Interpretability of OthelloGPT

Interpretierung von OthelloGPT

Jim Maar

Universitätsbachelorarbeit
zur Erlangung des akademischen Grades

Bachelor of Science
(B. Sc.)

im Studiengang
IT Systems Engineering

eingereicht am 20. October 2024 am
Fachgebiet Algorithm Engineering der
Digital-Engineering-Fakultät
der Universität Potsdam

Gutachter	Prof. Dr. Gerard De Melo
Betreuer	Prof. Dr. Gerard De Melo

Abstract

Transformer models are rapidly used more and more in. But the internal computation of these modelels is barely understood. Recent work on OthelloGPT a toy language model trained to produce legal moves in the Board game Othello, has revealed insight into the behaviour of transformel model, such as thathey can havet a world model. In this work we try to reverse engineer the computation of the world model in othelloGPT. In particular, we show that the attention layers of OthelloGPT can be approximated by what we call the last flipped heuristic. Using Decision Trees, we also show that a lot of neurons in the MLP layer, can be described in terms of human understandable rules, making use of the world model. These understandable Neurons can be put into families of neurons with similiar behaviou. This understanding can be used to make testable predictions of the model behaviour. (This is a preliminary Document, The last part isn't fully finsihed and the language still needs to be refined)

Zusammenfassung

Acknowledgments

Contents

1 Introduction

How can we find out what Machine Learning Models do to achieve results (This should be really catchy). In recent years Transformer Models have made huge progress on NLP Tasks [Bub+23]. Yet our understanding of the models remains very limited with, with Machine Learning Models often viewed as Black Boxes. Mechanistic Interpretability is an emerging field that seeks to understand the internal reasoning processes of trained neural networks and gain insight into how and why they produce the outputs that they do [Has]. The field has made recent progress, but is a long way of from fully reverse engineering highly complex models. [BG24]. Previous Mechanistic Interpretability has focused on finding Cicuits in the attention layers [Elh+21] [Wan+22], while the mechanistic understanding of the MLP layer in Transfomer models remains limited. There are two popular ways to make progress in mechanistic interpretability. One is to pick a concrete question in language models, and set out to specifically answer that question e.g. [Wan+22] [Ols+22]. Another is To pick a toy model that's a good enough proxy for LLMs in general, and just try to get traction on reverse-engineering that model e.g. [Elh+21]. This work falls into the second category. We build upon [Li+24a] who train a Transfomer Model called OthelloGPT on the task of predicting legal moves in the board game Othello given the prior moves. They show that the model learns to track the correct board state using nonlinear probes. [NLW23] Show that the world model is actually encoded linearly. They also show that the model tracks which tiles are flipped and how the model calculates which tiles are empty. We do not yet have a complete understanding of how exactly the Model calculates the board state. This is the topic of this thesis. We start with an overview of the transformer architechture, the game othello, the model othelloGPT and the previous work on othelloGPT. Next we describe out early investigations into the model behaviour. After that we dive into the attention layer, and describe and test what we call the last-flipped-heuristic. Then we dive into the MLP layers identifying that neurons follow understandable rules and can be put into groups, where a group consists of neurons all producing very similar behavior, but on different parts of the board. Thruout the Thesis we will come up with hypothesis for what the model is doing, that end up in a sketch of how the model computes the board state.

Part I

Preliminaries

2 Othello and OthelloGPT

Othello

Othello is a two player board game played on a 8 x 8 board. Every move each player plays a tile of their color on the board. The Tile must be played, such that a steight line (horizontal vertical or diagonal) of the other players tiles is enclosed on both sides by two pieces of your color. Then the enclosed pieces get flipped to the other color. This is repeated until the every sqaaure on the board has been placed. The Person who has more tiles of their color wins the game. 2.1 shows the start of an example game.

OthelloGPT

OthelloGPT is an 8 Layer Decoder Only Transformer Model trained next token prediction on a dataset of randomly generated legal othello Games. Because the next move is a uniformly random legal move. The model learns to output roughly uniformly positive logits for all legal moves and negative logits for non legal moves. The model has an input vocabulary of 60, each corresponding to a single playable square in the game (e.g B4) . Each layer has a 512-dimensional hidden space, 8 attention heads and an mlp layer with 2048 neurons. The mlp layer uses a GELU [HG23] nonlinear activation function. The model was trained using attention and

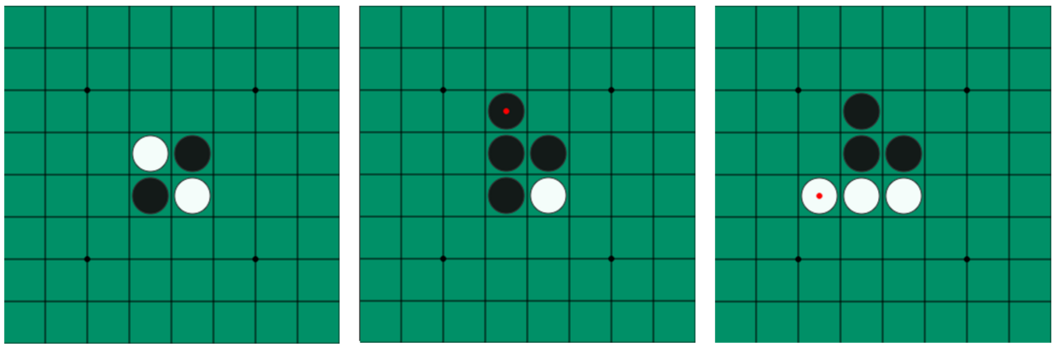

Figure 2.1: The start of an othello game

residual dropout. It’s important to note that OthelloGPT is not meant to be a useful model. It’s meant to be a testbed for interpretability [Li+24b] becuase it allows studying a model that uses the same architechture as modern LLMs, but might be less complicated to study.

