Tensor Product Representation Probes Reveal Shared Structure Across Linear Directions

Anonymous Authors[1]

Abstract

While researchers are finding concepts represented as linear directions in language models, a bag of linear directions fails to capture relational structure. To better understand this dichotomy, we study a model with known linear representations, but trained in a highly structured domain – the board game Othello. While the model's internal board-state representation is linearly decodable, we find additional structure in the form of tensor product representations (TPRs). We train TPR probes to recover shared structure amongst the linear probes, yielding a factorization into square-embeddings, color-embeddings, and a binding matrix that composes them to construct the model's board-state representation. We find geometric signatures within the weights of our TPR probe that align with the structure of the board, but perhaps more importantly, that the linear probes can be recovered directly from the parameters of our TPR probe. Our findings suggest that directional representations may be projections of more structured underlying representations.

1. Introduction

As Transformers become widely adopted in various domains, researchers are finding numerous forms in which they represent information. Among a popular line of work is the *linear representation hypothesis* (Park et al., 2023), which posits that Transformers represent concepts as linear directions in the model's activation space. This hypothesis has been supported by various empirical findings, with rank-1 representations for concepts like sentiment (Tigges et al., 2023), toxicity (Lee et al., 2024), refusal (Arditi et al., 2024), or even user attributes (Chen et al., 2024).

While the simplicity of linearity makes it an appealing representation, it fundamentally lacks structure by treating concepts as a bag of linear directions – which does not reflect how the world is structured (Wattenberg & Viégas, 2024). How do we reconcile this dichotomy?

We study this question by revisiting prior work with known linear representations in a domain with inherent structure – board games. We study OthelloGPT (Li et al., 2022), a Transformer model trained on the board game Othello.

OthelloGPT provided one of the earliest evidence for linear representations in Transformers. Namely, (Li et al., 2022) train a Transformer on move transcripts of Othello: given *only* move sequences as training data, the objective is to predict next legal moves that can follow. Interestingly, the model learns to internally represent the underlying board-state corresponding to the input moves, despite never being told about the existence of a board to begin with. Follow-up work by (Nanda et al., 2023) find *linear* representations of the board-state, in which the latent board-state can be linearly decoded from the model's activations when the board is viewed from an egocentric perspective (see Section 2).

However, Othello has additional structure in that squares and color pieces are distinct concepts, and pieces occupy different squares to form a board-state. Standard linear probing may yield many readout directions, but does not specify whether these directions arise from shared structured factors. In our work we find that such structure can be recovered with the right decoder architecture.

Namely, Smolensky (1990) demonstrates how symbolic structure, such as variable binding, could be represented in connectionist systems, using *tensor product representations* (TPR). TPR is a general framework for representing complex structures as the binding of simpler components, where the binding operation is implemented as a tensor product.

This framework is a natural fit for Othello, in that the board-state is a binding of two objects: the squares and the colors. Thus we design and train **Tensor Product Representation (TPR) probes** to decode OthelloGPT's board-state representations in a factorized manner, allowing us to recover "square-embeddings", "color-embeddings", and the binding matrix that composes the two objects.

TPR thus factorizes shared structure across linear probes

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

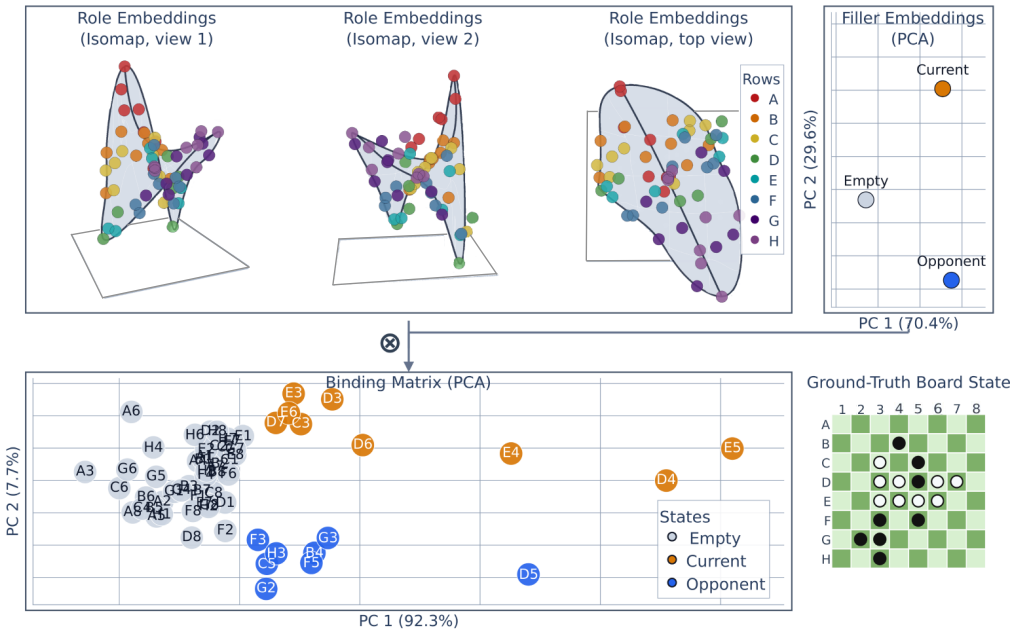

Figure 1. **Tensor product representation probes recover a structured factorization of OthelloGPT's board-state representation**, including (**i**) role (square) embeddings $\mathbf{R} \in \mathbb{R}^{64 \times d_r}$ (top-left, Isomap), (**ii**) filler (color) embeddings $\mathbf{F} \in \mathbb{R}^{3 \times d_f}$ (top-right, PCA), (**iii**) a binding matrix $\mathbf{B} \in \mathbb{R}^{d_r \times d_f}$ to bind the two objects (bottom-left, PCA) to represent a board-state (bottom-right).

(see Figure 1). Furthermore, the learned TPR weights exhibit geometric signatures that align with the structure of the board. Perhaps most importantly, we find that the independently trained linear probes from prior work can be recovered directly from the parameters of our TPR probe. Put differently, the linear probes are a projection of the more structured TPR probes.

Our findings suggest that meaningful structure may be hiding under our linear representations.

2. Background, Notations

In our work we study OthelloGPT (Li et al., 2022), an 8-layer Transformer trained on 20 million game transcripts of the board game Othello. Given only move sequences as training data ["D3", "E3", ...]), it predicts with near perfect accuracy the next legal moves that can follow. Importantly, OthelloGPT has no a priori knowledge of the game nor its rules, or even the existence of an underlying board.

Interestingly, (Li et al., 2022) demonstrate that OthelloGPT internally represents the underlying board-state in its activations. Subsequent work (Nanda et al., 2023) shows that such board-state representation is *linear* decodable when the board is viewed from an *egocentric* perspective. Namely, rather than labeling each square of the board as black or white pieces, one must consider whose turn it is to play at each token timestep. For odd timesteps, because it is black's turn to play, every black piece is labeled CURRENT while white pieces are labeled OPPONENT, and vice versa on even moves. Despite this nuance, we refer to each square's label as "color" for simplicity.

While a bag of linear probes can successfully decode the board-state, Othello has additional structure in that squares and colors are distinct concepts. In our work we demonstrate that such structure can be captured via tensor product representations (Smolensky, 1990).

Notations. We use boldface lowercase for vectors ($\mathbf{h}, \mathbf{w}$), boldface uppercase for matrices ($\mathbf{W}$), non-boldface for scalars or discrete symbols (e.g., s, c), and calligraphics to denote sets or lists ($\mathcal{C}$).

We denote OthelloGPT as $\boldsymbol{\theta}$, and an input sequence of moves as $\mathcal{X} := [x_0, \ldots, x_{T-1}]$ where x_i is a move token (e.g., $B5$). Every partial move sequence $\mathcal{X}_{:t}$ has a corresponding 8-by-8 board-state, denoted as $\mathcal{B}(\mathcal{X}_{:t}) \in \mathcal{C}^{8 \times 8}$, where $\mathcal{C} := \{\text{CURRENT}, \text{OPPONENT}, \text{EMPTY}\}$ is the set of possible labels for each square. For simplicity we refer to square labels $\mathcal{C}$ as "colors". Finally, $s \in \mathcal{S} := \{1, \ldots, 64\}$ denotes a square index.

3. Linear Probes vs. Tensor Product Representations

Here we describe linear probes and tensor product representation probes.

