# How OthelloGPT computes if a cell is mine or yours: a case study

In this note, I'll look into how OthelloGPT computes if non-blank cells are "mine" or "theirs" via a case study.

I'll start by looking at how it does this for a specific cell at a specific move in a specific game, then pull on the thread and see where it leads.

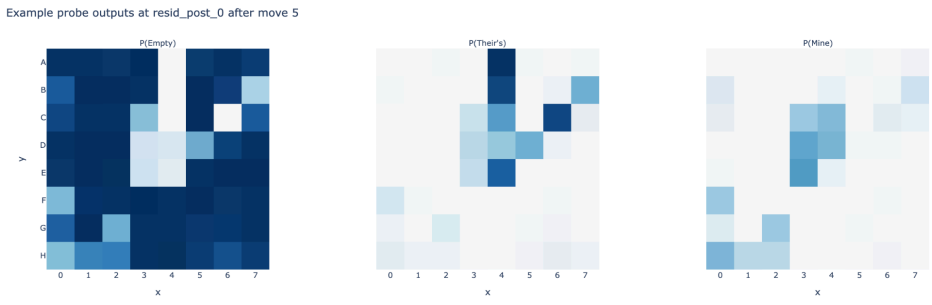
Ultimately, I'll argue that OthelloGPT implements a "mini-circuit" **"If A4 is played AND B4 is not blank AND C4 is not blank, update B4+C4+D4 towards "theirs in the world-model"** across all games, explaining the mechanism by which it implements the logic.

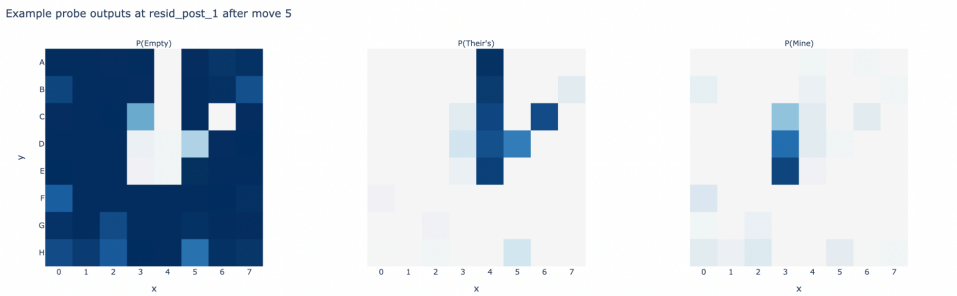
From this example, it's natural to hypothesize that OthelloGPT computes its world-model by aggregating the contributions of many qualitatively similar mini-circuits. However, I'll leave verification of this broader claim for future work.

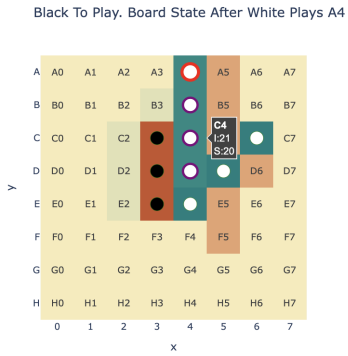
## C4 at move 5 in game 0 suggests a simple circuit

Since Transformers process information serially across model layers, I decided to start by finding a cell/move/game where OthelloGPT "changes its mind" about whether the cell is "mine" or "theirs" over the course of different layers.

Move index 5 in game 0 is the earliest move in game 0 where the model (almost) does this. At move 5, the model goes from being confused about tile C4 in layer 0 - assigning ~equal probabilities to whether it's "mine" or "theirs" - to being able to correctly tell it's "theirs" at layer 1.





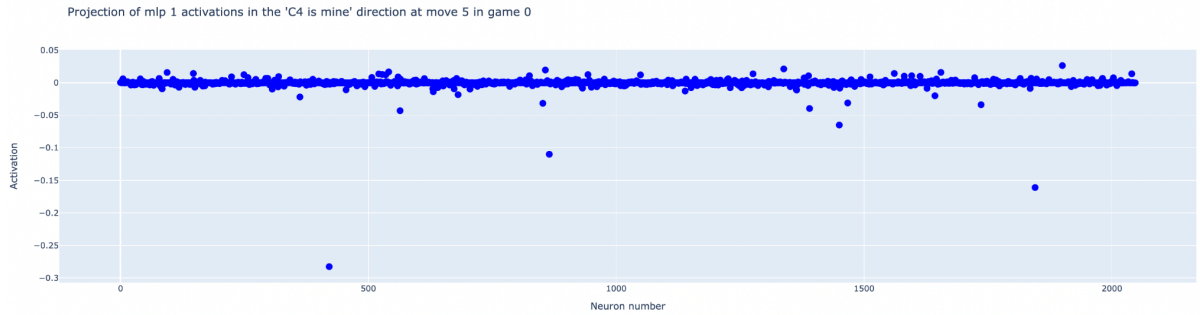




Can we understand the circuit in layers 0+1 that caused the model to “change its mind?”

