train a different probe for each layer l. Hyperparameters are provided in the Appendix.

3.2 Results

Table 1 shows the accuracy for various probes.

We include four baselines. The first is a linear probe trained on a randomly initialized GPT model. We also include a probabilistic baseline, in which we always choose the most likely colour per tile at each timestep, according to a set of 60,000 games from training data. The next two baselines are probe models used in Li et al. (2023a): a linear and non-linear probe trained to classify amongst {BLACK, WHITE, EMPTY}.

Our linear probes achieve high accuracy by layer 4. Unbeknownst previously, we show that the emerged board state is linearly encoded.

---

[4]In practice, because we are predicting the state of all 64 tiles, the shape of our probe is $D \times 64 \times 3$.

18

4 Intervening with Linear Directions

In this section we demonstrate how we intervene on OthelloGPT's board state using linear probes.

4.1 Method

An inherent issue with probing is that it is correlational, not causal (Belinkov, 2022). To validate that our probes have found a true world model, we confirm that the model uses the encoded board state for its predictions.

To verify this, we conduct the same intervention experiment as Li et al. (2023a). Namely, given an input game sequence (and its corresponding board state B), we intervene to make the model believe in an altered board state B'. We then observe whether the model's prediction reflects the made-believe board state B' or the original board state B.

Our intervention approach is simple (Figure 2): we add our linear vectors to the residual stream of each layer:

$$x' \leftarrow x + \alpha p_d^\lambda(x)$$

where d indicates a direction amongst {MINE, YOURS, EMPTY} and α is a scaling factor. In other words, to flip a tile from YOURS to MINE, we simply push the residual stream at every layer in the MINE direction, or to "erase" a previously played tile, we push in the EMPTY direction. [5] [6]

Note that this intervention is much simpler than that of Li et al. (2023a). Namely, Li et al. (2023a) edits the activation space (x) of OthelloGPT using their non-linear probes. More specifically, they use non-linear probes to predict board state B, then compute gradients had the correct board state been the target board state B', and finally use the gradients to update the *activation space* of OthelloGPT rather than the weights of the probe model. Instead, we perform a single vector addition.

4.2 Experiment Setup

For our intervention experiment, we adopt the same setup and metrics as Li et al. (2023a). We use an evaluation benchmark consisting of 1,000 test cases. Each test case consists of a partial game sequence (B) and a targeted board state B'.

Flipping colours	Avg. # Errors
Null Intervention Baseline	2.723
Non-Linear Intervention	0.12
Linear Probe Addition	**0.10**
Erasing	Avg. # Errors
Null Intervention Baseline	2.73
Non-Linear Intervention	0.11
Linear Probe Addition	**0.02**

Table 2: Error rates from interventions. We measure the number of false positives and false negatives in the top-N predictions post-intervention, where N is the number of legal moves in the target board state B'.

We measure the efficacy of our intervention by treating the task as a multi-label classification problem. Namely, we compare the top-N predictions post-intervention against the groundtruth set of legal moves at state B', where N is the number of legal moves at B'. We then compute error rate, or the number of false positives and false negatives.

Li et al. (2023a) only considers the scenario of flipping the colour of a tile. To also validate our EMPTY direction, we also experiment with "erasing" a previously played tile by making it empty.

4.3 Results

Table 2 shows the average error rates after our interventions. A null intervention measures the number of errors by comparing pre-intervention predictions on post-intervention groundtruths. Our interventions are equally effective as that of gradient-based editing (Li et al., 2023a), and confirms that our interpretation of each linear direction matches how the model uses such directions.

5 Additional Linear Interpretations

The linear representation hypothesis is of interest to the mechanistic interpretability community because it provides a foothold into understanding a system. The internal state of the transformer, the residual stream, is the sum of the outputs of all previous components (heads, layers, embeddings and neurons) (Elhage et al., 2021). Albeit the residual stream consisting of linear and non-linear transformations, linear functions of the residual stream allow us to identify where a computation of interest takes place, or trace how a representation of interest evolves over a forward pass.

In this section we leverage our newfound linear representation of board state to provide additional

interpretations of OthelloGPT, as proof of concept of how discovering linear representations unlocks downstream interpretability applications.

5.1 Interpreting Empty Tiles

Here we interpret how OthelloGPT derives the status of empty tiles.

The EMPTY Circuit. A key insight for EMPTY is that input tokens each correspond to a tile on the board (i.e., A4), and once played, the tile can only change colour but remains non-empty.