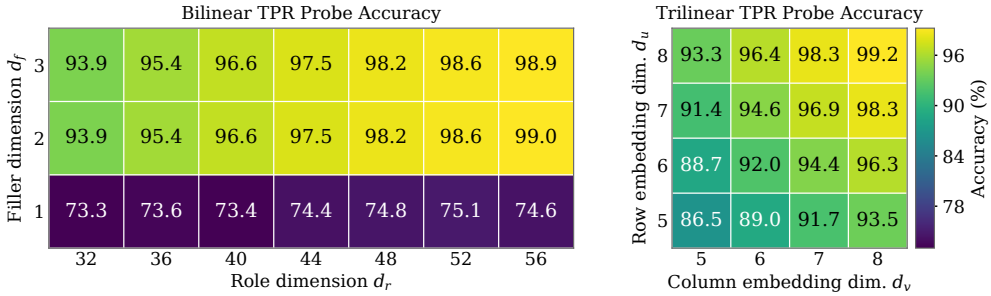

Figure 2. **TPR probe accuracy:** The board-state can be reconstructed using low-rank role embeddings (rank-d_r, or d_u, d_v) and filler embeddings (rank-d_f).

3.1. Probing methods

Linear Probes. Assume an input sequence $\mathcal{X}_{:t}$, its corresponding board-state $\mathcal{B}(\mathcal{X}_{:t}) \in \mathcal{C}^{8\times 8}$, and the model's activations at layer l as $\mathbf{h}_t^l \in \mathbb{R}^d$. Per square s, we train a linear probe $\mathbf{W}_s \in \mathbb{R}^{3\times d}$ to predict whether the model encodes that square s is occupied by color c:

$$\boldsymbol{\ell}_s^{probe} = \mathbf{W}_s \mathbf{h}_t^l \in \mathbb{R}^3, \tag{1}$$

$$\mathcal{L}_s^{probe} = \text{CrossEntropy}(\boldsymbol{\ell}_s^{probe}, \mathcal{B}(\mathcal{X}_{:t})_s). \tag{2}$$

where $\boldsymbol{\ell}_s^{probe}$ is the probe's logits for the three possible colors occupying square s and $\mathcal{B}(\mathcal{X}_{:t})_s \in \mathcal{C}$ is the groundtruth color of square s. Each row of $\mathbf{W}_s$ is a linear readout of whether s is occupied by color c. This yields $64 \times 3 = 192$ separate linear directions to represent the board.

Tensor Product Representation. Tensor product representations (TPRs, (Smolensky, 1990)) provide a classical account of how distributed vector representations can encode symbolic structure through role-filler binding. TPRs represent structured objects by superimposing the outer products of *roles* (e.g., squares of the board) and *fillers* (e.g., colors).

Formally, assume k possible roles and let $\mathbf{r}_i \in \mathbb{R}^{d_r}$ denote a vector representation for role i. Let $y(i)$ denote the filler occupying that role, and $\mathbf{f}_{y(i)} \in \mathbb{R}^{d_f}$ denote the vector representation of such filler. Then the resulting binding is

$$\mathbf{B} := \sum_{i=1}^{k} \mathbf{r}_i \otimes \mathbf{f}_{y(i)} \in \mathbb{R}^{d_r \times d_f}. \tag{3}$$

This framework fits naturally with Othello board-states. In our finite-dimensional setting, we identify the tensor product $\mathbf{r} \otimes \mathbf{f}$ in coordinates as the rank-1 outer product matrix $\mathbf{r}\mathbf{f}^\top \in \mathbb{R}^{d_r \times d_f}$. Treating each of the 64 squares as roles and colors as fillers, the board-state can be represented as

$$\mathbf{B} = \sum_{s\in\mathcal{S}} \mathbf{r}_s \otimes \mathbf{f}_{y(s)} = \sum_{s\in\mathcal{S}} \mathbf{r}_s \mathbf{f}_{y(s)}^\top \in \mathbb{R}^{d_r \times d_f}, \tag{4}$$

where $\mathbf{r}_s \in \mathbb{R}^{d_r}$ is the role vector for square s, $y(s) \in \mathcal{C}$ is the square's corresponding color, and $\mathbf{f}_{y(s)} \in \mathbb{R}^{d_f}$ is the filler vector for such color.

Training TPR Probes. We train TPR probes to decode OthelloGPT's board representation. The probe learns three sets of weights: role embeddings $\mathbf{R} := [\mathbf{r}_1^\top; \ldots; \mathbf{r}_{64}^\top] \in \mathbb{R}^{64\times d_r}$, filler embeddings $\mathbf{F} := [\mathbf{f}_{empty}^\top; \mathbf{f}_{current}^\top; \mathbf{f}_{opponent}^\top] \in \mathbb{R}^{3\times d_f}$, and a linear map $\mathbf{M} \in \mathbb{R}^{d_r \times d_f \times d_{\text{model}}}$ from hidden states to a latent binding matrix $\mathbf{B}$. Given a hidden state $\mathbf{h}_t^l$, the probe first constructs the binding matrix $\mathbf{B}$.

$$\mathbf{B} := \mathbf{M}(\mathbf{h}_t^l) \in \mathbb{R}^{d_r \times d_f}. \tag{5}$$

It then scores each square–color pair with a bilinear unbinding:

$$\ell_{s,c} = \mathbf{r}_s^\top \mathbf{B}\, \mathbf{f}_c \tag{6}$$

The binding matrix $\mathbf{B}$ encodes which roles (squares) are occupied by which fillers (colors), and unbinding with the learned role $\mathbf{r}_s$ and filler $\mathbf{f}_c$ recovers the correct color c for square s.

Trilinear TPR Probes. Note that bindings do not have to be bilinear. While a natural decomposition for Othello is between the board and colors, the board is also structured with 8 rows and columns. Thus an alternative *trilinear* TPR probe may encode the state of square s (row i, column j) as

$$\mathbf{T} := \mathbf{M}(\mathbf{h}_t^l) \in \mathbb{R}^{d_u \times d_v \times d_f}, \ell_{ij,c} = \langle \mathbf{T}, \mathbf{u}_i \otimes \mathbf{v}_j \otimes \mathbf{f}_c \rangle \tag{7}$$

where $\mathbf{M} \in \mathbb{R}^{d_u \times d_v \times d_f \times d_{model}}$ and $\otimes$ denotes outer products. The probe learns four sets of weights: **(i)** a mapping $\mathbf{M}$ from hidden states to the binding space, **(ii)** row embeddings $\mathbf{U} := [\mathbf{u}_1^\top; \ldots; \mathbf{u}_8^\top] \in \mathbb{R}^{8\times d_u}$, where each row ($\mathbf{u}_i$) corresponds to a row embedding, **(iii)** a column embedding matrix $\mathbf{V} := [\mathbf{v}_1^\top; \ldots, ; \mathbf{v}_8^\top] \in \mathbb{R}^{8\times d_v}$, and **(iv)** the same color embedding matrix $\mathbf{F} \in \mathbb{R}^{3\times d_f}$ as before.

3.2. Probing Results

In line with (Nanda et al., 2023), our linear probes achieve 99% accuracy. For TPR probes, Figure 2 reports the average accuracy from layer 7 (out of 8) over all 64 squares (see

Appendix B for all other layers). In the case of bilinear, we sweep over multiple values of the role dimension d_r and filler dimension d_f. For trilinear probes we fix $d_f = 2$ and sweep over multiple values of d_u, d_v. Note that the TPR probe only requires low-rank role and filler embeddings to achieve 99% accuracy (note that $d_{\text{model}} = 512$). Also note that the TPR probe requires at least 2 dimensions for filler embeddings, indicating that high accuracy is not being achieved by role embeddings alone.

It is worth noting that the TPR probes have *fewer* parameters than the linear probe. The linear probe has $192 \times d_{\text{model}}$ parameters ($98,304$ with $d_{\text{model}} = 512$), while bilinear TPRs have $(64 \times d_r) + (3 \times d_f) + (d_r \times d_f \times d_{\text{model}})$ ($56,582$ with $d_r = 52$, $d_f = 2$; 57.5% of the linear probe's parameters). For trilinear probes we have $(8 \times d_u) + (8 \times d_v) + (3 \times d_f) + (d_u \times d_v \times d_f \times d_{\text{model}})$ ($65,670$ with $d_u = d_v = 8$, $d_f = 2$). Despite having fewer parameters and additional structural constraints, the TPR probes achieve near perfect accuracy.

3.3. Visualizing TPR Probes

Our TPR probe provides a structured decomposition of the model's hidden states into role (square) embeddings, filler (color) embeddings, and a binding matrix that wires the two together. Figure 1 visualizes all three components using Isomap and PCA. The top left panel visualizes the role (square) embeddings $\mathbf{R} \in \mathbb{R}^{64 \times d_r}$ using Isomap, with points color-coded by each square's row. The resulting 3D embedding demonstrates a saddle-like manifold with visible features of the board: the upper curve captures rows A through H, while the lower curve captures columns 1 through 8.