To study this, I’ll follow the same strategy that I used to trace back how the model computes if a cell is blank or not blank in earlier work. (It might be useful to read that note before this one.)

First of all, which component(s) of resid_post_1 makes a big negative contribution to the “C4 is mine” direction? In layer one, the components resid_pre_1 , attn_out 1, mlp_out 1 project as -0.0653, 0.1932, **-0.7668** in this direction [1]. So let’s focus on mlp_out 1.

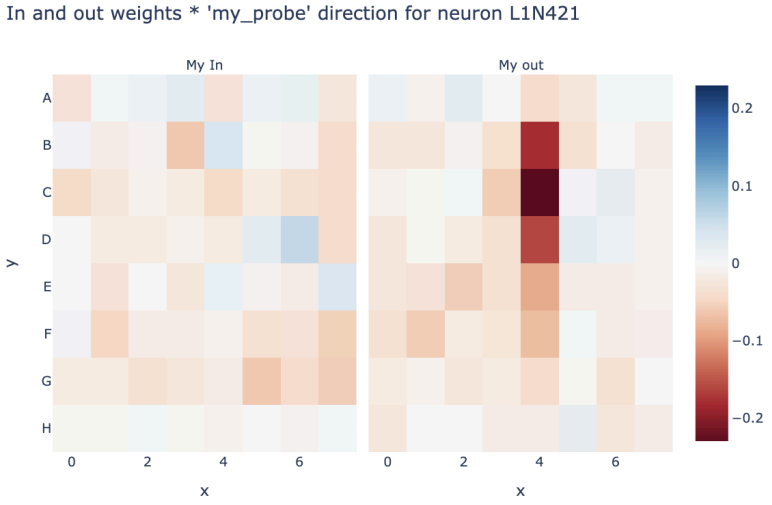
Looking at per-neuron contributions from mlp_out 1, there seem to be especially large negative contributions to the “C4 is mine” direction from a handful of individual neurons.




[1] Here and throughout this file, all numbers given are w.r.t. projection in the linear probe direction “my_probe_normalized” found by Neel Nanda for OthelloGPT (see code for details). However, only the ratios of such numbers and not their magnitudes are relevant for the story told throughout this note.

Let's dig into the first of these, L1N421.

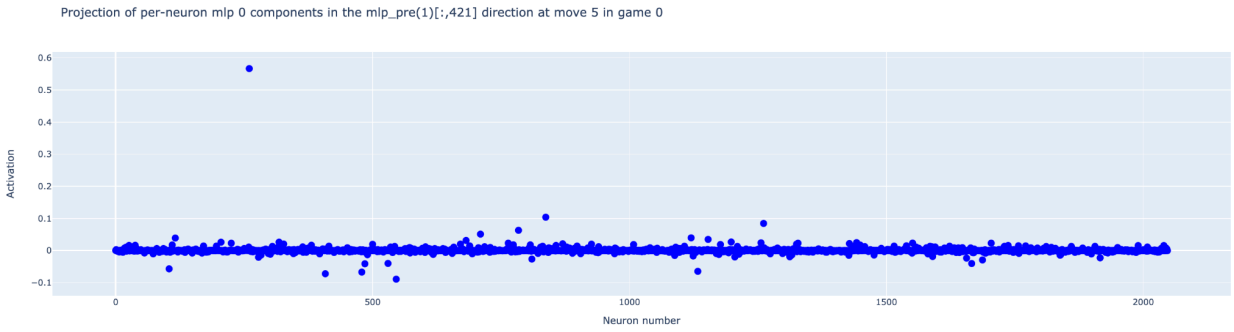
Since W_in, W_out for a fixed neuron are vectors of length 512, we can directly project them in the various probe directions. For L1N421, the projection of ingoing/outgoing weights in per-cell "this cell is mine" directions are




The RHS tells us that *if* L1N421 activates, it writes to the residual stream that squares B4/C4/D4 are not mine (as indeed is the case in this example). So a big activation from it could in part explain the model "changing its mind" about C4 being mine -> theirs in our example.