We view OthelloGPT as using attention heads to "broadcast" which moves have been played: given a move at timestep t, attention heads write this information into other residual streams. This information (PLAYED) can be represented as following. First, each move m (A4) is embedded: $Emb[m]$. Then the model writes this information to other residual streams using linear projections $Att.V$ and $Att.O$ (Section 2.3):

$$\text{PLAYED}_h(m) = Emb[m]@Att_h.V@Att_h.O$$

For each attention head in the first layer,[7] we compute the cosine similarity between PLAYED and the $p_{\text{EMPTY}}^{\lambda}$ direction:

$$\max_{h \in H} \text{CosSim}(\text{PLAYED}_h(m), p_{\text{EMPTY}}^{\lambda}(m))$$

Since the two terms encode *opposite* information, we expect a high negative cosine similarity.

We observe an average similarity score of **-0.862** across all 60 squares,[8], confirming that p_{EMPTY} is encoding NOT PLAYED. This tells us that p_{EMPTY} is a linear function of the token embeddings.

This also implies that OthelloGPT knows which tiles are empty by x^{0_mid}: after the first attention heads but before the MLP layer. On a binary classification task of EMPTY vs. NOT-EMPTY from 1,000 games in our test split, our probe achieves an accuracy of **76.8%** and **98.9%**, when projecting from the residual stream before and after the attention heads from the first layer.
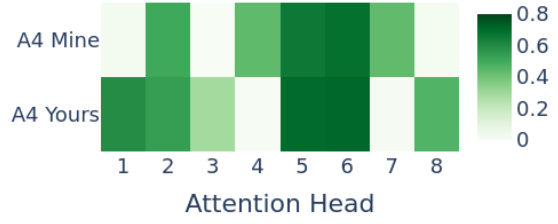

Figure 3: Difference in probability of A4 being empty, between our clean and corrupt sequences, measured in each attention head.
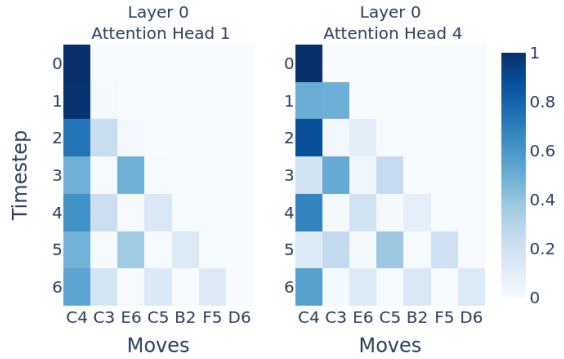

Figure 4: Examples of attention heads from the first layer attending to moves that are YOURS (left) or MINE (right).

Logit Attribute for EMPTY. The previous analysis is based on the *weights* of the model. Here we provide an alternative analysis by studying the *activations* during inference.

First, we select a move m (A4) that we wish to explain. We then construct a "clean" and "corrupt" set of partial game sequences (N=4,569). Our clean set always includes m, while our corrupt set replaces all timesteps with m in the clean set with an alternative move. We ensure that all games in our corrupt set remain legal sequences. Finally, we study the *difference in probability* that m is empty, according to our probes, in our two sets. Namely, we project the outputs from each attention head onto the EMPTY direction and apply a softmax:

$$P_{\text{EMPTY}[m]}(\sigma) = Softmax(\sigma * p_{\text{EMPTY}[m]}^{\lambda})$$

where σ is the output from each attention head.

Figure 3 shows the difference in probability that A4 is empty, between our clean and corrupt inputs, measured in each attention head of the first layer.

---

[7]Knowing which moves were PLAYED (i.e. show up in the input sequence), should not depend on any other computation, and thus we expect this information to be written by the attention heads in the first layer.

[8]The center 4 squares can never be empty.

20

	x^0	x^1	x^2	x^3	x^4	x^5	x^6	x^7
Linear {FLIPPED, NOT-FLIPPED}	74.76	85.75	91.62	94.82	96.44	97.13	96.82	96.3

Table 3: $F1$ score for probing on FLIPPED tiles. In addition to the board state, the model also linearly encodes concepts such as flipped tiles per timestep.

The figure decomposes two scenarios: when A4 was originally played as MINE or YOURS. This is because some attention heads only attend to moves that are MINE (4, 7), while some only attend to YOURS (1, 3, 8), which we show below.

5.2 Attending to MY & YOUR Timesteps

We find that some attention heads only attend to either MY or YOUR moves. Figure 4 shows two examples: at each timestep, each head *alternates* between attending to even or odd timesteps. Such behavior further indicates how the model computes its world model based on MINE and YOURS as opposed to BLACK and WHITE.

5.3 Additional Linear Concepts: FLIPPED

In addition to linearly representing the board state, we find that OthelloGPT also encodes which tiles are being flipped, or captured, at each timestep. To test this, we modify our probing task to classify between FLIPPED vs. NOT-FLIPPED, with the same training setup described above. Given the class imbalance, for this experiment we report $F1$ scores. Table 3 demonstrates high $F1$ scores by layer 3.