The top right panel visualizes the filler (color) embeddings $\mathbf{F} \in \mathbb{R}^{3 \times d_f}$ using PCA. Note that principal component (PC) 1 separates EMPTY vs. occupied, while PC 2 separates CURRENT vs. OPPONENT.

These two components, $\mathbf{R}$ and $\mathbf{F}$, are based on the TPR probe's parameters alone, and not based on any input. The bottom left panel visualizes an example of the binding matrix $\mathbf{B} \in \mathbb{R}^{d_r \times d_f}$, which is input-specific, using PCA. The bottom right panel shows its corresponding ground-truth board-state. Note that the binding matrix captures all the correct square–color bindings, while all 64 squares form clusters that mirror the structure of the filler embeddings. The center four squares (D4, D5, E4, E5) are outliers because according to the rules of Othello, they can never be empty – the game starts with the 4 squares already filled, and a square that is filled can never turn empty.

4. Recovered Structure of TPR Probes

Here we study the structure recovered by TPR probes.

4.1. Interventions

To validate that the TPR probes not only recover a valid structure but also one that captures causal mechanisms for next-move prediction, we run causal interventions. We intervene on the model's internal board-state representation and check whether this leads to the expected changes in the model's next-move predictions. Our intervention methods for both linear and TPR probes are described below.

Setup. Recall that $\boldsymbol{\theta}$ denotes OthelloGPT, $\mathcal{X}$ denotes an input move sequence, and $\mathcal{B}(\mathcal{X}) \in \{\text{CURRENT}, \text{OPPONENT}, \text{EMPTY}\}^{8\times 8}$ denotes the corresponding groundtruth board-state. Let $\boldsymbol{\ell} = \boldsymbol{\theta}(\mathcal{X}) \in \mathbb{R}^{64}$ be OthelloGPT's logits for next-move predictions, which we reshape and convert to binary predictions using a probability threshold: $\mathbf{m}_{orig} := \mathbb{I}(\text{Softmax}(\boldsymbol{\ell}) > 0.01) \in \{0,1\}^{8\times 8}$.

We manipulate the Transformer's internal board-state representation to check whether it is causally responsible for the model's next move predictions. To do so, let $\mathcal{B}^{(target)}$ be a *target* board-state that we wish the model to represent instead. To create $\mathcal{B}^{(target)}$, we randomly select a non-empty cell (i, j) from $\mathcal{B}$ and either flip its value (e.g., $\mathcal{B}^{(target)}_{ij} = \text{CURRENT} \rightarrow \text{OPPONENT}$) or set it to empty ($\mathcal{B}^{(target)}_{ij} = \text{EMPTY}$). Given the modified board-state $\mathcal{B}^{(target)}$, let $\mathbf{m}_{target} \in \{0,1\}^{8\times 8}$ denote its corresponding set of valid next moves according to the rules of Othello. We validate that every target board-state $\mathcal{B}^{(target)}$ is a legal board-state (i.e., there can be no disconnected "islands" of pieces) and that the new set of legal moves $\mathbf{m}_{target}$ does not equal the original set of legal moves $\mathbf{m}_{orig}$.

We then intervene on the Transformer's internal board-state to match $\mathcal{B}^{(target)}$ (which we describe how below), and check how the model's new next-move predictions $\mathbf{m}_{interv}$ compare to $\mathbf{m}_{target}$.

Linear probe interventions. Let $\mathbf{W}_s \in \mathbb{R}^{3\times d}$ denote the linear probe for square s, and $\mathbf{w}_{s,c} \in \mathbb{R}^d$ denote the row corresponding to color c in $\mathbf{W}_s$. Given the hidden-state $\mathbf{h}$, a high dot-product $\mathbf{w}_{s,c}^\top \mathbf{h}$ indicates that square s is occupied by color c. A simple intervention is to add $\mathbf{w}_{s,c}$ to the hidden state $\mathbf{h}$: $\widehat{\mathbf{h}} = \mathbf{h} + \alpha \frac{\mathbf{w}_{s,c}}{\|\mathbf{w}_{s,c}\|}$, where α is a scaling factor that controls the strength of the intervention.

TPR probe interventions. For TPR probes, the decoding of each square–color pair is now bilinear: $\ell_{s,c} = \mathbf{r}_s^\top \mathbf{B} \mathbf{f}_c$. $\mathbf{B}$ wires square and color vectors such that the bilinear product is high when s is occupied by c. Recall that the binding $\mathbf{B}$ can be expressed as a sum of outer products, $\sum_{s\in\mathcal{S}} \mathbf{r}_s \mathbf{f}_{y(s)}^\top$ where $y(s)$ denotes the color of square s. One simple intervention is to modify the binding $\mathbf{B}$ by swapping a single outer product $\mathbf{r}_s \mathbf{f}_c^\top$. To intervene on square s from its original color $y(s)$ to a new color $\widehat{y}(s)$, we replace the

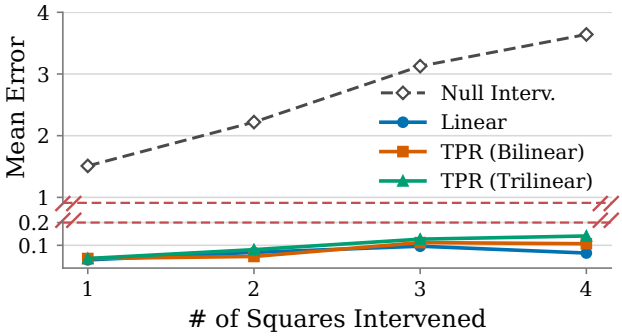

Figure 3. **Intervention results for linear & TPR probes.**

outer product $\mathbf{r}_s\mathbf{f}_{y(s)}^\top$ with $\mathbf{r}_s\mathbf{f}_{\widehat{y}(s)}^\top$, after which we map the change in the binding matrix ($\Delta\mathbf{B}$) back to the hidden state space using the pseudo-inverse of $\mathbf{M}$:

$$\Delta\mathbf{B} := \mathbf{r}_s(\mathbf{f}_{\widehat{y}(s)} - \mathbf{f}_{y(s)})^\top, \tag{8}$$

$$\mathbf{M}_{\text{flat}} := \text{reshape}(\mathbf{M}, (d_r d_f) \times d_{model}), \tag{9}$$

$$\mathbf{z}_s := \mathbf{M}_{\text{flat}}^+ \,\text{vec}(\Delta\mathbf{B}), \tag{10}$$

$$\widehat{\mathbf{h}} = \mathbf{h} + \alpha \frac{\mathbf{z}_s}{\|\mathbf{z}_s\|} \tag{11}$$

Interventions with trilinear probes is the same procedure, but replaces $\Delta\mathbf{B}$ with $\Delta\mathbf{T}$:

$$\Delta\mathbf{T}_{ij} := \mathbf{u}_i \otimes \mathbf{v}_j \otimes (\mathbf{f}_{\widehat{y}(i,j)} - \mathbf{f}_{y(i,j)}) \tag{12}$$

Composing multiple interventions. Since both linear and TPR probe interventions are linear operations, we should be able to compose multiple interventions by simply adding more intervention vectors. Imagine we wish to alter the color of k squares $\{s_i\}_{i=0}^{k-1}$ from their original colors $\{y(s_i)\}_{i=0}^{k-1}$ to alternate colors $\{\widehat{y}(s_i)\}_{i=0}^{k-1}$. Interventions can then be composed as follows:

$$\text{Linear} : \widehat{\mathbf{h}} = \mathbf{h} + \sum_{i=0}^{k-1} \alpha_i \frac{\mathbf{w}_{s_i,\widehat{y}(s_i)}}{\|\mathbf{w}_{s_i,\widehat{y}(s_i)}\|}, \tag{13}$$

$$\text{TPR (Bilinear)} : \Delta\mathbf{B} := \sum_{i=0}^{k-1} \beta_j \mathbf{r}_{s_i}(\mathbf{f}_{\widehat{y}(s_i)} - \mathbf{f}_{y(s_i)})^\top, \tag{14}$$

where β_j is also a scaling factor and the remaining steps are the same as the single intervention case. A similar composition can be done for trilinear probes by replacing $\Delta\mathbf{B}$ with $\Delta\mathbf{T}$. For α_i, β_j, we sweep over all possible combinations over values $\{0.25, 0.5, 0.75, \ldots 2.5\}$ per test sample. We intervene on every layer at the last timestep.