3 The Transformer Architecture and Notation

We describe the architecture of the Transformer model. Since our focus is on interpreting the inner workings of a Transformer, we will avoid using the most compact notation or the one typically found in code implementations. Instead, we aim for a more intuitive and accessible notation, similar to the style used by [San24]. Additionally, we will define intermediate variables that may aid interpretability. Our approach is inspired by [Elh+21]. We begin by defining key terms:

- $D = 512$: The dimension of the residual stream.
- $d = 128$: The dimension used within each attention head.
- $m = 2048$: The number of neurons in the MLP layer.
- $T = 60$: The number of tokens, where each token represents a playable tile (Note: this includes a placeholder token for "passing").[1]
- $C = 60$: The context size, meaning each tile can be played exactly once.
- $W_E \in \mathbb{R}^{D\times T}$: The embedding matrix.
- $t \in \{1, \dots, T\}_{\text{one-hot encoded}}$: The input tokens, represented as a one-hot encoded vector.
- $\mathbf{x_i} \in \mathbb{R}^{D\times C}$: The residual stream at layer i.
- $x_i \in \mathbb{R}^{D}$: The residual stream at position i within a given layer.
- $x^i \in \mathbb{R}^{D}$: The residual stream at a given position in layer i.
- H_i: The set of attention heads in layer i.
- $b_i \in \mathbb{R}^{D}$: A bias term after the attention layer
- MLP_i: The MLP layer in layer i.

1 In cases where a tile cannot be played, a special token for passing is used, but we don't have any games with passing in our dataset

Each Transformer layer i consists of a self-attention layer and a multilayer perceptron (MLP) layer. Both layers take the residual stream as input and modify it by adding their outputs to the stream. Specifically, the residual stream is updated as follows:

$$\mathbf{x_0} = W_E t$$

$$\mathbf{x'_{i-1}} = \mathrm{LayerNorm}(\mathbf{x_{i-1}})$$

$$\mathbf{x}_{\text{i\_mid}} = x'_{i-1} + \mathrm{Att}_i(x'_{i-1})$$

$$\mathbf{x}'_{\text{i\_mid}} = \mathrm{LayerNorm}(\mathbf{x}_{\text{i\_mid}})$$

$$x_i = \mathrm{MLP}_i(x'_{\text{i-1\_mid}})$$

The final output of the Transformer is given by:

$$T(t) = W_U x_{-1}$$

LayerNorm

The Layernorm is defined in the following way: [2]

$$\mathrm{LayerNorm}(x_i) = \frac{x_i - \mathbb{E}(x_i)}{\sqrt{\mathbb{V}(x_i) + \epsilon}}$$

Self-Attention Operation

The core attention operation looks like this

$$\mathrm{Att}_i(x'_{i-1}) = \sum_{h \in H_i} h(x'_{i-1}) + b_i$$

2 Normaly there is also a weights and bias term. But we use the OthelloGPT model from Transformer Lens Libary [NB22] with folden in weights, meaning that some weights and bias terms are sligtly modified to incorporate the bias and weights of the layernorm. This can make some things easier.

For each attention operation, there are four key matrices:

- $W_Q \in \mathbb{R}^{D\times d}$: The query matrix.
- $W_K \in \mathbb{R}^{D\times d}$: The key matrix.
- $W_V \in \mathbb{R}^{D\times d}$: The value matrix.
- $W_O \in \mathbb{R}^{D\times d}$: The output matrix.
- M: a lower tringluar matrix of $-\infty$

The attention pattern A is computed as:

$$A = \text{softmax}(\mathbf{x}^\top W_Q W_K^\top \mathbf{x} + M)$$

Here, $A_{i,j}$ represents the attention paid by token i to token j.
Now, the output of an attention head $h(x)_i$ is calculated as:

$$h(x)_i = \sum_{j\leq i} A_{i,j} \cdot (W_O W_V x_j)$$

To understand what some attention heads do in a specific case it's useful to conceptualize the general function of attention heads. "The fundamental action of attention heads is moving information. They read information from the residual stream of one token, and write it to the residual stream of another token" [Elh+21]. In the mechanistic interpretability literature, the operation $x_i^\top W_Q W_K^\top x_i$ is refered to as the "QK Circuit". It describes when the attention head moves information from one position to another. Similiarly $W_O W_V x_j$ is referred to as the "OV circuit," what information is read from position j and how it is written to the attending position. [Elh+21]

MLP Layer

In each layer i, the MLP has the following weights:

- $W_{\text{in}} \in \mathbb{R}^{m\times D}$: Input weight matrix.
- $b_{\text{in}} \in \mathbb{R}^{m}$: Input bias vector.
- $W_{\text{out}} \in \mathbb{R}^{D\times m}$: Output weight matrix.

- $b_{\text{out}} \in \mathbb{R}^m$: Output bias vector.

The MLP operation is computed as:

$$\text{MLP}(x_i) = b_{\text{out}} + W_{\text{out}} \cdot \text{GELU}(b + W_{\text{in}} \cdot x_i)$$

We can rewrite theis operation to be described in terms of neurons.

$$\text{MLP}(x_i) = b_{\text{out}} + \sum_{j=0}^{m} W_{\text{out}}[j,:] \cdot \text{GELU}(b_j + W_{\text{in}}[:,j] \cdot x_i)$$

Here we are using the Following Definitions:

- GELU: The GELU nonlinear activation function
- $W_{\text{in}}[j,:] \in \mathbb{R}^D$: The j'th row of W_{in} and the input weights of neuron j.
- $b_{\text{in}\,j} \in \mathbb{R}$: The bias of neuron j.
- $W_{\text{out}}[:,j] \in \mathbb{R}^D$: The j'th column of W_{out} and the output weights of neuron j.
- $\text{GELU}(b_j + W_{\text{in}}[:,j] \cdot x_i) \in \mathbb{R}$: the activation of neuron j.

4 Mechanistic Interpretability

The current paradigm in the mechanistic interpretability literature tries to understand Neural Networks in terms of features and circuits.[BG24]

- Feature: "The fundamental units of how neural networks encode knowledge, which cannot be further decomposed into smaller, distinct concepts" [BG24]
- Circuit: "Sub-graphs within neural networks consisting of features and the weights connecting them. Circuits can be thought of as computational primitives that perform understandable operations to produce (ideally interpretable) features from prior (ideally interpretable) features."

The Linearity Hypothesis [Elh+22] states that features correspond to directions in space.

I'd like to introduce some specific language, to make things more understandable / less unclear

- Feature Name: The name of a feature e.g. "C3 is Empty"
- Fature Direction: The corresponding direction
- Fature Variable: the boolean variable of whether the feature is present in a given residual stream x_i. This is the case if the residual stream has a large dot-product with the feature direction $F \cdot x_i \gg 0$.

