How OthelloGPT computes that cells are blank

In this note, I'll look into how OthelloGPT computes if a cell is blank. This should be fairly easy since it just requires the model to keep track of whether a token appeared before or not.

As we'll see, (the direct embedding in) **resid_pre 0[1] keeps track of whether a cell was just played, and a subset of attention heads in layer 0 pass this info to other sequence positions.**

Case study: Cell B3 in focus game 0

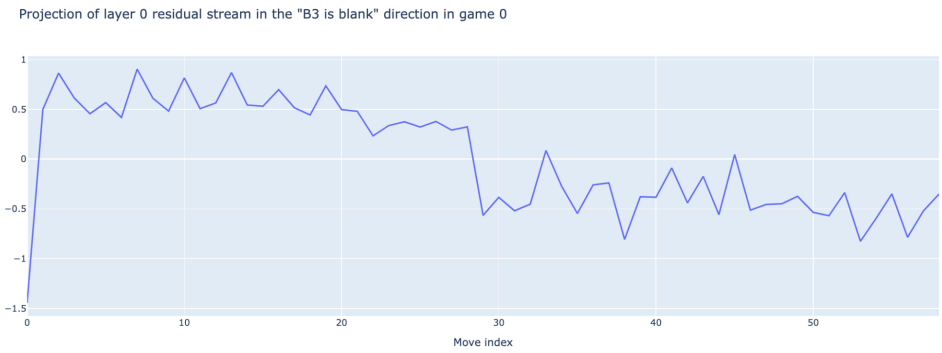
To get some surface area on the problem, let's look at how the model computes if cell B3 is(n't) blank in focus game 0 [2]. We'll understand how this case works then generalize it.

In game 0, B3 is played as the 30th move (move index 29) of the game. From now on, I'll refer to move index only instead of the actual move number, which is move index + 1.

Finding the relevant layer

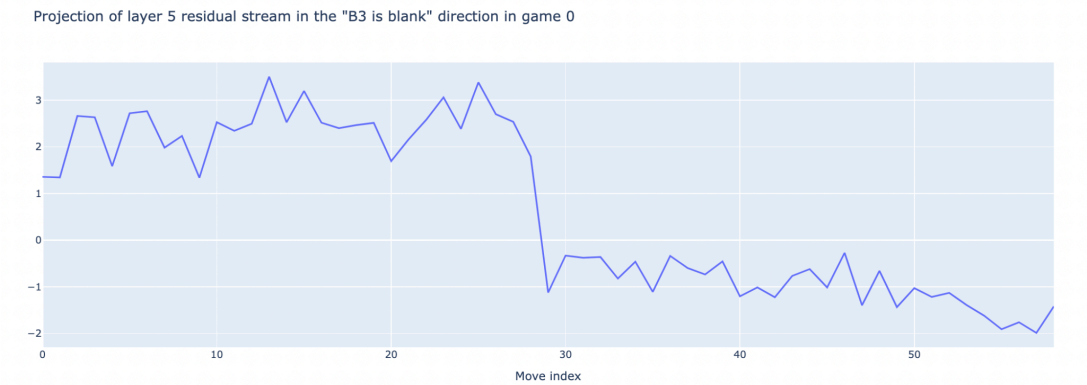
Let's first find the layer at which the model more or less learns if B3 is blank. To track this, we look for the earliest layer where the inner product of Neel's 'B3 is blank' probe with the residual stream is large up to move index 29 and small thereafter.

In fact, this seems to more or less happen by `resid_post_0`:



[1] The layer names I use throughout this file come from cache key names in TransformerLens which I used throughout this file.
[2] The focus games are a test set of 50 games that are available in the colab notebook.

although the model seems to become more sure about the decision boundary in later layers.

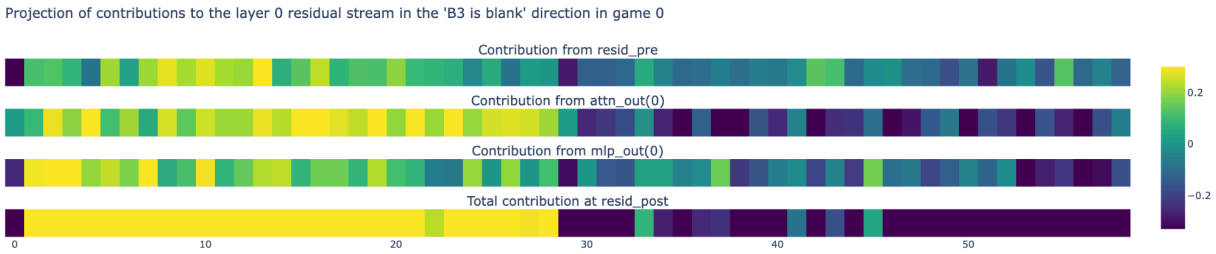


In the rest of this note, I'll focus on layer 0.

Finding the salient contribution to `resid_post_0`

`resid_post_0` is the sum of three contributions: `resid_pre_0`, `attn_out(0)` and `mlp_out(0)`. Which of these is directly responsible for resid_post_0 "knowing" that B3 is(n't) blank before (after) move 29?

To see this, I plotted the projection of their contributions to the 'B3 is blank' direction after each move. [3]



There seem to be different mechanisms for how the model tells if B3 is blank *after* move 29 and *at* move 29 itself.