We also conduct a modified version of our intervention experiment, in which we always randomly select a flipped tile at the current timestep to intervene on. Then, instead of adding either p^λ_{MINE}, p^λ_{YOURS}, or p^λ_{EMPTY}, we *subtract* $p^\lambda_{\text{FLIPPED}}$. This tests whether the FLIPPED feature is causally relevant for computing the next move, by exploring whether this is sufficient to cause the model to play valid moves in the new board state. We get an average error rate of **0.486**, compared to a null intervention baseline rate of **1.686**.

One can consider FLIPPED tiles as the difference between the previous and current board state. One might naturally think that a recurrent computation could derive the current board state by iteratively applying such differences. However, transformer models do **not** make recursive computations![9] Also, the derivative property of captured tiles being encoded in later layers might be

analogous to observations from previous studies of language models that show low-level lexical properties being encoded in lower layers and syntax and semantics being mostly captured in higher layers (Tenney et al., 2019).

5.4 Multiple Circuits Hypothesis

Although we find board state representations and their causality on move predictions, we find that they do not explain the entire model. Namely, if our understanding is correct, we expect the model to compute the board state before computing valid moves. However, we find that in end games, this is not the case.

To check for the correct board state, we apply our linear probes on each layer, and check the earliest layer in which all 64 tiles are correctly predicted.[10] To check for correct move predictions, we project from each layer using the unembedding layer, and check the earliest layer in which the top-N move predictions are all correct, where N is the number of groundtruth legal moves.

Figure 5 plots the proportion of times the board state is computed before (or after) valid moves (first y-axis). We also overlay the average earliest layer in which board or moves are correctly computed (second y-axis, aqua and lime curves). To our surprise, we find that in end games, the model often computes legal moves *before* the board state (black bars). We henceforth refer to this behavior as MOVEFIRST, and share some thoughts.

End Game Circuits. First, MOVEFIRST starts to occur around move 30, which is the mid-point of the game. Second, MOVEFIRST occurs more frequently as we near the end of the game (increasing black bars). Interestingly, in Othello, starting from the mid-point, there are progressively fewer empty tiles than there are filled tiles as the board fills up. Also note that as the game progresses, it becomes more likely for every empty tile to be a legal move.

One possible explanation for this phenomenon is that in the end game, it may be possible to pre-

---

[9]Doing so would require our transformer model to have the same number of layers as the maximum game sequence length of 60.

[10]It might be the case that legal moves could be predicted without 100% accuracy of the board state. We try variants (see Appendix), but observe similar trends.
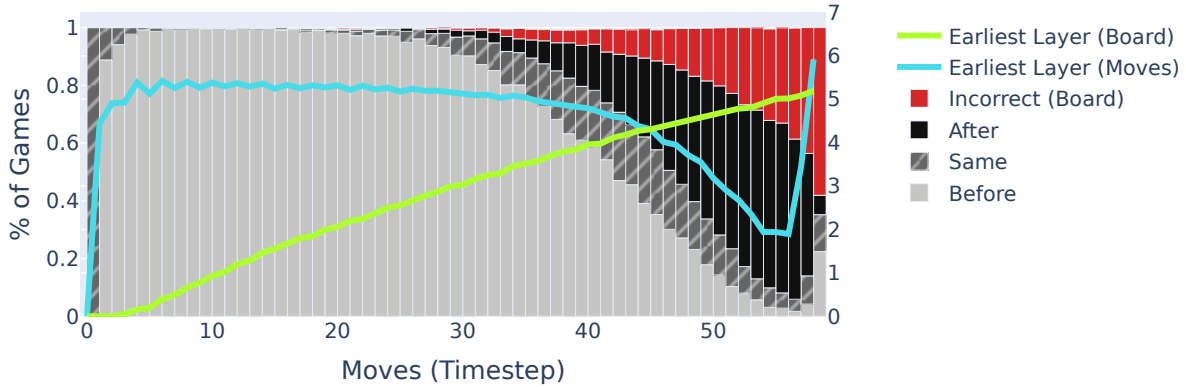
Figure 5: Proportion of times the board state is computed before/after move predictions are made (First y-axis). **Light Grey:** Boards are computed in an earlier layer than moves. **Dark Grey, Black:** Boards are computed in the same or later layer than moves. **Red:** Model never computes the correct board state. **Aqua, Lime (Curves):** Average earliest layer in which the board or moves are correctly computed (Second y-axis). Starting from the midgame, we start observing the model compute moves before boards (black bar), and this occurs more frequently as the game progresses.

dict legal moves with simpler circuits that do not require the entire board state. For instance, perhaps it combines EMPTY with other features such as IS-SURROUNDED-BY-MINE or IS-BORDER and so on.