Part II

Findings

5 Training Linear Probes

This work builds upon [NLW23]. They showed that the OthelloGPT model has a linear representation of the board state using linear probes. Their linear probe p is a simple linear projection from the residual stream: $p(x_t^l) = \text{Softmax}(Wx_t^t), W \in \mathbb{R}^{D\times 3}$. Such a probe is trained for each tile and they use it to classify the given Tile as EMPTY, MINE or YOURS depending on the turn of the player. Let's see this on an example. Let's say the Square D3 has a black tile over the first few moves. In move 1 it's the white players turn. so D3 should be classified as YOURS. then in turn 2 it's the black players turn, so D3 should be classified as MINE. The linear probe can be seen as 3 Features corresponding to 3 directions in D-dimensional space ($W[:,0]$, $W[:,1]$ and $W[:,2]$). If for example $W[:,2]$ has a high dotproduct with the residual stream at some position $x_i \cdot W[:,2] \gg 0$, then the Feature "D3 is Yours" is present.

They these linear probes on 3,500,000 games for each layer and tile. The accuracy goes from 90.9 percent in layer 0 to 99.6 percent in layer 6. They also use the trained linear probes to succesfully intervene on the model, e.g. changing the color of a tile in the representation of the board state, leading an expected change in what the model predicts are the legal moves. This shows that the model is actually causaly using the found internal represantation of the board state. They also succesfully train another linear probe on classifying whether a tile has been flipped from one Color to another this turn with similiarly good results.

These Probes where trained on the residual stream after each transformer layer. For later analysis of the MLP Layer we train these probes on the residual stream after the attention Layer and before the MLP layer. As seen in 5.1 and 5.2 we get similiar results as [NLW23].

Module	L0	L1	L2	L3	L4	L5	L6	L7
Post	0.908	0.947	0.971	0.982	0.989	0.993	0.994	0.993
Mid	0.888	0.943	0.968	0.980	0.989	0.993	0.995	0.994

Table 5.1: Accuracy of linear board state probe over differend Layers. Post means after MLP and Min means before MLP

The Flipped Probe might as to how the model computes the board state, for example it might be used in the following way:

Module	L0	L1	L2	L3	L4	L5	L6	L7
post	0.870	0.926	0.957	0.973	0.982	0.985	0.984	0.981
mid	0.801	0.895	0.934	0.964	0.978	0.985	0.986	0.984

Table 5.2: F1 Score of flipped probe over differend Layers. Post means after MLP and Min means before MLP

Hypothesis 1. *For each tile the model saves the information of which color was first played there and it also gathers the information of how many times a specific tile has been flipped. Using this information the current color of the tile is easy to compute.*

If the hypothesis where true, we would expect to find a representation of the color of the first tile, as well has the number of flips of each tile. We test this hypothesis by training linear probes on the number of times a tile has been flipped and on the first color that has been placed there, but do not find convincing results. As seen in 5.3, the results are not nearly as good as the other probes. So 1 is not true.

probe name	L0	L1	L2	L3	L4	L5	L6	L7
Num Flipped	0.145	0.151	0.152	0.154	0.155	0.160	0.161	0.160
Num Flipped Even/Odd	0.714	0.727	0.742	0.748	0.752	0.761	0.774	0.803
First Tile Black/White	0.145	0.151	0.152	0.154	0.155	0.160	0.161	0.160
First tile Mine/Yours	0.825	0.805	0.817	0.795	0.801	0.797	0.801	0.796

Table 5.3: Accuracy of Different Probes over all layers

[He+24] used Sparse Autoencoders on a smaller OthelloGPT model that they trained. These found Features of the form Some Tile A, flipping a neighbouring Tile B. We did not find such directions by training a linear probe in this model though 5.4.

Module	L0	L1	L2	L3	L4	L5	L6	L7
post	0.213	0.175	0.140	0.119	0.101	0.085	0.081	0.065
mid	0.137	0.166	0.137	0.113	0.101	0.097	0.081	0.076

Table 5.4: F1 Score of the placed and flipped probe

[jyl+] found the model first finds which tiles are accesible (neugbouring a non-empty tile), before computing if the tile is also legal. We confirm that the model has a representation of which tiles are accesible in 5.5

Module	L0	L1	L2	L3	L4	L5	L6	L7
Accesible post	0.9939	0.9945	0.9970	0.9991	0.9992	0.9932	0.9738	0.9675
Accesible mid	0.9996	0.9964	0.9964	0.9981	0.9994	0.9994	0.9931	0.9714
Legal post	0.8658	0.8968	0.9114	0.9195	0.9503	0.9835	0.9846	0.9831
Legal mid	0.8535	0.8907	0.9076	0.9149	0.9213	0.9516	0.9838	0.9841

Table 5.5: F1 Score of the accesible probe and legal probe

For later use we also succesfully train Probes to Detect if a tile has been placed 5.6.

Module	L0	L1	L2	L3	L4	L5	L6	L7
post	1.0000	0.9999	1.0000	0.9999	0.9999	0.9998	0.9992	0.9978
mid	1.0000	1.0000	1.0000	0.9999	1.0000	0.9998	0.9997	0.9993

Table 5.6: F1 Score of the placed probe

All probes are trained on 300000 games. using the Adam Optimizer with Hyperparameters

- Learning Rate : 0.01
- Batchsize : 256
- weight decay : 0.1
- β_1 : 0.9, β_2 : 0.99

Looking at the accuracy of the board state probe 5.1 we seee that each layer can calculate a pretty much perfect board representation until some position n_l where the accuracy starts to drop off.

We now have a collection of useful features e.g. "C3 is Empty", "C3 is Mine", "C3 is Yours", "C3 is Flipped", "C3 is placed", "C3 is accesible" and "C3 is Legal", that we can use later on. Taking the residual stream and appling all the probes on it lets us see which features are active and which aren't. We call the collection of probe results, the world model of OthelloGPT.