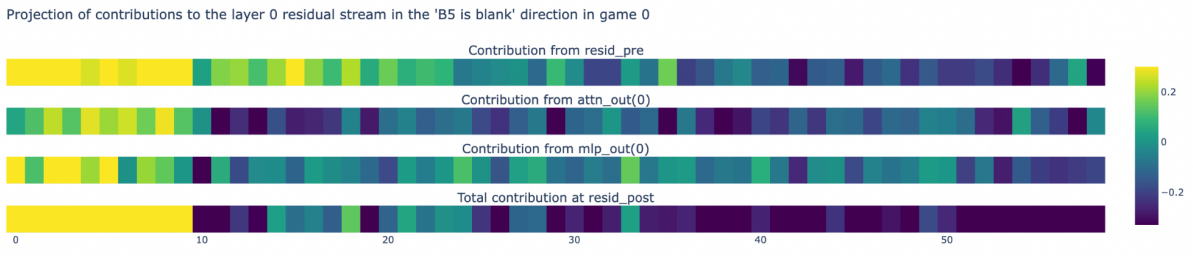
- Before and after - but not at - move 29, the attention layer outputs seem quite good at discriminating if B3 is blank.
- *At* move 29, the main culprits seem to be `resid_pre` and/or `mlp_out`.

A "validation" example: cell B5 in game 0

[3] I also checked that the total contribution from resid_post_0 is the sum of the three.

To confirm and sharpen our hypotheses, let's look at a different cell, B5 in game 0. This cell is played at move index 10 in game 0.

In this case, the analog of the plot above is



From column 10 of this plot, we hypothesize that `mlp_out` (not `resid_pre`) is the main contributor for the direct contribution to resid_out by which the model tells that a tile was just played.

How the model tells that a tile was just played

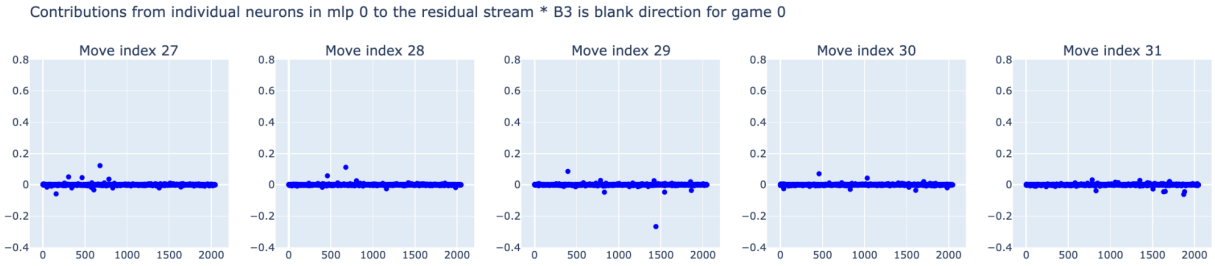
With that said, let's work backwards from the MLP layer to see how the model tells that a tile was just played. In this section, I'll again first look at (cell B3 in) game 0 to make some hypotheses. I'll then check the hypotheses across all cells in all games.

The MLP layer itself

Neuron L0N1442 contributes to the "B3 is blank" direction iff B3 was just played

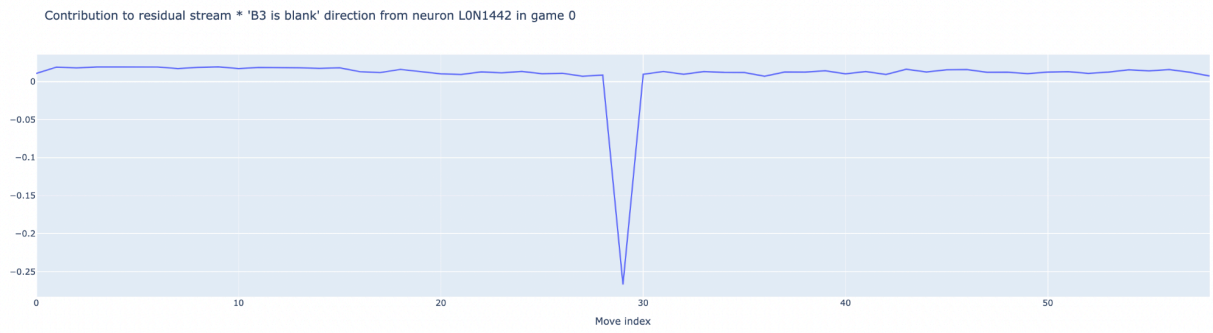
One thing we can do is break the contribution from mlp_out 0 in a given probe direction into a per-neuron contribution for each of 2048 neurons in the layer.

Doing this for the "B3 is blank" direction at/near move 29 in game 0, we find

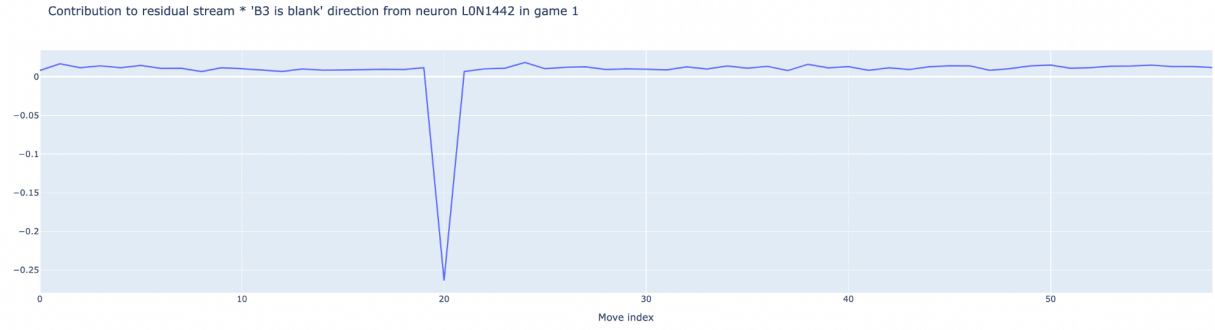


Interestingly, there's an outlier neuron L0N1442 that gives a large negative contribution in the "B3 is blank" direction (only) at move 29 itself.