Evaluation. We test our interventions on 1,000 held out samples. We compare the intervened move predictions $\mathbf{m}_{interv} \in \{0,1\}^{8\times 8}$ against the groundtruth set of valid moves $\mathbf{m}_{target} \in \{0,1\}^{8\times 8}$ corresponding to the target board-state $\mathcal{B}^{(target)}$. Per prior work (Li et al., 2022; Nanda et al., 2023), we report *mean error count* – the average number of false positives and false negatives given $\mathbf{m}_{interv}$ and $\mathbf{m}_{target}$. As a baseline measure we report the average number of errors under a null intervention – i.e., the number of errors if we simply compare the model's original next-move predictions $\mathbf{m}_{orig}$ against $\mathbf{m}_{target}$.

Results. Figure 3 shows the results. For our bilinear TPR probe, we use $d_r = 52, d_f = 2$ and for our trilinear probe we use $d_u, d_v = 8, d_f = 2$. All interventions achieve near zero error counts, even when multiple interventions are composed. This confirms that the TPR probes have not only recovered a valid structure, but also that it captures causal mechanisms for next-move prediction.

4.2. TPR vs. Linear Directions

How do our recovered structure relate to linear probes? We study this question by deriving "effective linear probes" $\widetilde{\mathbf{W}}_{s,c}$ from the parameters of our TPR probes, by projecting them onto the activation space, the same space from which linear probes were trained:

$$\text{Bilinear} : \tilde{\mathbf{w}}_{s,c} := \mathbf{M}_{\text{flat}}^\top \text{vec}(\mathbf{r}_s\mathbf{f}_c^\top), \in \mathbb{R}^{d_{\text{model}}}, \tag{15}$$

$$\text{Trilinear} : \tilde{\mathbf{w}}_{ij,c} := \mathbf{M}_{\text{flat}}^\top \text{vec}(\mathbf{u}_i \otimes \mathbf{v}_j \otimes \mathbf{f}_c) \in \mathbb{R}^{d_{\text{model}}}. \tag{16}$$

We then measure the cosine similarity between these effective linear probes $\widetilde{\mathbf{W}}$ and the independently trained linear probes $\mathbf{W}$. Because the probes go through a softmax, each (effective) probe could be offset by a constant vector without affecting its logits. Thus we mean-center $\mathbf{W}$ and $\widetilde{\mathbf{W}}$ before taking cosine similarities. Figure 4 shows the results for two bilinear TPR probes: a "full-dimensional" one with $d_r = 64$ and a "compressed" TPR probe with $d_r = 56$ (see Figure 9 for trilinear probes).

In the full-dimensional case, we see near perfect alignment with the linear probes for all square–color pairs, suggesting that the TPR probe learns a simple reparameterization of $\mathbf{W} \in \mathbb{R}^{192\times d_{\text{model}}}$, which we validate in Appendix C. In the latter compressed case, the TPR probe no longer has enough dimensions to simply reshape $\mathbf{W}$, as it must go through the binding matrix $\mathbf{B} \in \mathbb{R}^{d_r\times d_f}$ as a bottleneck. However, we still see significantly high cosine similarity scores, suggesting that the same linear directions can be nearly recovered with our TPR probes.

To summarize, the TPR probes learn the same "effective" directions as the linear probes, but with additional structure. This suggests that linear directions may be projections of more structured underlying components. Because the full-dimensional case is simply a reparameterization of the linear

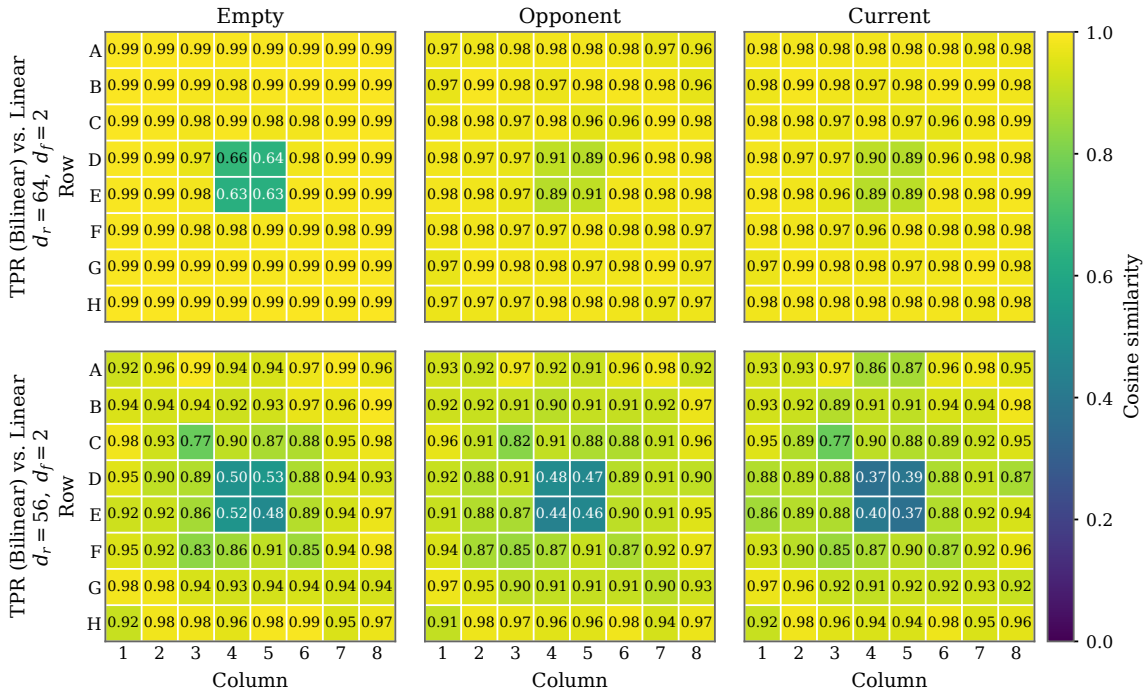

Figure 4. **Cosine similarity scores between linear probes vs. "effective linear probes" from TPR probes.** Linear probes can be recovered from the parameters of our TPR probes, suggesting that linear directions may be a projection of more structured underlying components.

probes, for the rest of the analyses we use a compressed TPR probe with $d_r = 52$.

Local vs. distributed codes. The two cases above are closely related to the coding scheme described in (Thorpe, 1989). A local code encodes each concept (e.g., a square–color pair) with a single dimension, whereas a distributed code encodes a concept across multiple dimensions. When $d_r = 64$, the TPR probe has enough dimensions to index all linear probes, yielding a local code of all 192 square–color probe directions (see Appendix C). With $d_r < 64$, the probe no longer has enough dimensions, yielding a distributed code (i.e., superposition).

4.3. Structural Decomposition or Simply Low-Rank?

Thus far we show that TPR yields a structural low-rank decomposition of the linear probes $\mathbf{W}$ – is this because $\mathbf{W}$ is simply low-rank?

Thus we compare the accuracy of rank-k truncated SVD of $\mathbf{W}$ (denoted $\texttt{SVD}_k(\mathbf{W})$) against our TPR probe ($d_r = 52$). We sweep over k and compare both the accuracy but also the number of parameters in $\texttt{SVD}_k(\mathbf{W})$ with respect to the TPR probes. Figure 5 shows the results. The x-axis indicates the number of parameters in $\texttt{SVD}_k(\mathbf{W})$ as the percentage of the TPR probe's number of parameters and the y-axis indicates probing accuracy. At $k = 80$, $\texttt{SVD}_k(\mathbf{W})$ matches the number of parameters as our TPR probe but only reaches 85% accuracy, and only reaches 99% accuracy at $k = 120$ (150% of the TPR probe's number of parameters). This suggests that the TPR probe has learned a structural decomposition beyond just a low-rank decomposition.

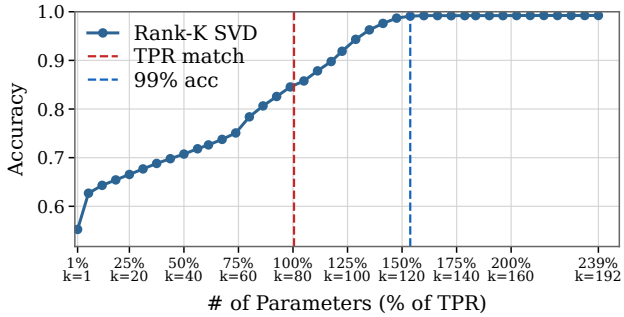

Figure 5. **Rank-k truncated SVD accuracy.** At rank-80, $\texttt{SVD}_k(\mathbf{W})$ matches the number of parameters as our TPR probe, but only achieves 85% accuracy.