What caused L1N421 to activate at move 5 in game 0? Continuing to trace back,

- In earlier parts of layer 1, the layernormed projections of resid_pre 1 and attn 1 in the W_in[layer 1, move 5, 421] direction are (**3.622**, -0.081) resp.
- Splitting resid_pre 1 into components, the direction gets similar contributions (0.998, 1.504, 1.121) from resid_pre 0, attn 0, and mlp 0 respectively.
- Splitting the mlp 0 contribution into per-neuron contributions in this direction, they're dominated by the firing of a single neuron, L0N260:



L0N260 is the “A4 is not blank” neuron that I found in my blank vs. not blank note. I also checked that the projection of resid_pre 0 in this direction spikes when ‘A4’ is played. [2]

So in this example, resid_pre 0 and mlp 0 contribute to the firing of L1N421, hence the model’s change of mind from “C4 is mine” to “C4 is yours” between layers 0 -> 1, via a simple rule: “if A4 was just played, update things in the column below to ‘theirs’.”

This rule accounts for 20-25% of why OthelloGPT ‘changed its mind’ between layers 0 and 1 about who owns tile C4 at move 5 of game 0. [3]

# Refining the conjecture

Having found that in game 0, neuron L1N421 mediates a rule to upweight B4-C4-D4 towards “theirs” when it gets a signal from layer 0 that A4 was just played, we’d like to understand if this ‘mini-circuit’ is robustly present in other games.

## A modified conjecture from situations where the rule is wrong

Notably, in the ground-truth game of Othello, it need not be the case that playing A4 causes the column of tiles below it to flip to “theirs”. Playing A4 could instead cause tiles to flip along row A or one of two diagonals, leaving tiles in column 4 as either ‘blank’ or ‘mine’.

It’s natural to ask what happens to the conjectured ‘mini-circuit’ in these situations. Does L1N421 still (mis)fire, updating B4-C4-D4 towards ‘theirs’ (presumably to be corrected in later layers)? Or does OthelloGPT somehow know to suppress L1N421 in these situations?

It turns out that the former happens if B4/C4 should be ‘mine’ and the latter if B4/C4 should be ‘blank’.

As a first step, let’s go through the focus games and pick out the ones where at least one of B4 or C4 does *not* flip to “theirs” when A4 is played.

There are 13/50 such games. In 6 of them, at least one of B4 or C4 is blank:

---

[2] The attn 0 contribution to firing of L1N421 seems ~equal across the heads, and I didn’t try to interpret it. I’ll give an example of a circuit involving an attention head below.

[3] In the sense taht L1N421 contributes to about 37% of the change of mind, and about 60% of its activation can be attributed to A4 being played via resid_pre 0 and the associated firing of L0N260 in mlp 0. (Besides L1N421, another 30% of the direct contribution to the ‘change of mind’ from mlp 1 come from the firing of the next two neurons L1N1845 and L1N865 resp, both of which also fire sparsely and nudge multiple tiles in the column below A4 towards being ‘theirs’.)

- Game index 3, move index 50; *B4 is blank*
- Game index 11, move index 28; *B4 is blank*
- Game index 15, move index 41; *B4 is blank*
- Game index 33, move index 25; *B4 is blank*
- Game index 36, move index 28; *C4 is blank*
- Game index 41, move index 36. *B4 is blank*

and in another 7 at least one of B4 or C4 is ‘mine’ instead of ‘theirs’:

- Game index 4, move index 35; *C4 is mine*
- Game index 12, move index 50; *C4 is mine*
- Game index 23, move index 25; *B4 and C4 are mine*
- Game index 26, move index 11; *B4 and C4 are mine*
- Game index 32, move index 54; *C4 is mine*
- Game index 39, move index 57; *C4 is mine*
- Game index 47, move index 40; *C4 is mine.*

On the other hand, from the Othelloscope page for L1N421, we see that it *almost* always activates when A4 is played, with six exceptions: precisely the six in the top list.