Here's its contribution in the "B3 is blank" direction as a function of the sequence position.



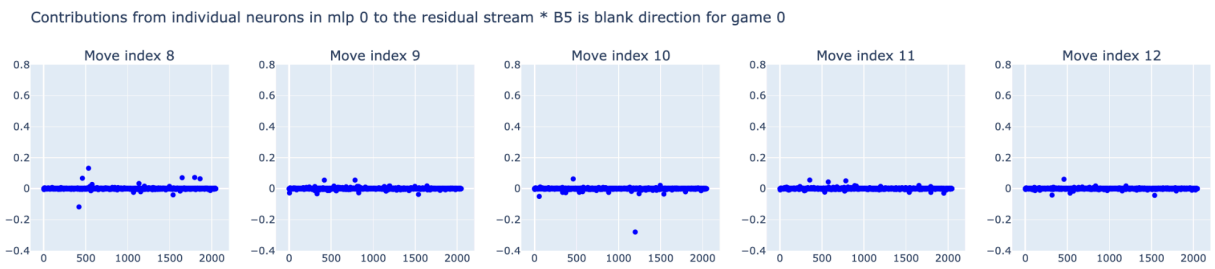
In a different game, this neuron contributes to the "B3 is blank" direction exclusively on the (different) move when B3 is played.

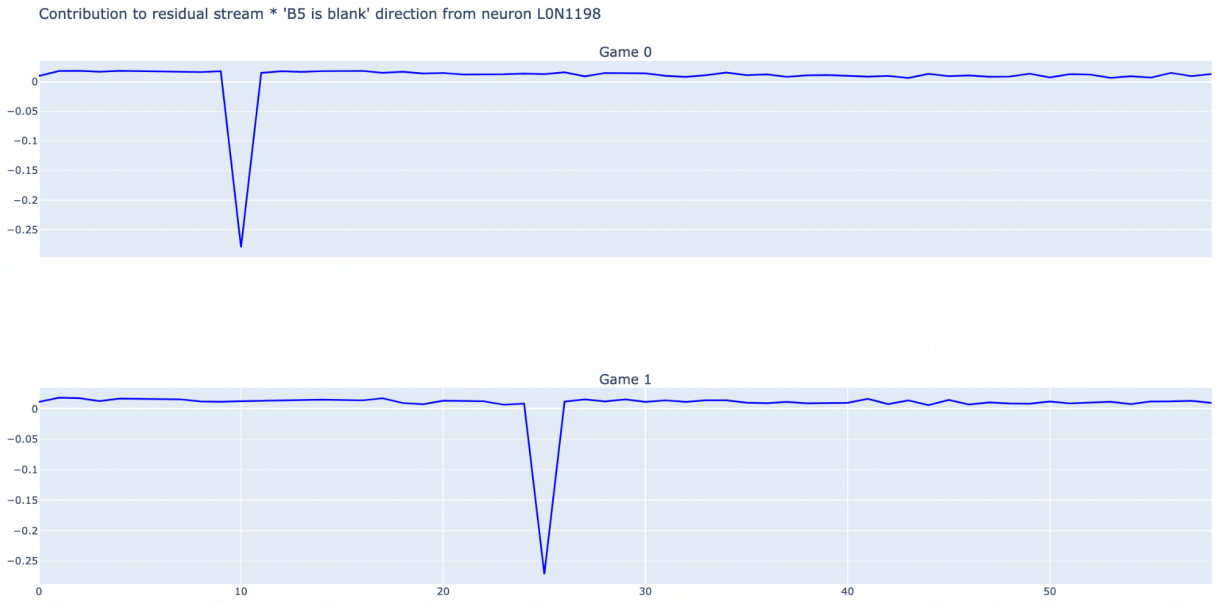


More generally, its Othelloscope page confirms that it fires iff B3 was just played across all focus games.

Neuron L0N1198 contributes to the "B5 is blank" direction iff B5 was just played

A different neuron, L0N1198 for a different cell B5 seems to play a similar role.





Finding a “move detector neuron” in mlp 0 for every tile

Is there perhaps a similar “move detector” neuron in mlp 0 for every tile, that activates iff that tile was just played? The answer appears to be yes!

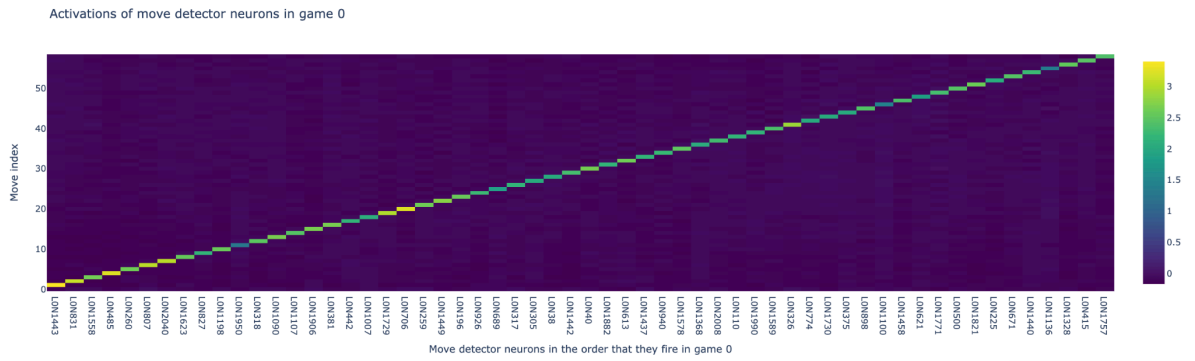
To find them, I

1. Iterated through the moves of game 0, finding the neuron with the greatest negative contribution to the residual stream in the “tile just played is not blank” direction for each move index <> tile played.