Multiple Circuits. Interestingly, the model still uses the board state at end games. To demonstrate this, we run our intervention experiment on 1,000 *end games*,[11] and still achieve a low error rate of **0.112**.[12] We thus hypothesize that OthelloGPT (and more broadly, sequence models) consist of multiple circuits. Another hypothesis is that residual networks make "iterative inferences" (Section 5.5), and for end games, OthelloGPT uses simpler circuits in the early layers and refines its predictions at late layers using board state.

End Game Board Accuracy. We observe that board state accuracy drops near end games. This can be seen by the growing red bars, but also by measuring per-timestep accuracy of our probes (see Appendix). It is unclear whether 1) the model does not bother to compute the perfect board state, as alternative circuits allow the model to still correctly predict legal moves, or 2) the model learns an alternative circuit because it struggles to compute the correct board state at end games.

Memorization. Note that in the first few timesteps, the board and legal moves are sometimes both computed in the same layer (dark grey bars). This may be due to memorization: 1) these

predictions both occur at the first layer, and 2) there are only so many openings in an Othello game.

5.5 Iterative Feature Refinements

Figure 6 visualizes OthelloGPT's "iterative inference" (Jastrzebski et al., 2018; Belrose et al., 2023; Veit et al., 2016; nostalgebraist, 2020), or iterative refinement of features. For each layer, we plot the projected board states using our probes, and projected next-move predictions using the unembedding layer. Multiple evidence of iterative refinements are provided in the Appendix.

6 Discussions

6.1 On Linear vs. Non-Linear Interpretations

One challenge with probing is knowing which features to look for.[13] For instance, classifying {BLACK, WHITE} versus {MINE, YOURS} leads to different takeaways, which illustrates the danger of *projecting our preconceptions*. What might seem "sensible" to a human interpreter (BLACK, WHITE) may not be for a model. In hindsight, given the symmetric game-play of Othello, encoding MINE, YOURS is perfectly sensible for the model (For more examples of non-obvious, sensible features, see (McCoy et al., 2019; Nanda et al., 2023)).

More broadly, what is sensible, or alternatively, how we choose to interpret linear or non-linear encodings, can be relative to how we see the world. Suppose we had a perfect world model of our physical world. Further suppose that if and when it

---

[11]We intervene on a timestep > 30
[12]Non-intervention baseline: 1.988.

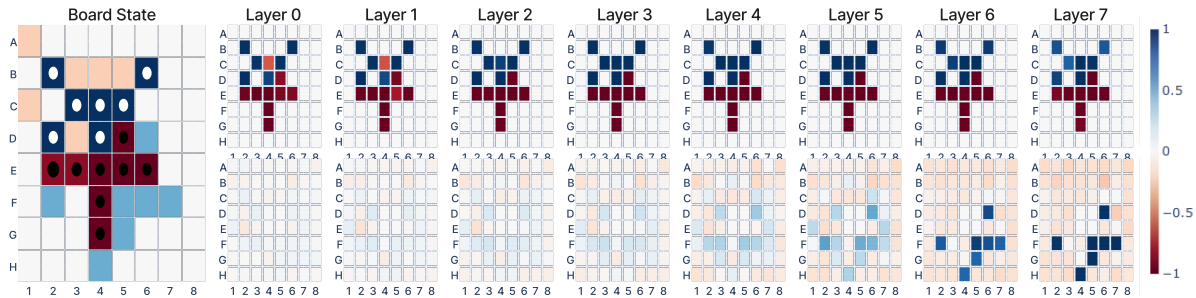[13]For a longer discussion on probing, see Appendix.

Figure 6: Iterative refinements: the top row shows each layer projected using our linear probes. The bottom row shows the model's predictions for legal moves at each layer, by applying the unembedding layer on each layer.

computes a gravitational force between two objects (Newton's law), we discover a neuron whose square root was the distance between two objects. Is this a non-linear representation of distance? Or, given the form of Netwon's law, is the square of the distance a more natural way for the model to represent the feature, and thus considered a linear representation? As this example shows, what constitutes a natural feature may be in the eye of the beholder.

6.2 On the Emergence of Linear Representations

Linear representations in sequence models have been observed before: iGPT (Chen et al., 2020), which was autoregressively trained to predict next pixels of images, lead to robust linear image representations. The question remains, why do linear feature representations emerge? What linear representations are currently encoded in large language models? One reason might be simply that matrix multiplication can easily extract a different subset of linear features for each neuron. However, we leave a complete explanation to future work.

7 Related Work

We discuss three broad related areas: understanding internal representations, interventions, and mechanistic interpretability.