This leads us to make the modified conjecture that

**OthelloGPT has a mini-circuit “A4 is played AND B4 is not blank AND C4 is not blank -> update weights of B4+C4+D4 towards “theirs”,**

mediated via the firing or not of neuron L1N421, which appears to follow this rule across all focus games. [4] [5]

## How the model mechanistically checks that B4 is not blank

[4] By eyeballing the Othelloscope page of L1N421 in this case, we get the same info as a spectrum plot, since the rule that we think it implements can be read off directly from the input string. Neurons in later layers may mediate logical rules involving learned features where this won’t be the case.

[5] This circuit *mistakenly* fires, directionally nudging the model’s representation of cells B4/C4/D4 the wrong way, when B4 and C4 are not blank but should be “mine” instead of “theirs”. In the 7 games where this happens, the model takes longer to settle on the ‘correct’ state for B4/C4. Namely,

- Game index 4, move index 35; *C4 is mine* — *C4 is ‘theirs’ in L1, confused in L2 and ‘mine’ in L3*
- Game index 12, move index 50; *C4 is mine* — *C4 is ‘theirs’ in L2, confused in L3, ‘mine’ in L4*
- Game index 23, move index 25; *B4 and C4 are mine* — *B4 is ‘theirs’ in L2, confused in L3, ‘mine’ in L4*; *C4 is ‘theirs’ in L2, confused in L3, ‘mine’ in L4*
- Game index 26, move index 11; *B4 and C4 are mine* — *B4 is ‘theirs’ in L0, confused in L1,L2, ‘mine’ in L3*; *C4 is confused in L0-L2, ‘mine’ in L3*
- Game index 32, move index 54; *C4 is mine* — *C4 is confused in L0-L2, ‘mine’ in L3*
- Game index 39, move index 57; *C4 is mine* — *C4 is ‘theirs’ in L0, confused in L1,L2, ‘mine’ in L3*
- Game index 47, move index 40; *C4 is mine.* — *C4 is confused in L0, ‘mine’ in L1.*

where I label a square as theirs/mine if P(theirs/mine) > 0.8, else confused.

We can go further and mechanistically understand how the model implements the logic “B4 is not blank → Don’t fire L1N421 → Don’t nudge the weights of B4+C4+D4 towards ‘theirs’”.

To do this, let’s look at how the model does it for an example then generalize to other games.

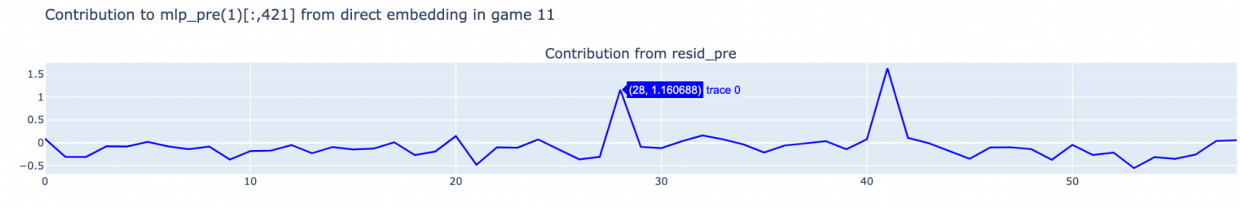
**Example: move index 28 in game 11**

In game 11, move 28, where B4 is blank when A4 is played, L1N421 *doesn’t* fire when A4 is played. We’d like to understand why not.

To see why, let’s compare the inputs to mlp_pre(1)[:,421] in game 11 at move 28 with the inputs to mlp_pre(1)[:421] for game 0 that we found on page 3.

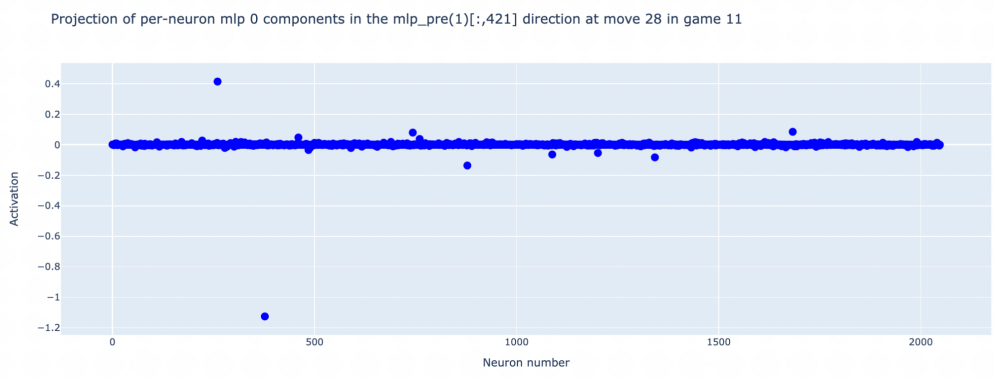
In game 11, the contributions to mlp_pre(1)[:,421] from resid_pre(1) and attn(1) are fairly small: (0.785, -0.305) resp, whereas the contribution from resid_pre(1) in game 0 was large. This suggests that something changes in resid_pre(1) depending on if B4 is played before A4 or not.