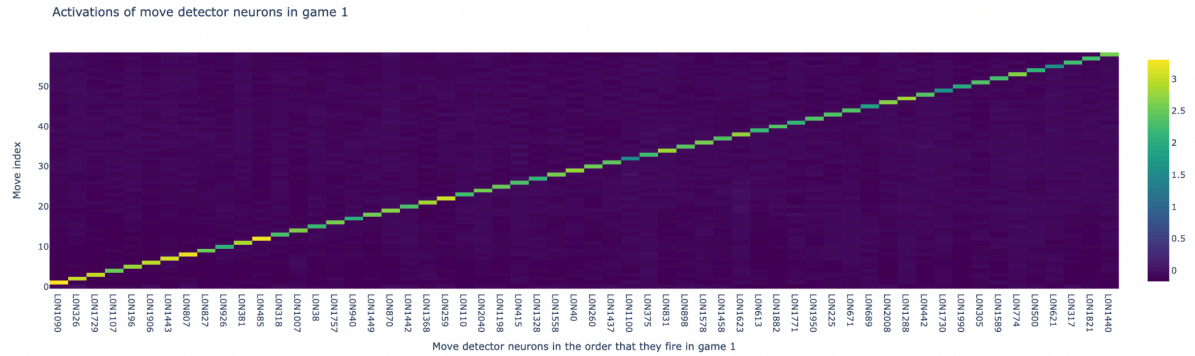
2. Checked that its contribution to the “tile just played is not blank” direction is >100x greater than the mean absolute value contribution from all neurons in mlp 0. (This is an analog of the 1st plot above.)

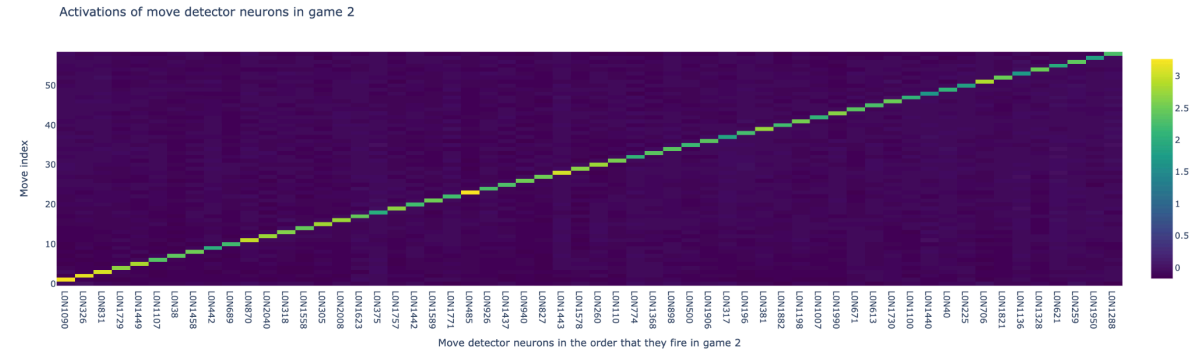
3. Checked that it does NOT make a large contribution to the residual stream in the “tile just played is blank” direction at all other sequence positions in the game. (This is an analog of the 2nd plot above). [4]

In fact, these neurons seem to fire at all, & not just in the “current tile is blank” direction, when the corresponding tile was just played. Here are plots of the neurons’ activations across move number, listed in the order that they’re played in game 0:



I then checked that the same neurons detect the filling of paired tiles in other games. Here are representative plots for two other games, although I checked it manually for 20 games.





[4] I also checked that there appears to be *exactly* one neuron in mlp 0 satisfying properties 2+3 for each cell in game 0. (In some cases I found at most one other neuron that also satisfies condition 2 for a given cell, but which fires on other cells besides that one.)

In the above three plots, the neurons appearing on the x-axis are the same, permuted in the order that their matching tiles were played in the different games.

One can also look these neurons up on OthelloScope: e.g. L0N1443, L0N831, L0N1558.

To summarize so far, **for every tile in OthelloGPT** besides the four center ones, there seems to be **exactly one** neuron in MLP0 that activates **exactly when** that tile is played.