7.1 Understanding Internal Representations

Multiple researchers have studied concept representations in sequence models. Li et al. (2021) train sequence models on a synthetic task, and uncover world models in their activations. Patel and Pavlick (2022) demonstrate that language models can learn to ground concepts (e.g., direction, colour) to real world representations. Burns et al. (2022); Marks and Tegmark (2023) find linear vectors that encode "truthfulness". Probing techniques have also

been used to extract linguistic characteristics in sentence embeddings (Conneau et al., 2018; Tenney et al., 2019). Researchers have also used *structural* probes to uncover syntactic structures in word embeddings (Hewitt and Manning, 2019) and language models (Eisape et al., 2022). Prior to current day language models, word embeddings (Mikolov et al., 2013b,a) built vectoral word representations.

Linear representations are found outside of language models as well. Merullo et al. (2022) demonstrate that image representations from vision models can be linearly projected into the input space of language models. McGrath et al. (2022) and Lovering et al. (2022) find interpretable representations of chess or Hex concepts in AlphaZero.

7.2 Intervening On Language Models

A growing body of work has intervened on language models, by which we mean controlling their behavior by altering their activations.

We consider two broad categories. Parametric approaches often use optimizations (i.e. gradient descent) to locate and alter activations (Li et al., 2023a; Meng et al., 2022a,b; Hernandez et al., 2023; Hase et al., 2023). Meanwhile, inference-time interventions typically apply linear arithmetics, for instance by using "truthful" vectors (Li et al., 2023b), "task vectors" (Ilharco et al., 2022), or other "steering vectors" (Subramani et al., 2022; Turner et al., 2023).

7.3 Mechanistic Interpretability

Mechanistic interpretability (MI) studies neural networks by reverse-engineering their behavior (Olah et al., 2020; Elhage et al., 2021). The goal of MI is to understand the underlying computations and representations of a model, with a broader goal of validating that their behavior aligns with what researchers have intended. Such framework has allowed researchers to better understand grokking

(Nanda et al., 2023), superposition (Elhage et al., 2022b; Scherlis et al., 2022; Arora et al., 2018), or even individual neurons (Mu and Andreas, 2020; Antverg and Belinkov, 2021; Gurnee et al., 2023).

8 Conclusion

In this work we demonstrated that the emergent world model in Othello-playing sequence models is full of linear representations. Previously unbeknownst, we demonstrated that the board state in OthelloGPT is linearly represented by encoding the colour of each tile *relative* to the player at each timestep (MINE, YOURS, EMPTY) as opposed to absolute colour (BLACK, WHITE, EMPTY). We showed that we can accurately control the model's behaviour with simple vector arithmetic on the internal world model. Lastly, we mechanistically interpreted multiple facets of the sequence model, analysing how empty tiles are detected, and linear representations of which pieces are flipped. We find hints that multiple circuits might exist for predicting legal moves in the end game, as well as further evidence that residual networks iteratively refine their features across layers.

9 Acknowledgements

10 Author Contributions

Neel Nanda discovered the linear representation in terms of relative board state, and showed that simple vector arithmetic sufficed for causal interventions. He led an initial version of the experiments and write-ups, and advised throughout.

Andrew Lee led this write-up and performed all experiments in this paper. He discovered the flipped linear representation, the empty circuit, and the multiple circuit hypothesis results.

Martin Wattenberg helped with editing and distilling the paper, and contributed the analogy about a linear vs quadratic representation of distance.

References

Omer Antverg and Yonatan Belinkov. 2021. On the pitfalls of analyzing individual neurons in language models. *arXiv preprint arXiv:2110.07483*.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2018. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495.

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. 2020. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2022. Discovering latent knowledge in language models without supervision. *ArXiV*.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. 2020. Generative pretraining from pixels. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1691–1703. PMLR.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Tiwalayo Eisape, Vineet Gangireddy, Roger Levy, and Yoon Kim. 2022. Probing for incremental parse states in autoregressive language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2801–2813, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Nelson Elhage, Tristan Hume, Catherine Olsson, Neel Nanda, Tom Henighan, Scott Johnston, Sheer ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, Danny Hernandez, Amanda Askell, Kamal Ndousse, Andy Jones, Dawn Drain, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, Zac Hatfield-Dodds, Jackson Kernion, Tom Conerly, Shauna Kravec, Stanislav Fort, Saurav Kadavath, Josh Jacobson, Eli Tran-Johnson, Jared Kaplan, Jack Clark, Tom Brown,

Sam McCandlish, Dario Amodei, and Christopher Olah. 2022a. Softmax linear units. *Transformer Circuits Thread.* Https://transformer-circuits.pub/2022/solu/index.html.

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

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread.* Https://transformer-circuits.pub/2021/framework/index.html.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem Zuidema. 2018. Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 240–248, Brussels, Belgium. Association for Computational Linguistics.