The contributions from resid_pre(1) can in turn be broken down into (1.16, 0.20, -0.58) from resid_pre(0), attn_out(0), mlp_out(0) resp. The direct contribution of resid_pre(0) is actually similar in magnitude and kind in games 11 and 0,




but something different seems to be going on at mlp(0).

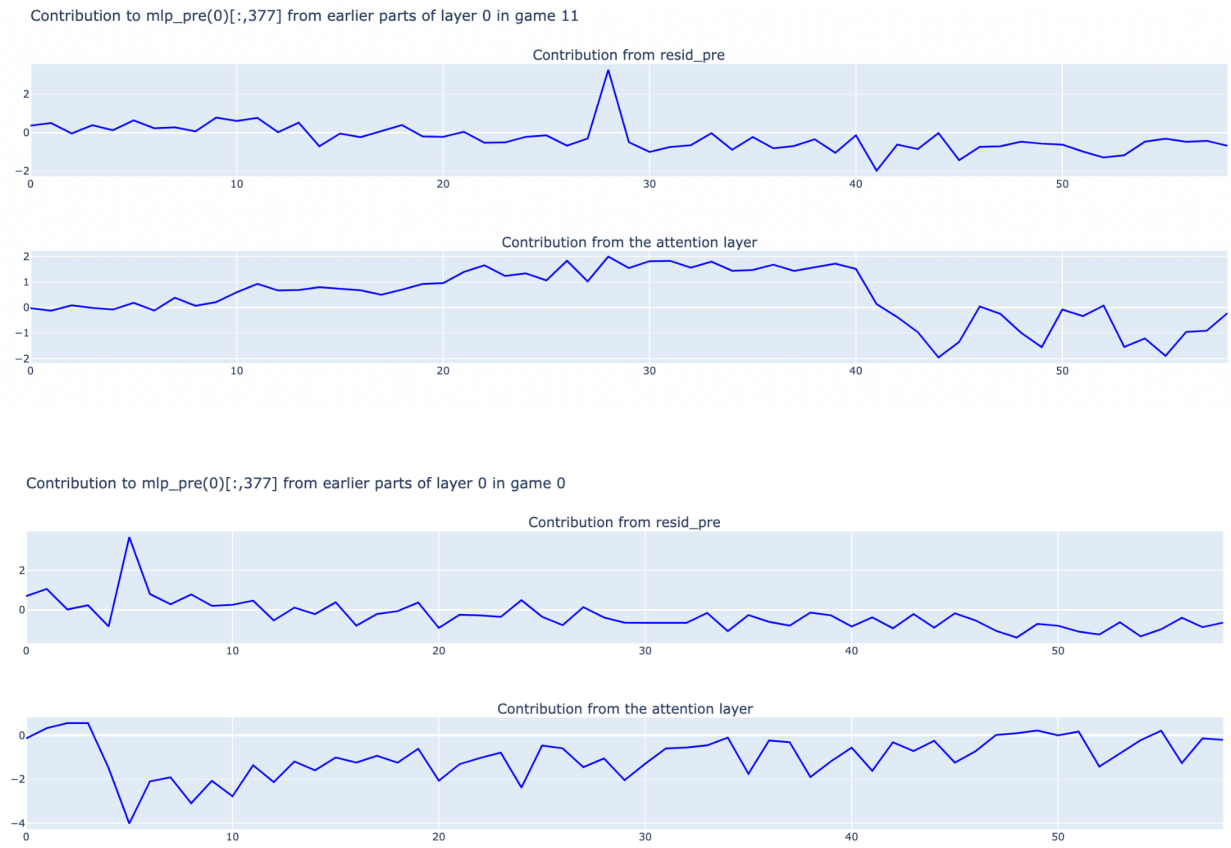
Looking at per-neuron components, the difference between these two games is clear. Compare the following plot for game 11 to the same for game 1 at the bottom of page 3:



While L1N421 is incentivised to fire by the firing of the A4 “move detector” neuron L0N260 in both cases, it gets an even larger negative signal from a different neuron, L0N377, in game 11.