Inputs to the MLP layer

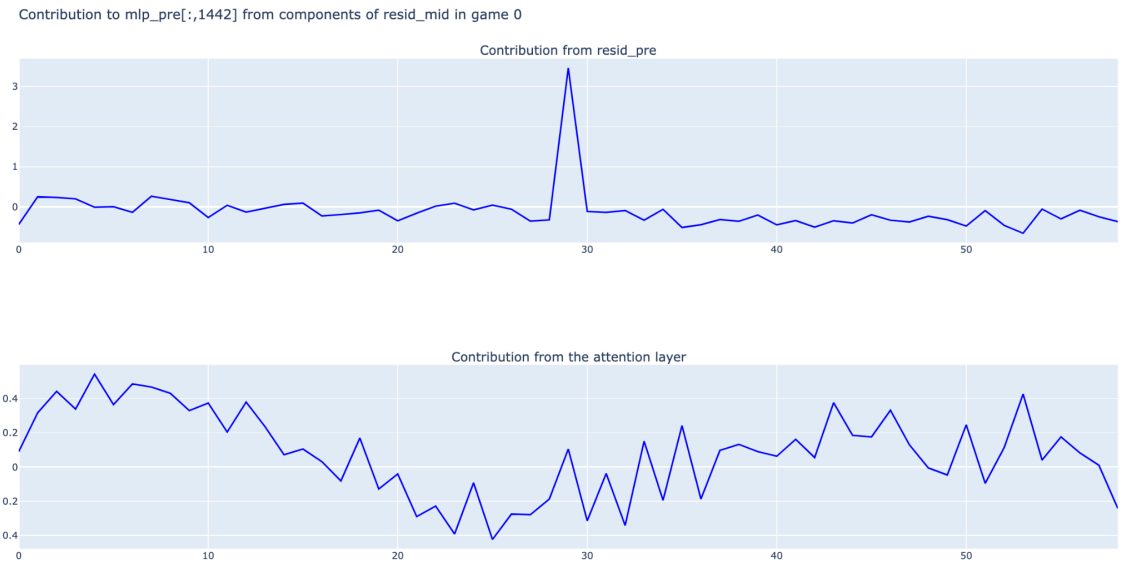
Having established that the firing of a "move detector neuron" in mlp 0 is responsible for the direct contribution to resid_post 0 in the "current tile is not blank" direction, we'd next like to find out which earlier parts of layer 0 cause these neurons to fire. The possibilities are either resid_pre 0 or (one of the attention heads in) attn 0.

Again, let's start with our running example.

Case study: Which inputs activate L0N1442 at move 29 in game 0?

To study this, let's plot the contributions of `resid_pre` and from `attn_out(0)` to `mlp_pre(0)[:,1442]` in game 0 as a function of the move number. [5] The activation of neuron L0N1442 is gelu(mlp_pre[:,1442]), i.e. very roughly the positive part of these plots if their combined contribution is positive.

[5] I.e., the sum of these two contributions with model.b_in[0] reproduces the activation at mlp_pre(0)[:,1442].



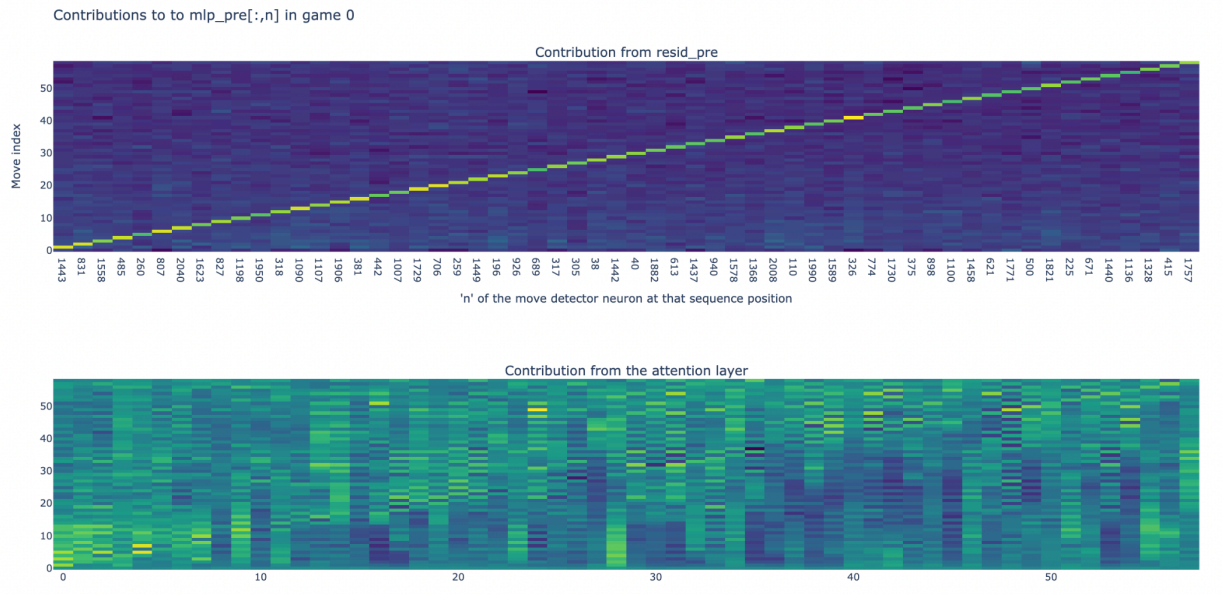
We see that the activation of L0N1442 comes almost entirely from resid_pre 0; the attention layer doesn't do much here. [6]