Gabriel Goh, Nick Cammarata †, Chelsea Voss †, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. 2021. Multimodal neurons in artificial neural networks. *Distill.* Https://distill.pub/2021/multimodal-neurons.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610.*

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2023. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *arXiv preprint arXiv:2301.04213.*

Evan Hernandez, Belinda Z Li, and Jacob Andreas. 2023. Measuring and manipulating knowledge representations in language models. *arXiv preprint arXiv:2304.00740.*

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Association for Computational Linguistics.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089.*

Stanisław Jastrzebski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, and Yoshua Bengio. 2018. Residual connections encourage iterative inference. In *International Conference on Learning Representations.*

Belinda Z. Li, Maxwell Nye, and Jacob Andreas. 2021. Implicit representations of meaning in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1813–1827, Online. Association for Computational Linguistics.

Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023a. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations.*

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023b. Inference-time intervention: Eliciting truthful answers from a language model. *arXiv preprint arXiv:2306.03341.*

Charles Lovering, Jessica Forde, George Konidaris, Ellie Pavlick, and Michael Littman. 2022. Evaluation beyond task performance: Analyzing concepts in alphazero in hex. In *Advances in Neural Information Processing Systems*, volume 35, pages 25992–26006. Curran Associates, Inc.

Samuel Marks and Max Tegmark. 2023. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *arXiv preprint arXiv:2310.06824.*

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Thomas McGrath, Andrei Kapishnikov, Nenad Tomašev, Adam Pearce, Martin Wattenberg, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. 2022. Acquisition of chess knowledge in alphazero. *Proceedings of the National Academy of Sciences*, 119(47):e2206625119.

Thomas McGrath, Matthew Rahtz, Janos Kramar, Vladimir Mikulik, and Shane Legg. 2023. The hydra effect: Emergent self-repair in language model computations. *arXiv preprint arXiv:2307.15771*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

Jack Merullo, Louis Castricato, Carsten Eickhoff, and Ellie Pavlick. 2022. Linearly mapping from image to text space. *arXiv preprint arXiv:2209.15162*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751.

Jesse Mu and Jacob Andreas. 2020. Compositional explanations of neurons. *Advances in Neural Information Processing Systems*, 33:17153–17163.

Neel Nanda, Lawrence Chan, Tom Liberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*.

nostalgebraist. 2020. interpreting gpt: the logit lens.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020. Zoom in: An introduction to circuits. *Distill*. Https://distill.pub/2020/circuits/zoom-in.

Roma Patel and Ellie Pavlick. 2022. Mapping language models to grounded conceptual spaces. In *International Conference on Learning Representations*.

Tiago Pimentel, Naomi Saphra, Adina Williams, and Ryan Cotterell. 2020a. Pareto probing: Trading off accuracy for complexity. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3138–3153, Online. Association for Computational Linguistics.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020b. Information-theoretic probing for linguistic structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners.

Naomi Saphra and Adam Lopez. 2019. Understanding learning dynamics of language models with SVCCA. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3257–3267, Minneapolis, Minnesota. Association for Computational Linguistics.

Adam Scherlis, Kshitij Sachan, Adam S Jermyn, Joe Benton, and Buck Shlegeris. 2022. Polysemanticity and capacity in neural networks. *arXiv preprint arXiv:2210.01892*.

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

Nishant Subramani, Nivedita Suresh, and Matthew Peters. 2022. Extracting latent steering vectors from pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 566–581, Dublin, Ireland. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Mycal Tucker, Peng Qian, and Roger Levy. 2021. What if this modified that? syntactic interventions with counterfactual embeddings. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 862–875, Online. Association for Computational Linguistics.

Alex Turner, Monte MacDiarmid, David Udell, lisathiergart, and Ulisse Mini. 2023. Steering gpt-2-xl by adding an activation vector - ai alignment forum.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Andreas Veit, Michael J Wilber, and Serge Belongie. 2016. Residual networks behave like ensembles of relatively shallow networks. *Advances in neural information processing systems*, 29.

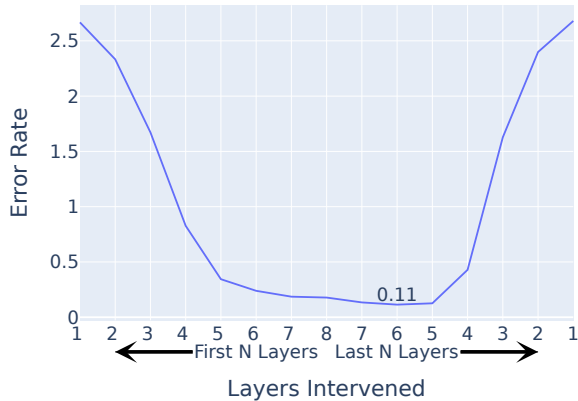| Hyperparameter	Value
Optimizer	AdamW
Learning Rate	1e-2
Weight Decay	1e-2
Betas	0.9, 0.99
Validation Step	200
Validation Size	512
Validation Patience	10

Table 4: Hyperparameters used for our linear probes.