4.4. Geometric Signatures of Board Structure in TPR Weights

Figure 1 suggests that the learned square embeddings $\mathbf{R} \in \mathbb{R}^{64 \times d_r}$ recover a geometry aligned with the structure of the Othello board. We now quantify this observation. Recall that each row $\mathbf{r}_s \in \mathbb{R}^{d_r}$ corresponds to a square s on the board, and we use a TPR probe with $d_r = 52$.

Baselines. To distinguish the geometry induced by OthelloGPT from that induced by either the TPR architecture or

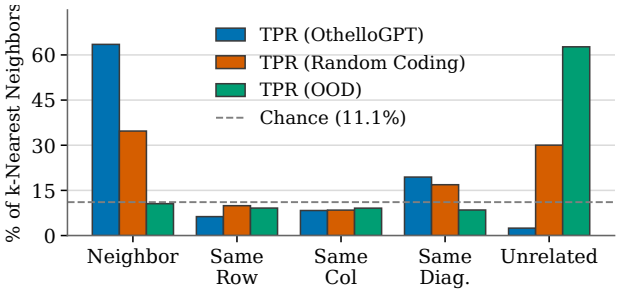

Figure 6. **Local k-NN based classification of neighbors.**

board-state statistics alone, we compare against two baselines. **(i) TPR-OOD** is a TPR probe trained on out-of-distribution board-states in which the colors of the squares are sampled independently at random. These are likely invalid board-states, but allow us to test whether the TPR architecture alone induces board-aligned geometry. **(ii) TPR-Random Coding** is a TPR probe trained on the same distribution of board-states as the original TPR probes, but replaces OthelloGPT activations with synthetic encodings. Namely, for each square–color pair (s, c), we sample a random vector $\mathbf{q}_{s,c} \in \mathbb{R}^{d_{\text{model}}}$ and encode each board-state by summing the 64 corresponding square–color vectors. This baseline tests how much geometry can be explained by the board-state distribution together with the TPR architecture, rather than by structure present in OthelloGPT's activations. Both baseline probes achieve 99% accuracy on their respective in-domain distributions.

Local neighborhood structure. We first evaluate local neighborhood structure. Depending on its position, each square s has k_s adjacent board neighbors (including diagonals). For each square s, we retrieve the k_s-nearest neighbors of $\mathbf{r}_s$ in embedding space and classify each retrieved square into five disjoint categories: true board neighbor, same row, column, diagonal, or unrelated.

As shown in Figure 6, the square embeddings recovered from OthelloGPT exhibit much stronger local agreement with the board than either baseline: roughly 60% of the retrieved nearest neighbors are true board neighbors, and most of the remaining retrieved squares lie on the same row, column, or diagonal. In contrast, the baselines retrieve substantially more unrelated squares. This suggests that the square embeddings recovered from OthelloGPT reflect local board geometry beyond what is induced by the TPR architecture or data distribution alone.

Pairwise board geometry. We next evaluate pairwise board geometry. For each pair of squares $s = (i, j), s' = (i', j')$, we compute their row and column gaps $\Delta i = |i - i'|$ and $\Delta j = |j - j'|$. We then group all $\binom{64}{2} = 2016$ square pairs by $(\Delta i, \Delta j)$, and compute the average cosine similarity of $\mathbf{r}_s$ and $\mathbf{r}_{s'}$ within each group:

$$\text{GapSim}(\Delta i, \Delta j) = \mathbb{E}_{s,s':|i-i'|=\Delta i,|j-j'|=\Delta j}\left[\cos(\mathbf{r}_s, \mathbf{r}_{s'})\right] \tag{17}$$

This gives an 8×8 matrix of average cosine similarities, where the entry at row Δi and column Δj corresponds to the average cosine similarity between pairs of squares with row gap Δi and column gap Δj. Figure 7 shows that nearby squares on the same row ($\Delta i = 0$), column ($\Delta j = 0$), or diagonal ($\Delta i = \Delta j$) have higher cosine similarity. Some of this structure also appears in the random-coding baseline, indicating that the TPR architecture and board-state distribution can induce some board-aligned geometry. However, the effect is strongest for the TPR probe trained on OthelloGPT.

To summarize this effect quantitatively, we evaluate how much of the variance in pairwise cosine similarities is explained by row and column gaps. For each pair of squares, we use the corresponding gap-based average $\text{GapSim}(\Delta i, \Delta j)$ as the predictions for its cosine similarity. This yields a R^2 score of 0.54 for TPR-OthelloGPT, compared to 0.24 for TPR-Random Coding and 0.03 for TPR-OOD. The pairwise geometry of the embeddings recovered from OthelloGPT is substantially more aligned with board-relative row and column gaps than the geometry recovered by either baseline.

Overall, these results suggest that the learned square embeddings recover board-aligned geometry that is not explained by the TPR architecture or board-state statistics alone.

5. Related Work

Here we provide an abridged overview of related work, with a more extensive one in Appendix A.

Tensor product representations (Smolensky, 1990) have long provided a framework for encoding compositional or relational structure in neural networks (Huang et al., 2018; Tang et al., 2018; Park et al., 2024b; Schlag & Schmidhuber, 2018). In Transformers, TPR-style ideas have been used to improve performance on tasks such as mathematics and abstractive summarization (Schlag et al., 2019; Jiang et al., 2021). Closest to our setting, (McCoy et al., 2018) show that RNN hidden-states can be reconstructed using Tensor Product Decomposition Networks.

Meanwhile, a growing body of work finds that features are not always well described by isolated rank-1 directions (Mueller et al., 2025). Rather, researchers are identifying low-dimensional manifolds and coordinate systems for concept representations (Engels et al., 2024; Kantamneni & Tegmark, 2025; Modell et al., 2025; Gurnee et al., 2025; Sarfati et al., 2026; Lee et al., 2026).

This shift has motivated new methods for recovering faithful

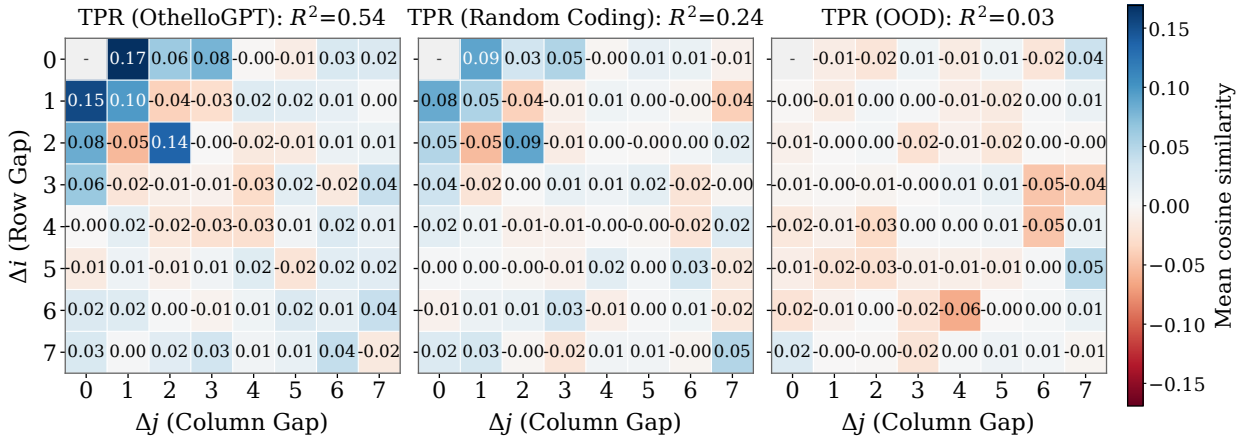

Figure 7. **Pairwise Board Geometry.** Each entry shows the average cosine similarity between pairs of square embeddings that are Δi rows and Δj columns apart on the board. Pairs that are close on the same row ($\Delta i = 0$), column ($\Delta j = 0$), or diagonal ($\Delta i = \Delta j$) exhibit higher cosine similarity.

structure from model activations. One line of work develops geometry-aware sparse autoencoders (SAEs) based on explicit structural assumptions (Hindupur et al., 2025; Costa et al., 2025; Bhalla et al., 2025; Bussmann et al., 2025). Our work highlights a complementary issue, as prior decompositions do not consider the possible *interactions* (i.e., binding) amongst components.

6. Discussions, Limitations

We study the dichotomy between linear directions and structured representations by studying OthelloGPT, a model with known linear representations yet trained in a domain with inherent structure. Our TPR probes factorizes shared structure across linear probes that can not only reconstruct the independently trained linear probes, but also exhibit geometry that reflects the structure of Othello. Could it be that linear directional representations in general are projections of more complex structures hiding underneath? We conclude with a few thoughts:

Structured Decoding. Our work demonstrates that linear directions can be factorized into components with shared structure. One limitation of TPR probes is that one must know a priori what structure to look for. While our specific TPR configuration is not meant to be a universal fit for all domains, we anticipate domain-specific structural probes to recover faithful components across various models from different domains.