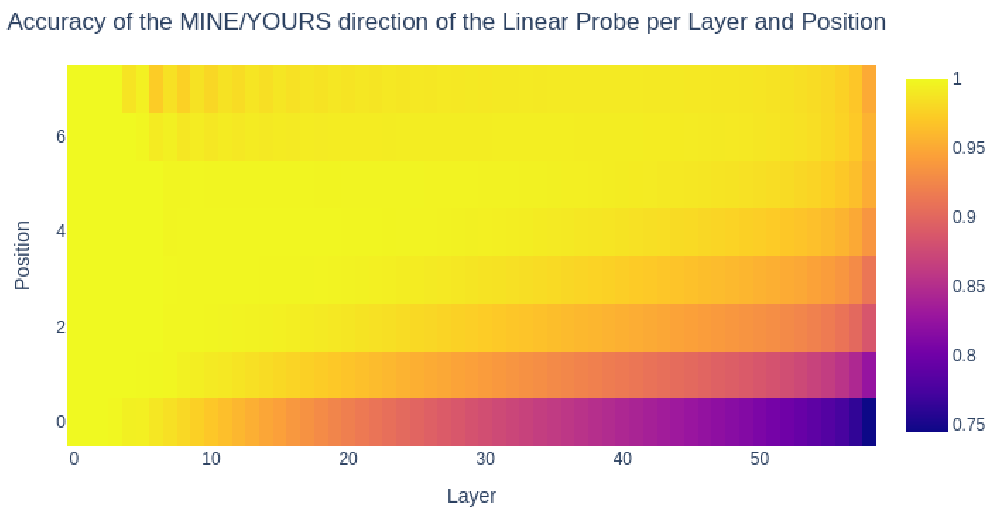

Figure 5.1: Accuracy of the Linear Probe over each layer and sequence position

6 The Attention Layer

Basic Observations

Previous Mechanistic Interpretability work has focused on interpreting the attention mechanism A Previous Paper [HZC23], investigating OthelloGPT, categorize the attention heads into 5 different groups, depending on their typical Attention pattern.

- "Last" Heads: primarily attend to the last sequence position
- "First" Heads: primarily attend to the first sequence position
- "Mine" Heads: primarily attention to the sequence-positions that are an even number of steps away. So it pays attention to the previous times when the current player was taking their turn.
- "Yours" Heads: primarily attention to the sequence-positions that are an odd number of steps away. So it pays attention to the previous times when the other player was taking their turn.
- Other

6.1 shows the distribution of Mine Heads and Their Heads over the different Layers.

Looking at the Mine/There Heads they seem to pay less attention to moves farther away as shown in 6.2 on an example. We compute the empricial mean and variance of all non-zero entries of the attention pattern of every attention head from 200 input sequences each. The Attention patterns have an average empirical varince of 0.0007, suggesting that they are mostly influenced by the sequence position [3]. This makes understanding the OV-Circuit more interesting than understanding the QK-Circuit.

3 I tested if only inputing the positional embedding would produce the same attention pattern, but it did not

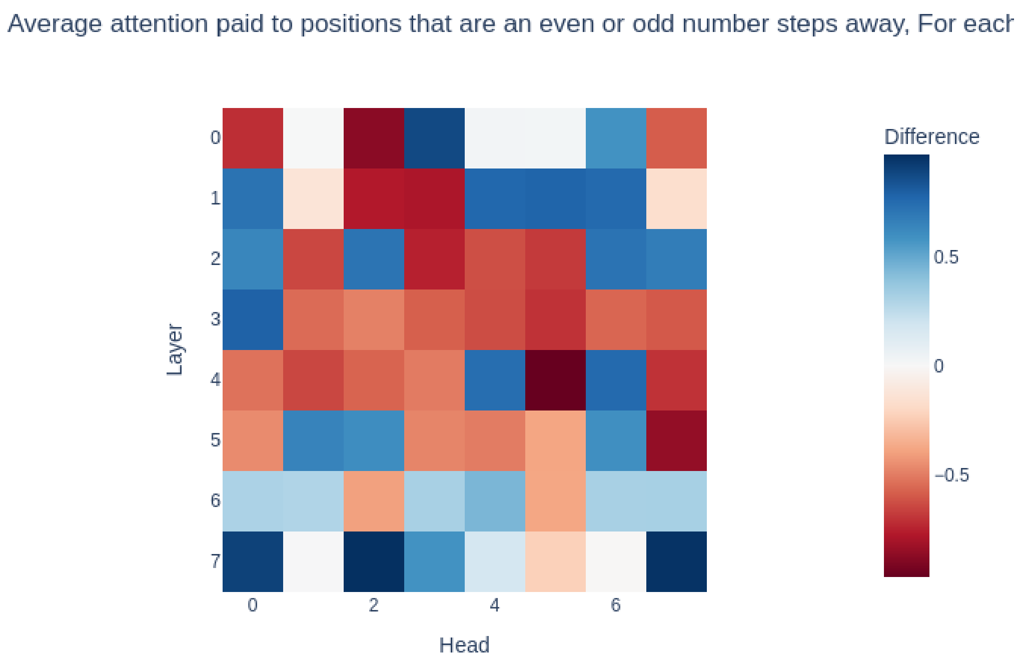

Figure 6.1: Average attention paid to positions that are an even steps away minus attention paid to postitions that are an odd number of steps away, For each Head and Layer. Mine Heads are Blue and Red Heads are Mine. "Last", "First", and "Other" Heads are white. L4H5 is a "Last" Head for example