Figure 7: Intervention results depending on layers intervened.

A Hyperparameters for Linear Probes

Table 4 provides hyperparameters used for our linear probes.

B Intervening on Different Layers

In practice there are a lot of ways to intervene using linear vectors. Figure 7 demonstrates different error rates depending on which layers are intervened. From our experiments, we observe that either a sufficient number of layers need to be intervened for OthelloGPT to alter its predictions. We offer a couple of hypotheses for this. First, we hypothesize that this is because of the residual structure of transformer models, and while each layer may write additional information into the residual streams, there may still be information from earlier layers that the model uses. A somewhat related hypothesis is that OthelloGPT might be demonstrating the Hydra effect (McGrath et al., 2023), in which language models demonstrate the ability to self-repair its computations after an intervention.

C Multiple Circuits

In Section 5.4, we find hints that OthelloGPT sometimes computes moves before boards at end games.

Namely, we check the earliest layers in which the board is correctly predicted with 100% accuracy. Could it be that at end games, legal moves can be predicted without needing the entire board? To this point, we experiment with variations of this experiment. In Figure 8, we check the earliest layer in which at least 90% of the board is first correctly computed. In Figure 9, we check the earliest layer in which the "minimum set" of tiles are correctly computed, where the minimum set is set of tiles that make each legal move playable (see Figure 10 for example). Despite a looser criteria for board state, we still see OthelloGPT computing moves before boards at end games.

Interestingly, our probes lose accuracy starts to drop in the end game as well (Figure 11). It is unclear whether 1) the model does not bother to compute the perfect board state, as alternative circuits might exist at end games, or 2) the model learns an alternative circuit because it struggles to compute the correct board state at end games.

D Evidence of Iterative Feature Refinements

As mentioned in Section 5.5, OthelloGPT demonstrates multiple evidence of iterative feature refinements: 1) Board state accuracy (as well as FLIPPED) improves from layer to layer (Table 1, 3). 2) Next-move predictions also improve from layer to layer. Table 5 reports the top-1 error rate when applying the unembedding layer on every layer using our test set from Section 3. As a baseline, we apply the same unembedding layer from OthelloGPT to the residual streams of a randomly initialized GPT model. 3) Linear probes across layers share similar directions. Figure 12 plots the cosine similarity between all linear probes, averaged across all 64 tiles and directions (MINE, YOURS, EMPTY).

E On Principled Ways of Probing

Probing has produced both excitement and skepticism amongst researchers (Belinkov, 2022). Here we provide our learnings regarding probing.

One criticism of probes is whether the discovered features are actually used by the model, i.e., correlation vs. causation. Intervention is commonly used to study causality (Giulianelli et al., 2018; Tucker et al., 2021), but have often reached mixed conclusions (Belinkov, 2022). While both linear and non-linear probes have demonstrated
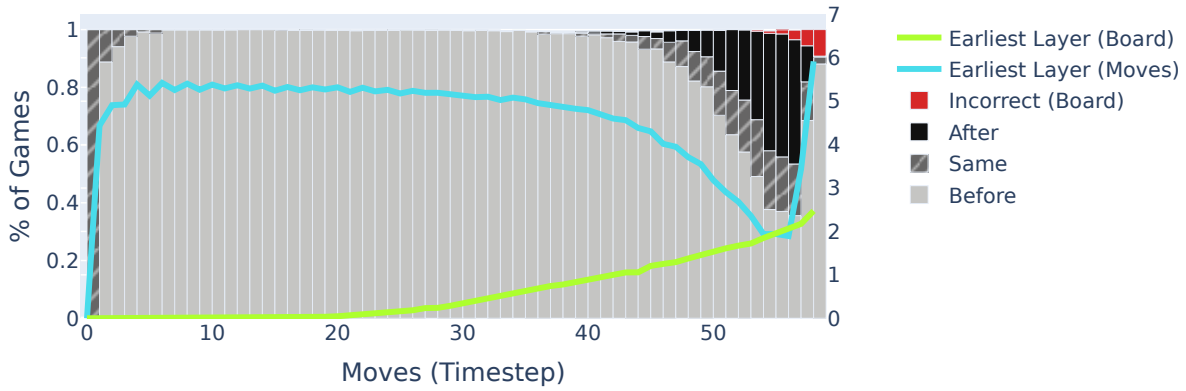
Figure 8: Percentage of times **90%** of the board state is computed before/after move predictions are made.
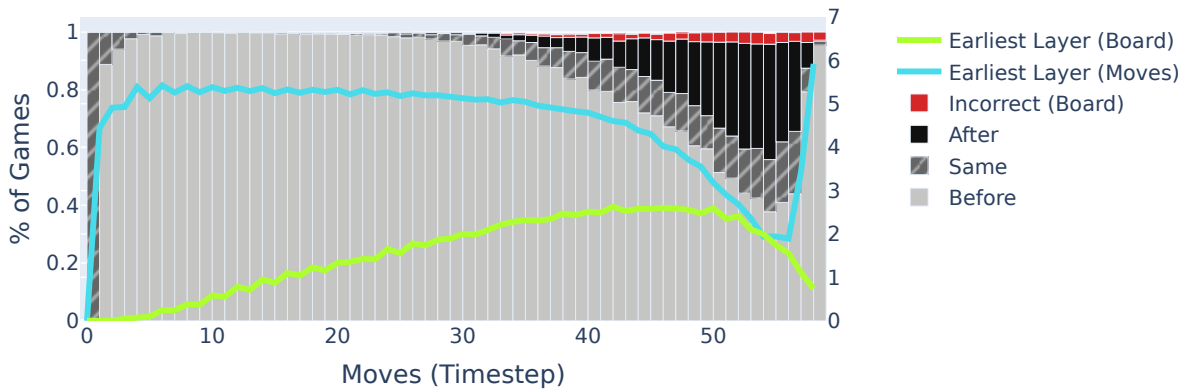