Resid_pre 0 activates move detector neurons for other tiles & games

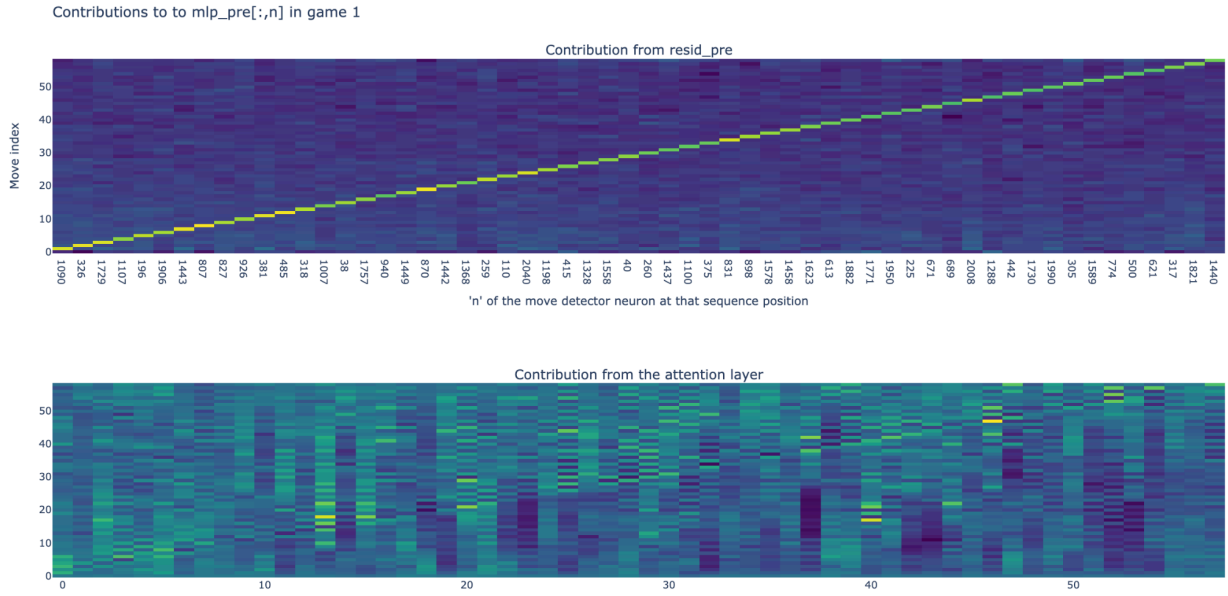
We hypothesize that resid_pre 0, and not attn(0), is also responsible for activating the other "move detector neurons" in layer 0.

To validate this hypothesis, we first check that across all moves in game 0, resid_pre contributes positively to mlp_pre of the corresponding "move detector neuron", hence causes it to fire, while the attn 0 contributions in this direction have no qualitatively discernable pattern to them.

[6] Here and below, I also checked that the direct embedding part of resid_pre 0, not the positional embedding is the relevant part.



I then validated it for other games. E.g.



Summary

To summarize, here's how OthelloGPT appears to tell if a tile was just played.

- For each tile, mlp 0 has a neuron that writes to the "This tile is not blank" probe direction iff the tile was just played.

- That neuron fires when the corresponding tile appears in the input sequence, due to resid_pre 0 having a large projection in the W_in[:,(neuron number)] direction there.

- In other words, resid_pre itself stores info at each sequence position about which tile was just played. This is at most 60 of its 512 dimensions so seems fine. [7]

How the model tells that a tile was played 1+ moves ago

Let's move on to how the model tracks that *previously played* cells are not blank.

In this case, the attention heads must move info from resid_pre 0, whose output at a given sequence position contains info that a cell was *just* played, to other sequence positions. (We also saw that the attention layer is empirically relevant on pages 2+3.)

Below, I'll carefully trace out the circuits by which the model tracks that the tile played 1 move ago and 2 moves ago is not blank. I'll then conjecture and give some evidence that a similar mechanism explains how the model tracks which tiles were played 3+ moves ago.

One move ago

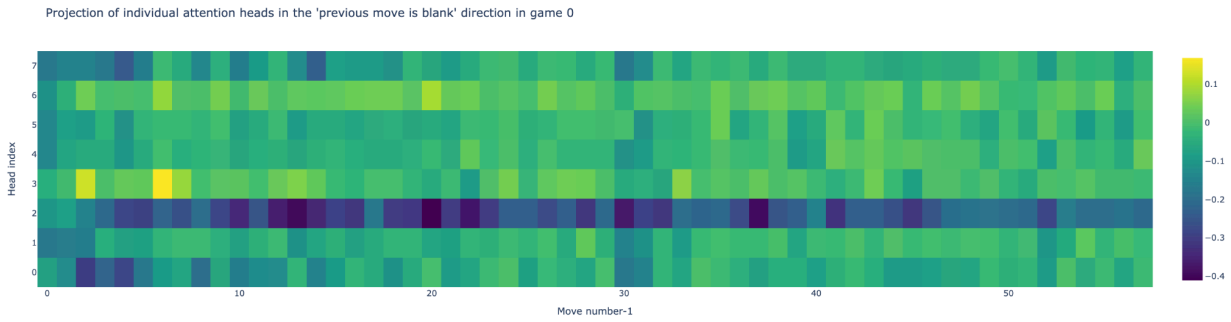
As described above, we expect the direct contribution to come from the attn 0 layer.

Head 0.2 tracks tiles filled one move ago

Let's see if we can isolate an individual head or heads in attn 0 that are responsible for the direct contribution.

To study this, I decomposed the projection of attn_out_0 in the "tile played 1 move ago is blank" direction in game 0 into per-head contributions. (I.e. in column 29 below, I plot the per-head projection of attn_out_0 in the "B3 is blank" direction.)

[7] It's not clear to me how the model gains from using a "move detector neuron" in mlp 0 to line resid_pre 0 up with the "current tile is(n't) blank" direction, instead of just having resid_pre 0 line up with that direction itself. Perhaps the model redundantly encodes this info since it has capacity to spare.



We see that the main contribution is from head 0.2.