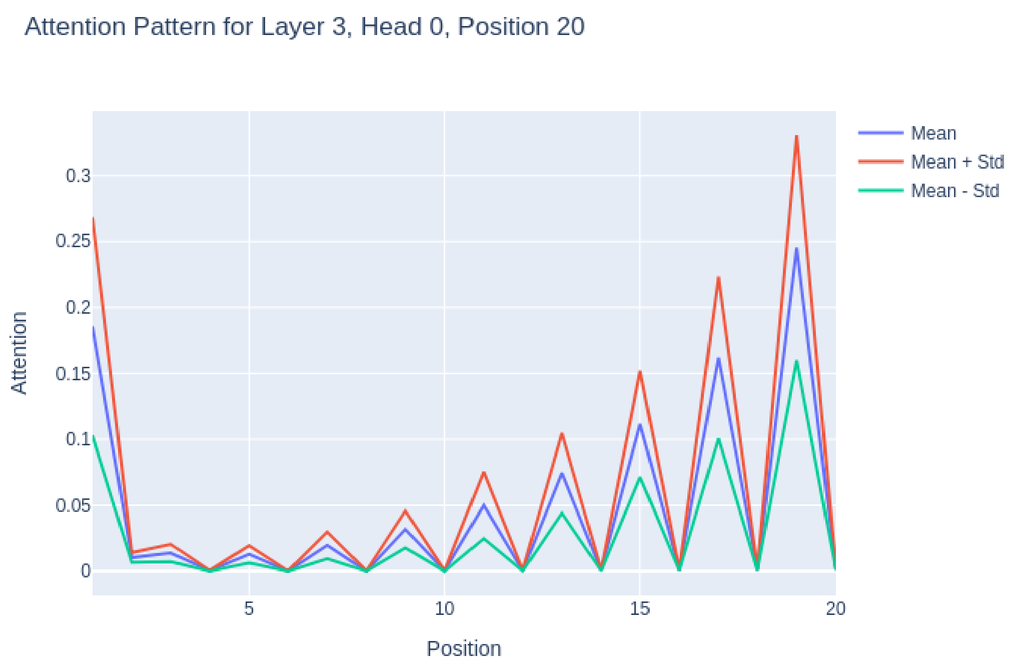

Figure 6.2: Attention Pattern for Layer 3, Head 0, Position 20)

Interface

To get more ideas of how the model computes the board state we created a visulization of the boardstate representation over the complete input sequence and all layers 6.3.

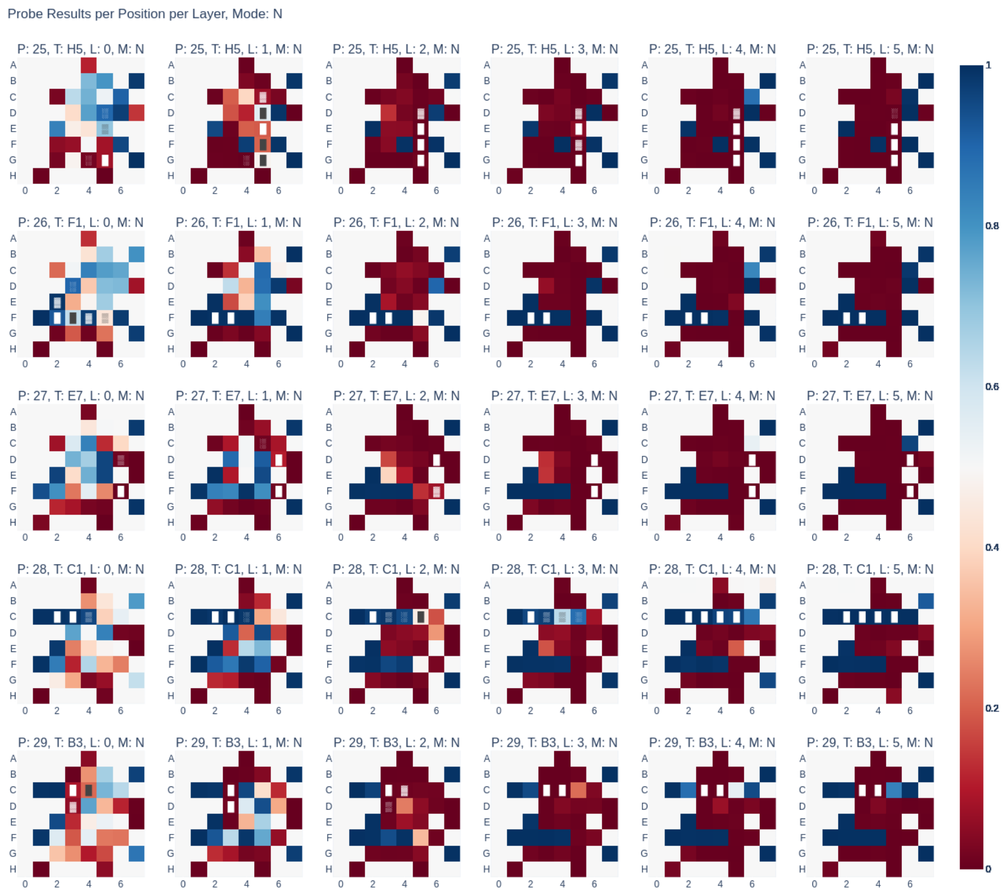

Figure 6.3: An example of the interface used to analyze the computation of the world model over the layers and positions

We can see some interesting things. The model seems to gradually build up the correct board state over the layers, and use the information of previous layers to fix inconsistencies. In the example 6.3 At the sequence position 26 in layer 1 the tiles D5, E5 and F5 are Black/Mine in the board representation (Blue in the Visualization), in Layer 2 they are updated, to become White/Their (Red in the Visualization). The reason might be that these tiles where flipped at sequence position 25 and this information was present in the boardstate representation at layer 1. So after layer 1 this information can be moved to sequence position 26 via the attention layer.

Direct Logit Attribution

Direct Logit Attribution is a basic interpretability technique, with the purpose of finding which part of the model contributed to some behaviour of interest. We can use this to confirm our theory on the example.

Let $\mu : \mathbb{R}^D \rightarrow \mathbb{R}$ be the Feature direction of "D5 is mine". To see which part of the model wrote this feature to the residual stream, we can split the residual stream into the contributions of the other layers

Let x^i be the residual stream at position 26 in layer i.

$$\mu(x^2) = \mu(x^1) + \mu(\sum_{h \in H_2} h(\mathbf{x}^{\mathbf{1}'})) + \mu(\mathrm{MLP}_2(x^{1\_\mathrm{mid}'}))$$

The contributions of the previous redidual stream, the attention layer and the mlp layer are 1.71, 4.21 and 0.61 respectively. We can further split contributions of the attention layer into the contributions of each head and sequence position, visualized in 6.4.We see that the biggest contribution comes from position 25 thrue the "yours" head L2H3, which is what we expected.

The Last-Flipped-Hypothesis/Approximation

similiar behaviour can be found in other places. a tile that was just flipped is always yours. So to move this information, the attention layer needs to write that the tile is Yours if it was flipped an even number of moves ago and Mine if the tile was flipped an odd number of moves ago. We formulate this behaviour in 2.