Unsupervised Structure Recovery. While SAEs have become a popular method for decomposition, they typically lack structure by treating concepts as a bag of linear directions. While other geometry-aware methods have been suggested, they do not account for the possible interactions between components (e.g., binding). Could it be that some of the latents being recovered by SAEs correspond to such component-wise interactions? If so, how might we interpret them? Furthermore, while our TPR probes only have a single layer of binding, one could have multiple layers to represent hierarchical or nested structure.

TPR vs. Feature Subspaces, Mechanisms. Note that we are *not* claiming that OthelloGPT performs tensor products. While Section 4.4 reveals some geometric structure, we are also not claiming that OthelloGPT has separate square or color subspaces either. Rather, we demonstrate that TPRs can provide a factorization of shared structure across a bag of linear directions. We do not study how our recovered structures relate to OthelloGPT's mechanisms and leave this exploration for future work.

References

Arditi, A., Obeso, O., Syed, A., Paleka, D., Panickssery, N., Gurnee, W., and Nanda, N. Refusal in language models is mediated by a single direction. *Advances in Neural Information Processing Systems*, 37:136037–136083, 2024.

Bai, X., Pres, I., Deng, Y., Tan, C., Shieber, S., Viégas, F., Wattenberg, M., and Lee, A. Why can't transformers learn multiplication? reverse-engineering reveals long-range dependency pitfalls. *arXiv preprint arXiv:2510.00184*, 2025.

Bhalla, U., Oesterling, A., Verdun, C. M., Lakkaraju, H., and Calmon, F. P. Temporal sparse autoencoders: Leveraging the sequential nature of language for interpretability. *arXiv preprint arXiv:2511.05541*, 2025.

Bussmann, B., Nabeshima, N., Karvonen, A., and Nanda, N. Learning multi-level features with matryoshka sparse autoencoders. *arXiv preprint arXiv:2503.17547*, 2025.

Chen, Y., Wu, A., DePodesta, T., Yeh, C., Li, K., Marin, N. C., Patel, O., Riecke, J., Raval, S., Seow, O., et al. Designing a dashboard for transparency and control of conversational ai. *arXiv preprint arXiv:2406.07882*, 2024.

Costa, V., Fel, T., Lubana, E. S., Tolooshams, B., and Ba, D. From flat to hierarchical: Extracting sparse representations with matching pursuit. *arXiv preprint arXiv:2506.03093*, 2025.

Engels, J., Michaud, E. J., Liao, I., Gurnee, W., and Tegmark, M. Not all language model features are one-dimensionally linear. *arXiv preprint arXiv:2405.14860*, 2024.

Gurnee, W., Ameisen, E., Kauvar, I., Tarng, J., Pearce, A., Olah, C., and Batson, J. When models manipulate manifolds: The geometry of a counting task. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/linebreaks/index.html.

Hindupur, S. S. R., Lubana, E. S., Fel, T., and Ba, D. Projecting assumptions: The duality between sparse autoencoders and concept geometry. *arXiv preprint arXiv:2503.01822*, 2025.

Huang, Q., Smolensky, P., He, X., Deng, L., and Wu, D. Tensor product generation networks for deep nlp modeling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1263–1273, 2018.

Jiang, Y., Celikyilmaz, A., Smolensky, P., Soulos, P., Rao, S., Palangi, H., Fernandez, R., Smith, C., Bansal, M., and Gao, J. Enriching transformers with structured tensor-product representations for abstractive summarization. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y. (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4780–4793, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.381. URL https://aclanthology.org/2021.naacl-main.381/.

Kantamneni, S. and Tegmark, M. Language models use trigonometry to do addition. *arXiv preprint arXiv:2502.00873*, 2025.

Lee, A., Bai, X., Pres, I., Wattenberg, M., Kummerfeld, J. K., and Mihalcea, R. A mechanistic understanding of alignment algorithms: A case study on dpo and toxicity. *arXiv preprint arXiv:2401.01967*, 2024.

Lee, A., Sun, L., Wendler, C., Viégas, F., and Wattenberg, M. The geometry of self-verification in a task-specific reasoning model. *arXiv preprint arXiv:2504.14379*, 2025a.

Lee, A., Weber, M., Viégas, F., and Wattenberg, M. Shared global and local geometry of language model embeddings. *arXiv preprint arXiv:2503.21073*, 2025b.

Lee, A., Belinkov, Y., Viégas, F., and Wattenberg, M. Decomposing query-key feature interactions using contrastive covariances. *arXiv preprint arXiv:2602.04752*, 2026.

Li, K., Hopkins, A. K., Bau, D., Viegas, F., Pfister, H., and Wattenberg, M. Emergent world representations: Exploring a sequence model trained on a synthetic task. *arXiv preprint arXiv:2210.13382*, 2022.

Luo, G., Feng, J., Darrell, T., Radford, A., and Steinhardt, J. Learning a generative meta-model of llm activations. *arXiv preprint arXiv:2602.06964*, 2026.

McCoy, R. T., Linzen, T., Dunbar, E., and Smolensky, P. Rnns implicitly implement tensor-product representations. In *International Conference on Learning Representations*, 2018.

Modell, A., Rubin-Delanchy, P., and Whiteley, N. The origins of representation manifolds in large language models. *arXiv preprint arXiv:2505.18235*, 2025.

Muchane, M., Richardson, S., Park, K., and Veitch, V. Incorporating hierarchical semantics in sparse autoencoder architectures. *arXiv preprint arXiv:2506.01197*, 2025.

Mueller, A., Lee, A., Joshi, S., Lubana, E. S., Sridhar, D., and Reizinger, P. From isolation to entanglement: When do interpretability methods identify and disentangle known concepts? *arXiv preprint arXiv:2512.15134*, 2025.

Nanda, N., Lee, A., and Wattenberg, M. Emergent linear representations in world models of self-supervised sequence models. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 16–30, 2023. URL https://arxiv.org/abs/2309.00941.

Park, C. F., Lee, A., Lubana, E. S., Yang, Y., Okawa, M., Nishi, K., Wattenberg, M., and Tanaka, H. Iclr: In-context learning of representations. *arXiv preprint arXiv:2501.00070*, 2024a.

Park, K., Choe, Y. J., and Veitch, V. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

Park, T., Choi, I., and Lee, M. Attention-based iterative decomposition for tensor product representation. *arXiv preprint arXiv:2406.01012*, 2024b.

Sarfati, R., Bigelow, E., Wurgaft, D., Merullo, J., Geiger, A., Lewis, O., McGrath, T., and Lubana, E. S. The shape of beliefs: Geometry, dynamics, and interventions along representation manifolds of language models' posteriors. *arXiv preprint arXiv:2602.02315*, 2026.

Schlag, I. and Schmidhuber, J. Learning to reason with third order tensor products. *Advances in neural information processing systems*, 31, 2018.

Schlag, I., Smolensky, P., Fernandez, R., Jojic, N., Schmidhuber, J., and Gao, J. Enhancing the transformer with explicit relational encoding for math problem solving. *arXiv preprint arXiv:1910.06611*, 2019.

Shai, A., Amdahl-Culleton, L., Christensen, C. L., Bigelow, H. R., Rosas, F. E., Boyd, A. B., Alt, E. A., Ray, K. J., and Riechers, P. M. Transformers learn factored representations. *arXiv preprint arXiv:2602.02385*, 2026.

Smolensky, P. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2):159–216, 1990.

Sun, L., Yan, L., Lu, X., Lee, A., Zhang, J., and Shao, J. Valence-arousal subspace in llms: Circular emotion geometry and multi-behavioral control. *arXiv preprint arXiv:2604.03147*, 2026.

Tang, S., Smolensky, P., and de Sa, V. Learning distributed representations of symbolic structure using binding and unbinding operations. *arXiv preprint arXiv:1810.12456*, 2018.

Thorpe, S. J. Local vs. distributed coding. *Intellectica*, 8:3–40, 1989. URL https://api.semanticscholar.org/CorpusID:70175501.

Tigges, C., Hollinsworth, O. J., Geiger, A., and Nanda, N. Linear representations of sentiment in large language models. *arXiv preprint arXiv:2310.15154*, 2023.

Wattenberg, M. and Viégas, F. B. Relational composition in neural networks: A survey and call to action. *arXiv preprint arXiv:2407.14662*, 2024.

Yocum, J., Allen, C., Olshausen, B., and Russell, S. Neural manifold geometry encodes feature fields. In *NeurIPS 2025 Workshop on Symmetry and Geometry in Neural Representations*, 2025. URL https://openreview.net/forum?id=MwU86qfCTW.