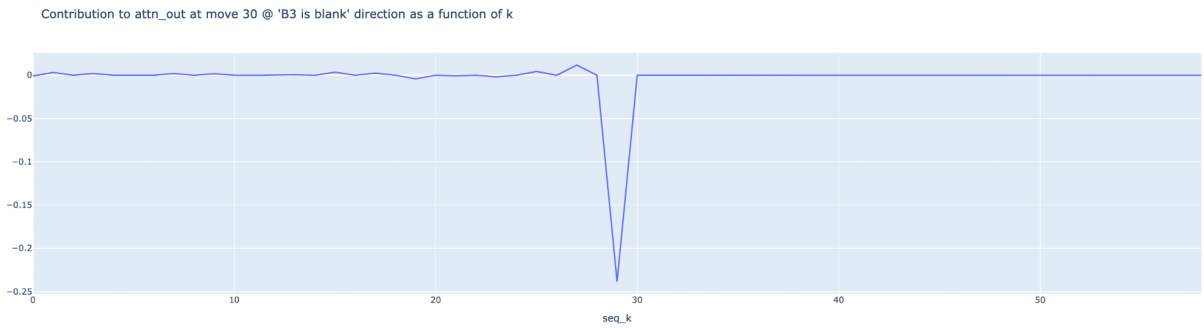
I checked manually that the same thing happens in 20 other games.

Information flow through the attention block

Can we backtrack through how the inputs to attn_out_0 (head = 0.2) explain this result? [8]

I think this is easiest to explain in a concrete example - just at the level of keeping the notation straight - so let me specialize to how the model tells at move 30 in game 0, one move after B3 was played, that B3 is not blank. However, I checked that the logic generalizes to all other sequence positions in game 0 and other focus games.

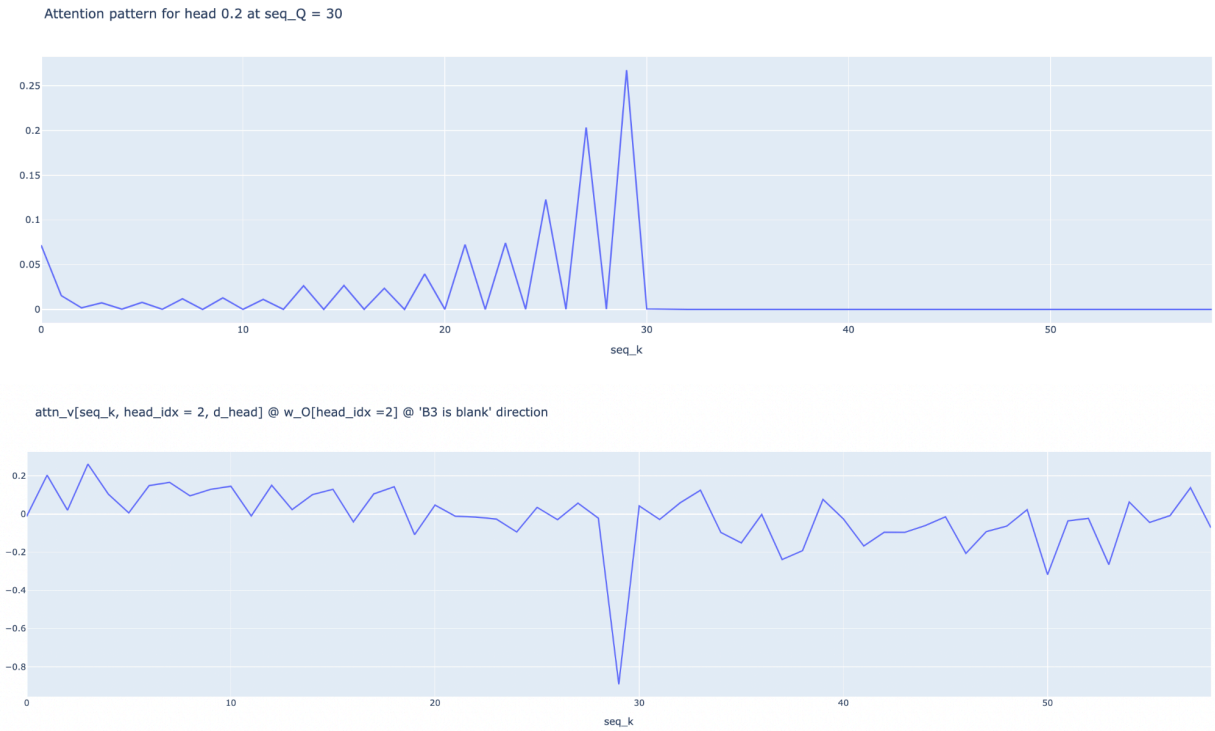
In move 30 of game 0, the number attn_out[seq_pos = 30, head_idx = 2, d_model] @ the “B3 is blank” probe direction [9] can be decomposed into attn_pattern[head_idx = 2, q = 30, k] @ attn_v[k, head_idx = 2, d_head] @ W_O[head_idx = 2] @ the “B3 is blank” probe direction, summed over k. Plotting attn_pattern[n_head= 2, seq_q = 30, seq_k] @ attn_v[seq_k, head_idx = 2, d_head] @ w_O[head_idx =2] @ “B3 is blank” probe direction *without* summing over k, we find that the main contribution comes from seq_k = 29, where B3 was in fact played.



[8] See here for the info flow diagram through the transformer block.
[9] The analog of this at move n \neq 30 is attn_out[seq_pos = n, head_idx = 2, d_model] @ the “tile filled at move n-1 is blank” probe direction.

I.e., attention head 0.2 moves the information that B3 was played at seq_pos 29 to seq_pos 30.