Hypothesis 2. *For each tile, the attention Layer predicts the color that the tile had when it was last flipped. Concreately: When a tile was flipped an odd number of moves ago, mine heads write "tile is mine" to the residual stream and when it was flipped an even number of moves ago, yours heads write "tile is yours" to the residual stream. Because the attention paid decreases with the distance in the sequence, the most recent flip dominates.*

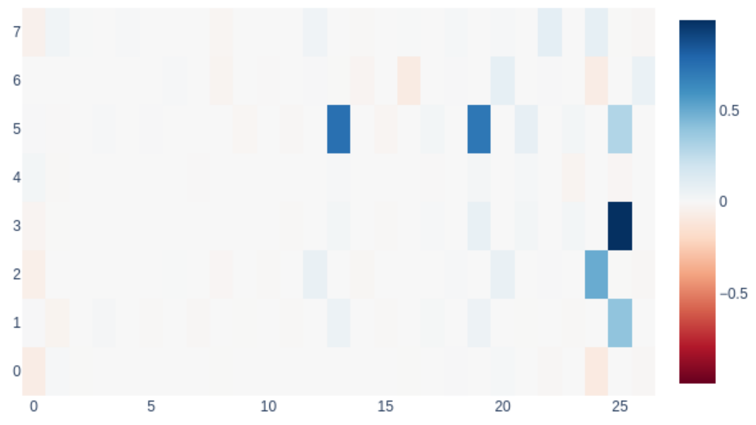

Figure 6.4: The direct logit attribution to "D5 is mine" in layer 2, position 26 or each attention head and sequence position

The OV Circuit

We test whether a tile being flipped leads to "yours" heads writing "tile is yours" and "mine" heads writing "tile is mine". For some specific Tile, let μ_i, ϕ_i and ψ_i be the feature directions for "tile is flipped", "tile is yours" and "tile is mine" respectively, in layer i. μ_i and ϕ_i have some overlap, so let $\mu_i' = \mu_i - \frac{\mu_i \cdot \phi_i}{\phi \cdot \phi} \cdot \phi_i$. μ_i' is the feature direction for "tile is flipped", but orthogonal to feature direction "tile is yours". Let W_O and W_V be the output and value matrix of some attention Head in layer i. Now $\mu_i * = (W_O W_V \mu_{i-1}')$ is the direction that the attention head writes to the residual stream, given the tile being flipped at a previous position. Now the attention head writes "tile is yours" if cosine_similiarity$(\mu_i *, \phi - \psi) \gg 0$ and "tile is yours" if cosine_similiarity$(\mu_i *, \phi - \psi) \ll 0$. 6.5 shows the results for all heads and tiles from layer 1 to 5. We can see that "mine" heads write to the "tile is mine" direction and "yours" heads write to the "tile is yours" direction. different attention heads seem to concentrate on specific regions of the board.

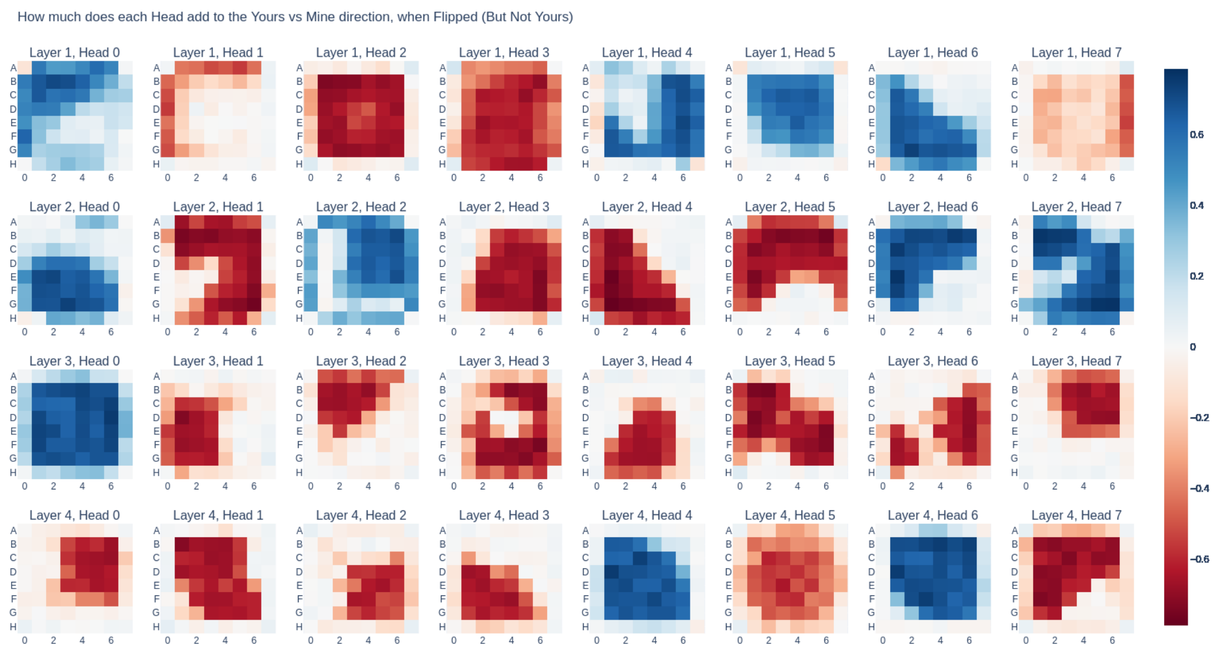

Figure 6.5: How much does each Head add to the "Tile is Yours" - "Tile is Mine" direction, when Flipped

Quantifiying the Last Flipped circuit

It's not yet clear, how much this This theory actually explains. To do that we propose an approximation for the attention layer.

Given a redisual stream x_i, we can split it up into a feature component, here "tile is flipped" for some tile, and a remainder. with $x_i = \mu_i^* + x_i^*$, with $\mu_i^* = \frac{\mu_i \cdot x_i}{\mu_i \cdot \mu_i} \cdot \mu_i$ and $x_i^* = x_i - \mu_i^*$, where x_i^* is othogonal to μ_i.