A. Related Work

Tensor product representations. Tensor product representations (Smolensky, 1990) have influenced neural networks in numerous ways, often aimed at representing compositional or relational structure (Huang et al., 2018; Tang et al., 2018; Schlag & Schmidhuber, 2018; Park et al., 2024b). In the context of Transformers, TPRs-inspired architectures have improved performance on structured tasks such as mathematical tasks (Schlag et al., 2019) or abstractive summarization (Jiang et al., 2021). Perhaps closest in spirit to our work, (McCoy et al., 2018) show that RNN hidden-states can be reconstructed using Tensor Product Decomposition Networks.

Feature geometry. In recent years, a large body of interpretability work have found numerous concepts that are encoded as linear directions, and that these representations often generalize across models (Lee et al., 2025b). Examples include sentiment (Tigges et al., 2023), toxicity (Lee et al., 2024), refusal (Arditi et al., 2024), "correctness" (Lee et al., 2025a), and even user-attributes (Chen et al., 2024).

A growing line of work suggest that many features are not best described with rank-1 linear directions, but instead occupy low-dimensional manifolds or coordinates in a subspace. Examples include circular geometry for periodic concepts like days of the week (Engels et al., 2024), but also emotions (Sun et al., 2026), helical structure for number representations (Kantamneni & Tegmark, 2025), and manifold structure for dates and years (Modell et al., 2025). Others find features represented as coordinates in low-rank subspaces: (Park et al., 2024a) find that models can represent in-context learning tasks, while (Lee et al., 2026) identify low-rank feature interactions in attention mechanisms. (Gurnee et al., 2025) similarly show that features such as word count and token position can lie on manifolds that are aligned to produce high attention scores.

Taken together, these works suggest that linear probes may sometimes only recover local readouts of a richer underlying structure. A good example might be of (Bai et al., 2025), who study a toy Transformer trained on multi-digit multiplication. The model forms clusters to represent each digit, from which linear directions (i.e., a vector towards the centroids of each cluster) can decode a predicted digit from hidden-states. However, the clusters themselves form a highly intuitive structure by organizing into a pentagonal prism that reflects parity (even versus odd) and modulo-5 relationships. This example illustrates the broader possibility that linear directions may be projections of more structured, domain-specific representations.

Perhaps most relevant to our work are three recent lines of work. (Shai et al., 2026) find that Transformers can learn factorized representations of underlying latent variables of data generating processes. (Yocum et al., 2025) argue that neural networks may encode structured feature fields, which linear probes can recover from their activations. (Sarfati et al., 2026) build on this idea to recover a manifold of "posterior beliefs": by training a family of linear probes across different latent parameter settings of a controlled in-context learning task, and by "tiling" the linear probes together, they are able to recover a manifold over inferred latent parameter values, similarly suggesting that linear readouts stem from underlying structure.

Unsupervised structure discovery. Another related line of work asks how to uncover latent, unknown structure from activations at scale. Various geometry-aware sparse autoencoders have been proposed, based on various structural assumptions (Hindupur et al., 2025; Costa et al., 2025; Bhalla et al., 2025; Bussmann et al., 2025; Muchane et al., 2025). A complementary approach is to make no assumptions regarding the underlying states and to rely on generative models to infer the latent structures from the activations (Luo et al., 2026).

We envision future steps of structure discovery to be a mix of supervised, domain or concept-specific architectures such as our TPR probe as well as scalable, unsupervised, unrestrictive architectures to recover unknown structure.

B. Additional Results

Figure 8 shows the TPR probe accuracy per layer for a range of d_r values. Note that with enough dimensions, the TPR probe achieves the same accuracy as the linear probes, because the TPR probe effectively learns the same directions as the linear probes (see Section 4.2).

Figure 9 shows the analogous of Figure 4 but for the trilinear TPR probe instead of the bilinear.

C. Full-Dimensional TPR Reparameterizes Linear Probes

In Section 4.2 we see that TPR probes with enough dimensions recover "effective linear directions" that closely align with independently trained linear probes. This occurs for the trilinear TPR probe as well (Figure 9), for a model with $d_u = d_v = 8, d_f = 2$ dimensions. Here we explain why this occurs in the full-dimensional setting.

Consider first the trilinear TPR probe,

$$\mathbf{T} := \mathbf{M}(\mathbf{h}) \in \mathbb{R}^{d_u \times d_v \times d_f}, \qquad \ell_{ij,c} = \langle \mathbf{T}, \mathbf{u}_i \otimes \mathbf{v}_j \otimes \mathbf{f}_c \rangle,$$

where $\mathbf{u}_i \in \mathbb{R}^{d_u}$ and $\mathbf{v}_j \in \mathbb{R}^{d_v}$ are the row and column embeddings, and $\mathbf{f}_c \in \mathbb{R}^{d_f}$ is the filler embedding for color c. The corresponding effective linear probe direction is

$$\widetilde{w}_{ij,c} = \mathbf{M}_{\text{flat}}^{\top} \text{vec}(\mathbf{u}_i \otimes \mathbf{v}_j \otimes \mathbf{f}_c) \in \mathbb{R}^{d_{\text{model}}}.$$

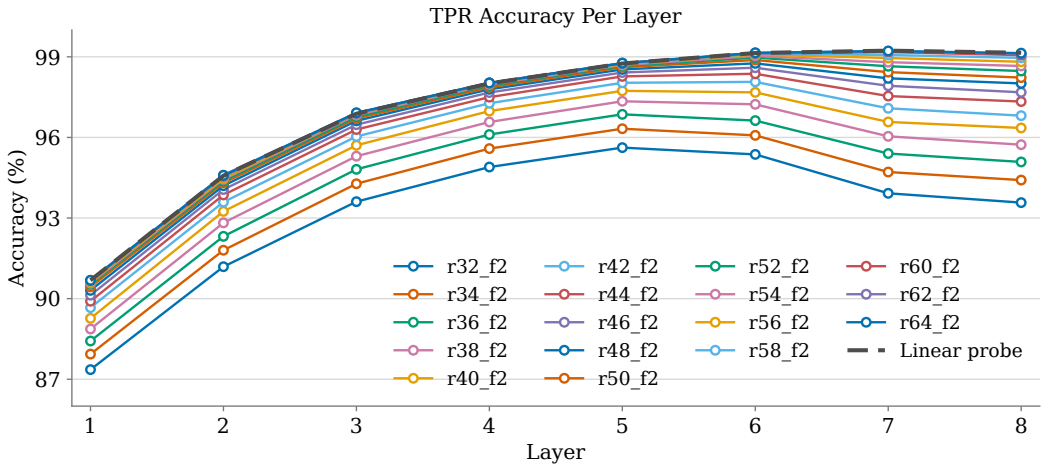

Figure 8. **TPR probe accuracy per layer.**

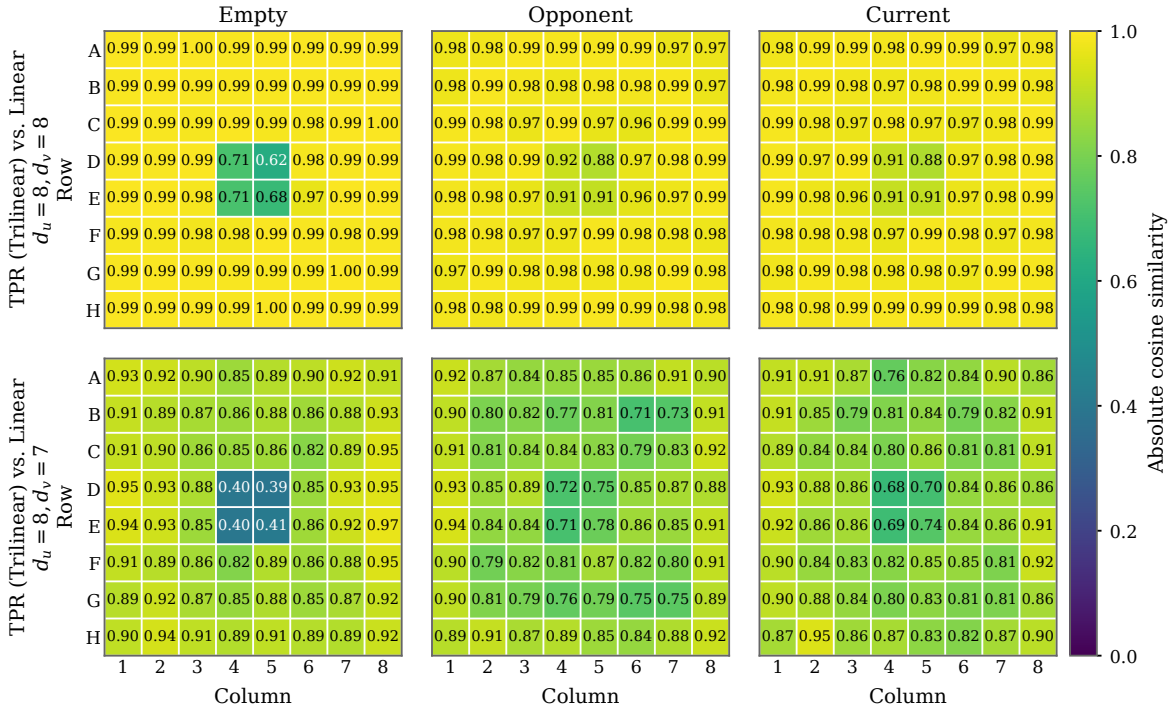

Figure 9. **Cosine similarity scores between linear probes vs. "effective linear probes" from trilinear TPR probes.**