Looking at the Othelloscope page for L0N377, it fires in precisely the five focus games 3, 11, 15, 33, 41 where A4 appears before B4. So we conjecture that L0N377 is responsible for telling L1N421 not to fire and turning off our mini-circuit if B4 is blank.

Which inputs to L0N377 cause it to behave differently depending whether A4 appears before B4? The natural guess is that it’s attn 0 and not resid_pre 0, since this requires us to pass information from earlier sequence positions to the move where A4 is played. And indeed, comparing inputs to L0N377 in games 0 and 11,




we see that in both cases, the projection of resid_pre 0 in the W_in(0)[:,377] direction is qualitatively the same, while the projection of attn 0 in this direction is different.

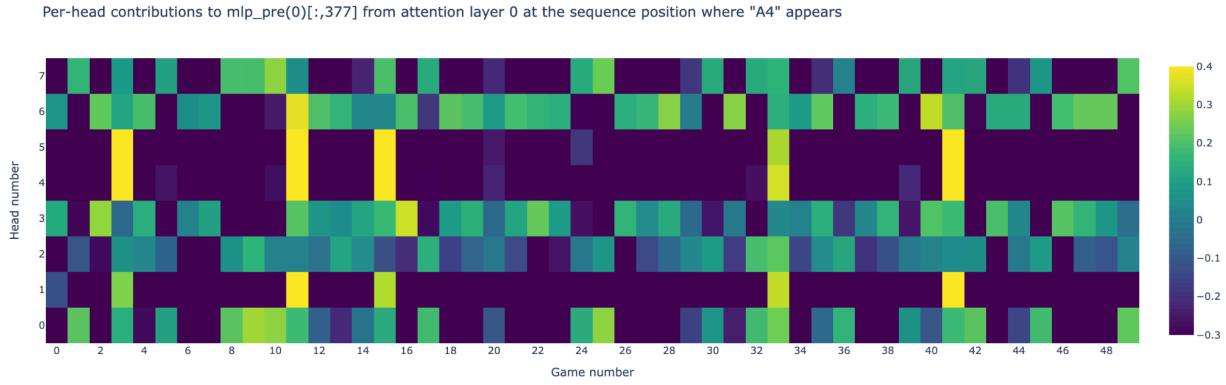
**Understanding how L0N377 “knows” if B4 appears before A4 in the input string**

To finish this section, we’ll give a mechanistic account of how L0N377 ‘knows’ to fire iff B4 does not appear before A4 in the input string.

**Per-head contribution**

Let's see how the per-head contributions in attn 0 to mlp_pre(0)[:,377] change depending on whether 'B4' appeared earlier in the input string.

Here I've plotted the per-head contributions to mlp_pre(0)[:,377] at whichever position 'A4' appears across all 50 focus games. Games 3, 11, 15, 33, 41 are the ones where 'B4' appears after 'A4' in the input string.

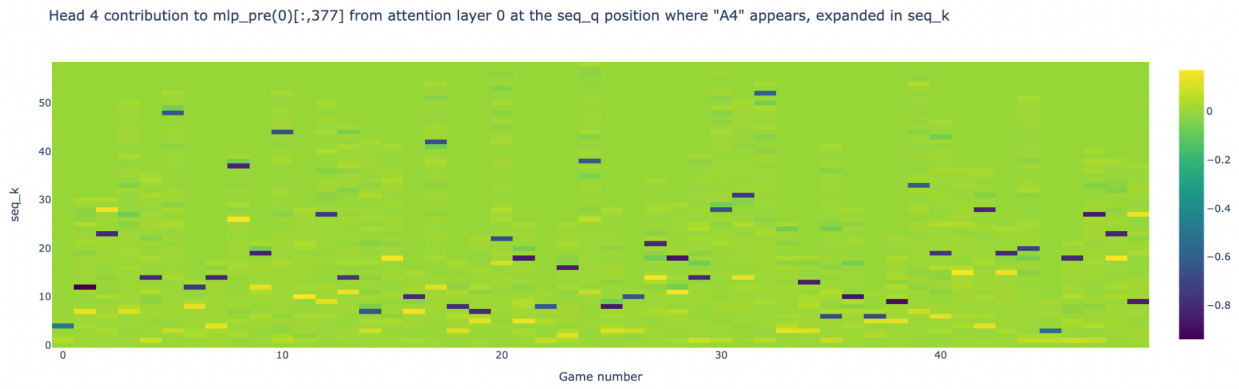



Heads 0.1, 0.4, 0.5 seem responsible for keeping track of this across all focus games.