To test out theory, that the attention layer writes the color that the tile had, when it was last flipped, we assume that the remainder has little influence on the actuall result. We approximate it using the average value of the residual stream calculated using 200 input sequences without the flipped direction $\mathbf{x}_{\text{avg}} = \frac{1}{200} \cdot \sum_{g=0}^{200} \mathbf{x}^{\mathbf{g}} - \frac{\mathbf{x}^{\mathbf{g}} \cdot \mu}{\mu \cdot \mu} \cdot \mu$. Where $\mathbf{x}^{\mathbf{g}}$ is the residual stream in the g'th input sequence. We can split the attention layer into the contributions of each head and sequence positions.

$$\begin{aligned}
\phi(\text{Att}_i(x'_{i-1})) &= \sum_{h \in H_i} \phi(h(x'_{i-1})) + \phi(b_i) \\
&= \sum_{h \in H_i} \sum_{j \leq i} h.A_{i,j} \cdot \phi_i(h.W_O h.W_V x'_j) + \phi_i(b_i) \\
&= \sum_{h \in H_i} \sum_{j \leq i} h.A_{i,j} \cdot \phi_i((h.W_O h.W_V x^*_j) + \phi_i(h.W_O h.W_V \mu^*_j)) + \phi_i(b_i) \\
&\approx \sum_{h \in H_i} \sum_{j \leq i} h.A_{i,j} \cdot (\phi_i(h.W_O h.W_V x_{\text{avg}}) + \phi_i(h.W_O h.W_V \mu^*_j)) + \phi(b_i)
\end{aligned}$$

This model predicts approximates the logit of "tile is yours" in layer i. Without loss of generality, we can do the same for the "tile is mine" logit. As noted earlier we can also approximate the Attention pattern A with an average approximation $A_{\text{avg}} = \frac{1}{200} \cdot \sum_{g=0}^{200} A$.

When our approximation gives a higher logit for "tile is yours" then "tile is mine", we say that it predicts "tile is yours" and vice versa. We compare this with the real values for all non-empty tiles and take the accuracy. 6.6 shows the accuracy our approximation for each tile. The average accuracy is 0.764. Using the real attention pattern instead of the average attention pattern gives an accuracy of 0.768.

This shows that our approximation is meaningful, but is less than I expected. It would be interesting to see where and how exactly the approximation diverges from the real model.

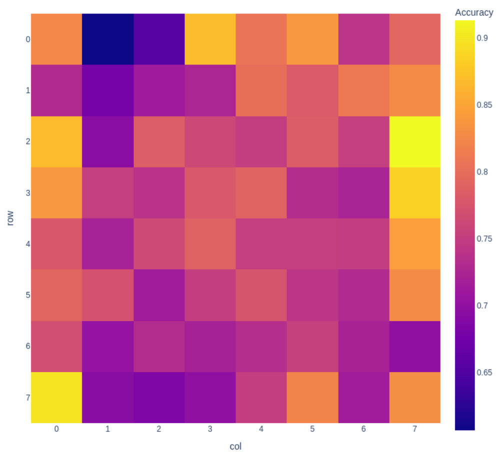

Figure 6.6: Accuracy of the Last-Flipped Heuristic of all Board Tiles over 10000 input sequences. We take the maximum accuracy over the layers and average over sequence positions

7 The MLP Layer

We also investigated the behavior of Neurons, because previous exploration in [Nan] revealed that investigating Neurons in OthelloGPT seemed fruitfull.

We project the Neuron Input Weights chapter 3 to the different probe directions and visualize the result. 7.1 shows an example on the Neuron L2N877 (Neuron 877 in Layer 2 of the model), which seems to activate when Tiles are getting flipped in column 4 downwards up until E4. Max Activating Dataset examples seem to confirm this 7.2.

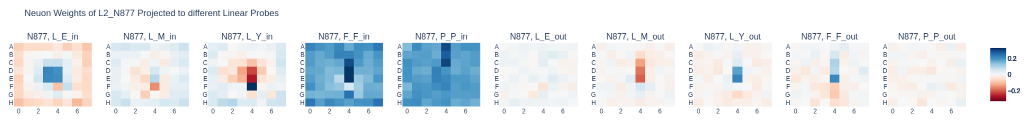

Figure 7.1: Example result of projecting neuron input and output weights to the different probes

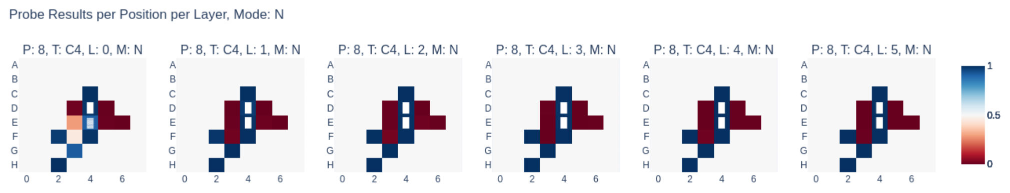

Figure 7.2: Max activating dataset example from 200 games of Neuron L2N877

Looking at this Visualizations seems to give the idea that many neurons follow human understandable rules, such as activate, when tile in a specific column get flipped.[jyl+] came up with the same idea.

Hypothesis 3. *A majority of Neurons in OthelloGPT follow rules that can be expressed as a boolean function of feature variables. e.g. Neuron L2N877 activates when ("F4 is Yours" AND "E4 is Mine" AND "F4 is flipped" AND E4 is flipped")*

To test this hypothesis we need to find rules and see how well they can approximate the actuall acitvations. We automatically find rules using decision trees. Decision Trees are great for this problem, the learned rules can be transformed into DNF format, which is exactly what we want.

For 100.000 residual streams we apply all probes and take the argmax to get a boolean value of 1 or 0 for each Feature as input variables, for the output variables we round neuron activations below 0 to 0 and above 0 to 1. For each layer and neurons we create such a dataset and then train a classification decision tree, we turn the learned rules into a DNF format for better readability. We use a constraint on the minumum inpurity decrease of each split, to reduce the size of the learned rules and therefore reduce overfitting and increase understandability.