When $d_u = d_v = 8$, the row and column embeddings have enough capacity to index all rows and columns of the board independently. In the simplest case, $\mathbf{U}$ and $\mathbf{V}$ could be the standard basis matrices. Then $\mathbf{u}_i \otimes \mathbf{v}_j$ selects the (i, j)-th slice of the binding tensor, so each board square is assigned its own d_f-dimensional latent vector.

Thus $d_u = d_v = 8, d_f = 2$ is sufficient to represent 64 three-way classifiers i.e., the 192 linear probe directions. Although each square has three possible labels, $\mathcal{C} = \{\text{EMPTY}, \text{CURRENT}, \text{OPPONENT}\}$, our TPR probes use $d_f = 2$. This is sufficient because a three-way softmax has only two identifiable degrees of freedom. Adding the same scalar offset to all three logits does not change the predicted probabilities, thus for each square, the probe only needs to represent two independent directions to represent three classes. This idea is analogous to representing K classes by a $(K-1)$-dimensional simplex. In the learned filler embeddings, PCA reveals exactly this structure: one axis separates EMPTY from occupied squares, while the other separates CURRENT from OPPONENT. To summarize, a trilinear probe with $d_u = d_v = 8, d_f = 2$ can represent the same 64 three-way classifiers (linear probes), and the same argument can be applied for bilinear TPR probes with $d_r = 64, d_f = 2$.

One thing worth noting is that TPR factorizations are not unique. Transformations applied to one component (row, column, or filler embeddings) can be compensated by another ($\mathbf{M}$) while leaving the resulting logits unchanged. For intuition, suppose the row and column embeddings adopt canonical standard basis vectors.

In fact, we empirically observe that row and column embeddings form "effective" standard basis vectors. In the full-dimensional trilinear probe, the recovered row embeddings form an approximately orthogonal basis. To quantify this, we row-normalize $\mathbf{U}$ and compute the Gram matrix $\mathbf{G}_\mathbf{U} = \widetilde{\mathbf{U}}\widetilde{\mathbf{U}}^\top \in \mathbb{R}^{8\times 8}$, where each entry gives the cosine similarity between a pair of row embeddings. Figure 10(a) shows that $\mathbf{G}_\mathbf{U}$ is close to the identity matrix, and Figure 10(b) shows that the singular values of $\mathbf{U}$ are all close to one. Together, these indicate that the learned row embeddings behave like an effective orthonormal basis.

Note that this behavior does not occur with incomplete dimensions. When $d_u < 8$, the probe can no longer assign mutually orthogonal basis vectors to all eight rows. Instead, the row identities must be represented in superposition. Figure 10(c) illustrates the row-normalized Gram matrix for this setting, in which we no longer observe the identity matrix.

We observe the same patterns for column embeddings in Figure 11.

In summary, in the full-dimensional regime, TPR probes can recover the same effective linear directions as standard linear probes because they can allocate independent basis elements to each board position, essentially reparameterizing the linear probes. In incomplete-dimensional regimes, the TPR probe must compress the board-state representation through a structured bottleneck, leading to a structural distributed code.

Table 1. Hyperparameters for probes.

Learning rate	1e-2
Weight decay	1e-2
Batch size	128
Num epochs	1
Validation patience	10
Validate every	100
Train size	295,699
Validation size	512
Test size	1,000

D. Training Details

Table 1 provides our hyperparameters for training linear and TPR probes. All of our experiments are conducted on a single Nvidia H100 80GB GPU, but significantly less memory (16GB) will likely suffice.

E. Societal Impact

Our work takes a step towards better understanding and interpreting the internal representations of language models. We hope a better understanding will lead to safer and more reliable use cases of models in the future.

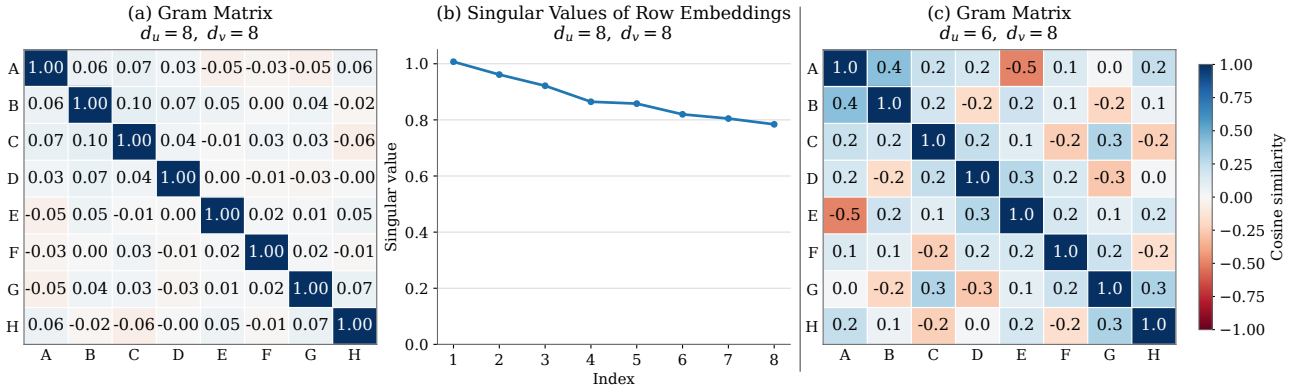

Figure 10. **In a "full-dimensional" case, row embeddings effectively behave like an effective orthonormal basis.** Given row-embeddings $\mathbf{U} \in \mathbb{R}^{8 \times d_u}$, with enough dimensions ($d_u = 8$) the row-normalized Gram matrix $\mathbf{U}\mathbf{U}^\top$ shows that the rows of $\mathbf{U}$ form an orthogonal set of basis vectors to encode each row of the board. The singular values of $\mathbf{U}$ are all close to 1, confirming that $\mathbf{U}$ behaves like an effective orthonormal basis. With fewer dimensions ($d_u < 8$), we instead observe a distributed encoding (i.e., superposition) of the 8 rows of the board.

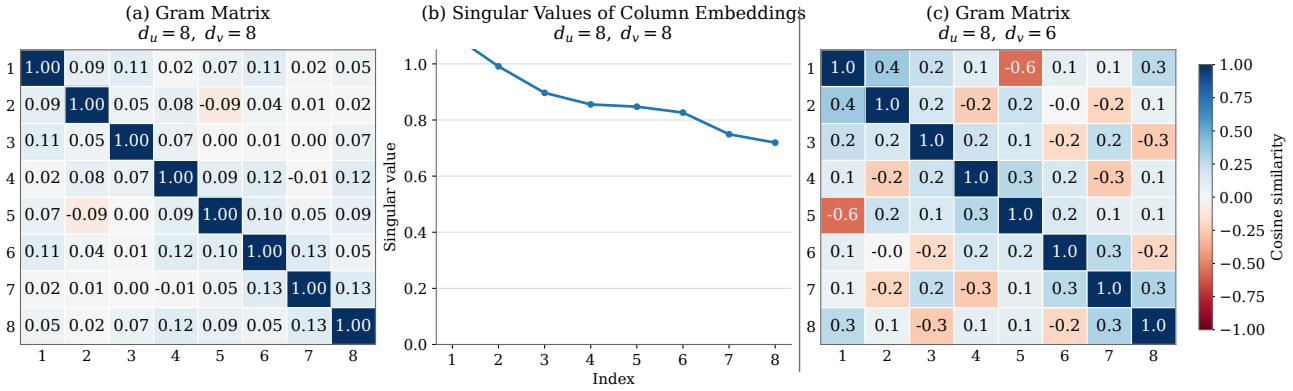

Figure 11. **Gram matrix and singular values of column embeddings.**