Figure 9: Percentage of times the **"minimum set"** of necessary board state is computed before/after move predictions are made.
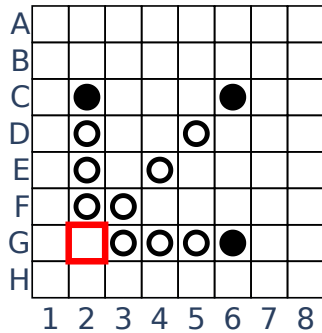

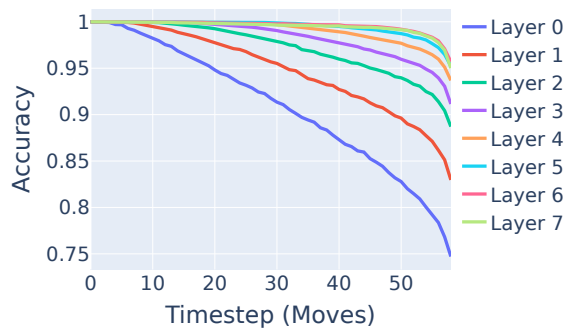Figure 10: Example of "minimum set" of tiles that make move G2 legal.

Figure 11: Accuracy per timestep for our linear probes.

successful interventions (Li et al., 2023b; Turner et al., 2023), linear probes are much easier to interpret, as they imply that features simply correspond to vectorial directions.

Another challenge is knowing which features to probe for, which can lead to pitfalls. Taking OthelloGPT as an example, classifying {BLACK, WHITE} versus {MINE, YOURS} leads to different

takeaways, which illustrates the danger of *projecting our preconceptions*.

Speaking of incorrect takeaways, our last point concerns the expressivity of probe models. With an expressive-enough probe, there is a danger of the probe computing or memorizing the desired feature that one is looking for, rather than extracting (Pimentel et al., 2020a; Saphra and Lopez, 2019). Still, some researchers view linear classification

29

Baseline: Random	x^0	x^1	x^2	x^3	x^4	x^5	x^6	x^7
0.856	0.215	0.152	0.112	0.079	0.049	0.015	0.004	0.001

Table 5: Top-1 error rates when applying the unembedding layer to earlier layers. As a baseline we apply OthelloGPT's unembedding layer on a randomly initialized GPT model.
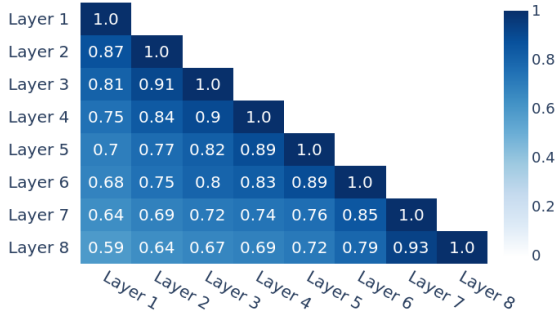

Figure 12: Cosine similarity scores between linear probes across layers.

as inadequate (Pimentel et al., 2020b; Saphra and Lopez, 2019). We view our work as evidence that linear probes do have interpretable and controllable power, and anticipate these findings to generalize to larger language models.