7.3, 7.1 and 7.2 show the results

layer	f1	f1_train	weighted_f1	variable_count	fraction_0
0	0.66	0.67	0.76	27	0.09
1	0.59	0.59	0.69	34	0.1
2	0.61	0.61	0.7	33	0.15
3	0.65	0.65	0.73	29	0.21
4	0.73	0.73	0.81	26	0.19
5	0.76	0.77	0.86	28	0.15
6	0.78	0.78	0.86	24	0.34
7	0.76	0.76	0.79	14	0.75

Table 7.1: Results of the Deciion Trees per layer with min_impurity_decrease of 0.0005

layer	f1	f1_train	weighted_f1	variable_count	fraction_0
0	0.73	0.75	0.83	93	0.01
1	0.66	0.68	0.77	148	0.01
2	0.66	0.68	0.76	139	0.02
3	0.67	0.68	0.76	121	0.04
4	0.73	0.74	0.81	97	0.04
5	0.76	0.77	0.85	98	0.05
6	0.72	0.74	0.8	82	0.22
7	0.76	0.78	0.8	71	0.74

Table 7.2: Results of the Deciion Trees per layer min_impurity_decrease of 0.0001

For some neurons, no single split could be given the min_impurity_decrease constraint. They have an F1 Score of 0, some other neurons that activate very

layer	f1	f1_train	weighted_f1	variable_count	fraction_0
0	0.62	0.62	0.7	9	0.37
1	0.53	0.53	0.62	10	0.39
2	0.57	0.57	0.64	9	0.43
3	0.64	0.64	0.71	9	0.47
4	0.75	0.75	0.83	10	0.39
5	0.76	0.76	0.86	11	0.28
6	0.78	0.78	0.85	10	0.43
7	0.74	0.74	0.77	5	0.76

Table 7.3: Results of the Deciion Trees per layer min_impurity_decrease of 0.002

sparsely have learned rules but still have an f1 of 0 on the test set. The fraction of these neurons is depicted in the fraction fraction_0 column. they are not taken into account for the f1 score.

Higher Neuron activation have higher influence on the residual stream and model output. So correctly predicting the activation of higher neuron activations is more important than lower activations. This is why we compute the weighted f1 score, which is the same as f1 score, but each positive datapoint is weighed by how much it difers from the average positive neuron activation.

Neurons who's distribution of positive activations has a low skewness tend to have better rules. With a correlation of -0.46 between skewness and f1_score.

8 Conclusions & Outlook

Bibliography

[BG24] Leonard Bereska and Efstratios Gavves. **Mechanistic Interpretability for AI Safety–A Review**. *arXiv preprint arXiv:2404.14082* (2024) (see pages 1, 11).

[Bub+23] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. *Sparks of Artificial General Intelligence: Early experiments with GPT-4*. 2023. arXiv: 2303.12712 [cs.CL]. URL: https://arxiv.org/abs/2303.12712 (see page 1).

[Elh+21] 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. **A Mathematical Framework for Transformer Circuits**. *Transformer Circuits Thread* (2021). https://transformer-circuits.pub/2021/framework/index.html (see pages 1, 7, 9).

[Elh+22] 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. **Toy Models of Superposition**. *Transformer Circuits Thread* (2022) (see page 11).

[Has] Sarah Hastings-Woodhouse. *Introduction to Mechanistic Interpretability*. URL: %5Curl%7Bhttps://aisafetyfundamentals.com/blog/introduction-to-mechanistic-interpretability/#:~:text=Mechanistic%20Interpretability%20is%20an%20emerging,1%5D%7D (see page 1).

[He+24] Zhengfu He, Xuyang Ge, Qiong Tang, Tianxiang Sun, Qinyuan Cheng, and Xipeng Qiu. **Dictionary learning improves patch-free circuit discovery in mechanistic interpretability: A case study on othello-gpt**. *arXiv preprint arXiv:2402.12201* (2024) (see page 16).

[HG23] Dan Hendrycks and Kevin Gimpel. *Gaussian Error Linear Units (GELUs)*. 2023. arXiv: 1606.08415 [cs.LG]. URL: https://arxiv.org/abs/1606.08415 (see page 5).

[HZC23] Dean S Hazineh, Zechen Zhang, and Jeffery Chiu. **Linear Latent World Models in Simple Transformers: A Case Study on Othello-GPT**. *arXiv preprint arXiv:2310.07582* (2023) (see page 19).

[jyl+] jylin04, JackS, Adam Karvonen, and Can. *OthelloGPT learned a bag of heuristics*. https://www.alignmentforum.org/posts/gcpNuEZnxAPayaKBY/othellogpt-learned-a-bag-of-heuristics-1. Accessed: 2024-10-01 (see pages 16, 29).

[Li+24a] Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. *Emergent World Representations: Exploring a Sequence Model Trained on a Synthetic Task*. 2024. arXiv: 2210.13382 [cs.LG]. URL: https://arxiv.org/abs/2210.13382 (see page 1).

[Li+24b] Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. *Emergent World Representations: Exploring a Sequence Model Trained on a Synthetic Task*. 2024. arXiv: 2210.13382 [cs.LG]. URL: https://arxiv.org/abs/2210.13382 (see page 6).

[Nan] Neel Nanda. *Actually, Othello-GPT Has A Linear Emergent World Representation*. https://www.neelnanda.io/mechanistic-interpretability/othello. Accessed: 2024-09-27 (see page 29).

[NB22] Neel Nanda and Joseph Bloom. *TransformerLens*. https://github.com/TransformerLensOrg/TransformerLens. 2022 (see page 8).

[NLW23] Neel Nanda, Andrew Lee, and Martin Wattenberg. *Emergent Linear Representations in World Models of Self-Supervised Sequence Models*. 2023. arXiv: 2309.00941 [cs.LG]. URL: https://arxiv.org/abs/2309.00941 (see pages 1, 15).

[Ols+22] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. **In-context Learning and Induction Heads**. *Transformer Circuits Thread* (2022) (see page 1).

[San24] Grant Sanderson. *Attention in transformers, visually explained | Chapter 6, Deep Learning*. Apr. 2024. URL: https://www.youtube.com/watch?v=eMlx5fFNoYc (see page 7).

[Wan+22] Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. **Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 small**. *arXiv preprint arXiv:2211.00593* (2022) (see page 1).

Declaration of Authorship

I hereby declare that this thesis is my own unaided work. All direct or indirect sources used are acknowledged as references.

Potsdam, October 6, 2024

Jim Maar