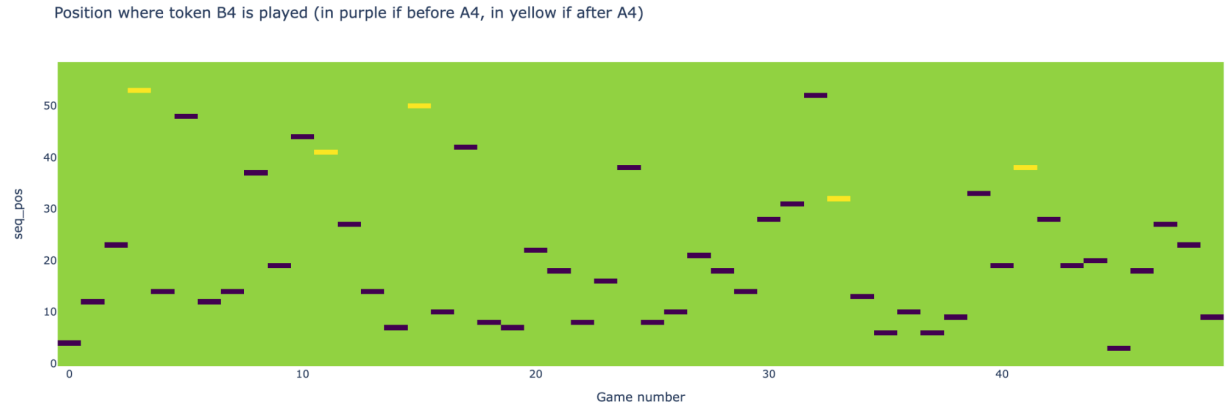
We'd like to verify that heads 0.1, 0.4, 0.5 are passing information specifically about 'B4' to the position where 'A4' is played. To see this, we expand the per-head contribution in the index k. Recall that attn_out is a product of contributions from attn_v and attn_pattern, summed over an internal index k that runs over the sequence positions. [6]

Below I plot the per-k contribution for head 0.4 (where I checked manually that the plots for heads 0.1 , 0.5 look qualitatively the same):

[6] More precisely, the thing that we plotted above for head 4 was attn_pattern[head_idx = (1,4,5), seq_q = where 'A4' appears, seq_k] @ attn_v[seq_k, head_idx = (1,4,5), d_head] @ w_O[head_idx = (1,4,5)] @ W_in[0,: 377]. We can plot the red and blue parts separately, as we did in the blank vs. not blank file. If we did this here, we would see that the attention patterns (blue part) are actually relatively constant for these heads, so their role in life is to pass in large signals from the value part (red part), which spikes negatively when the B4 token appears.




Comparing the above plot to a plot of the sequence position where token B4 is played (marked below in purple if before A4, in yellow if after A4),




we see that these heads' projections in the W_in[0,: 377] direction are large and negative exactly when B4 is played, in games where B4 is played before A4.

(There's presumably a similar mechanistic explanation in layers 0+1 for why L1N421 gets suppressed when C4 is blank, but I haven't tried to find it.)

# Summary

To recapitulate,

- We found evidence that OthelloGPT contains a 'mini-circuit' in layers 0+1 which says "A4 is played AND B4 is not blank AND C4 is not blank → update internal representation of B4+C4+D4 towards being "theirs"."

- Mechanistically, we found that
    - The update on B4+C4+D4 comes from the firing of neuron L1N421.

    - The logic “B4 is not blank” is mediated by an earlier neuron L0N377, which fires to inhibit L1N421 iff the token for B4 does not appear before the one for A4 in the input string.
    - The information about whether B4 appears before A4 in the input string is passed to L0N377 through attention heads 0.1, 0.4, 0.5, which send a large negative signal from the key position where B4 is played to inhibit L0N377 iff B4 appears before A4 so is visible to the attention head.

- This circuit “explains” 20-25% of why OthelloGPT “changes its mind” about whether C4 is blank at move 5 in game 0 between layers 0 and 1. We conjecture that it’s one of very many heuristic rules by which OthelloGPT aggregately learns the board state.