The above plot is the product of the blue and red parts above, which we can plot separately:



The latter, value part of the attention block tracks that B3 was played at move 29. [10]

(Note that we can further decompose this to resid_pre 0 @ W_V[layer 0, head_idx = 2] @ W_O[layer 0, head_idx = 2] @ ‘B3 is blank’ probe direction, where everything but resid_pre 0 is linear and fixed at inference time. So this plot reflects our previous observation that the direct embedding already keeps track of whether a tile was just played.

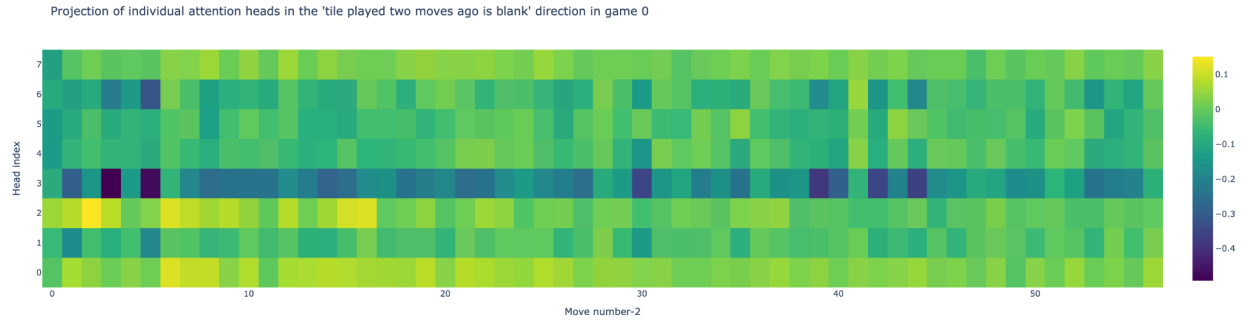
It could be interesting to check if the direction W_V [head 0.2] @ W_O [head 0.2] @ ‘(some cell) is blank’ direction in which we project the direct embedding here is aligned with the W_in[: move detector neuron for that cell] direction that we projected in to get a qualitatively similar plot back on page 8, but I won’t do this now.)

Two moves ago

Head 0.3 tracks tiles filled two move ago

[10] More generally, I checked that the analog attn_v(head_idx = 0.2) @ W_O(head_idx = 0.2) @ ‘some cell is blank’ probe direction spikes on the move where that cell was played.

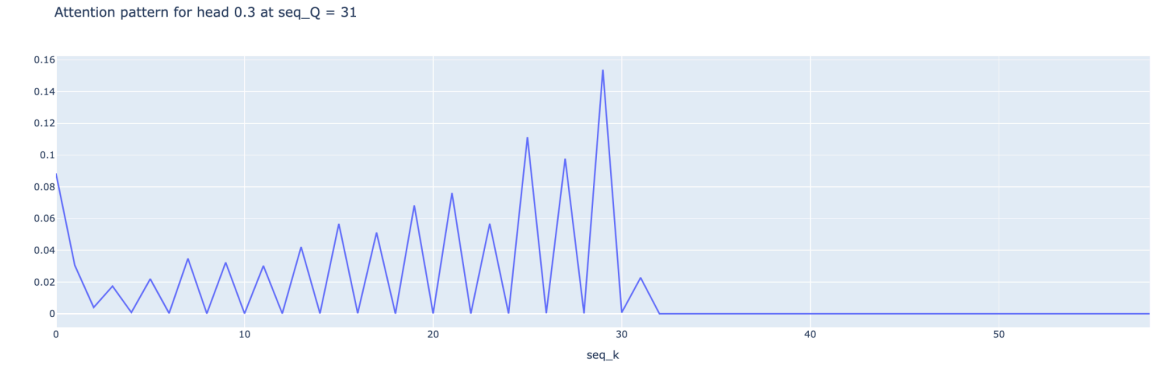
Going back two moves, we can similarly decompose the projection of attn_out_0 in the “tile played 2 moves ago is blank” direction in game 0 into per-head contributions,

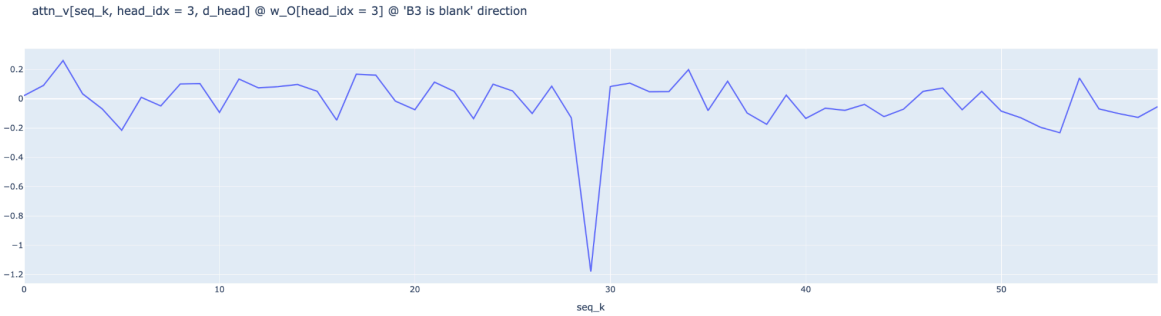


which I again checked manually for several games.

Information flow through the attention block

In this case, we can make analogous plots as above in the seq_k index. Here they are without comment for move 31 of game 0. I checked they’re similar for other moves/games. It seems clear that a similar story goes through.





Earlier moves

Rather than work out all earlier moves, I'll end with some general evidence that we expect a similar mechanism to work for them.

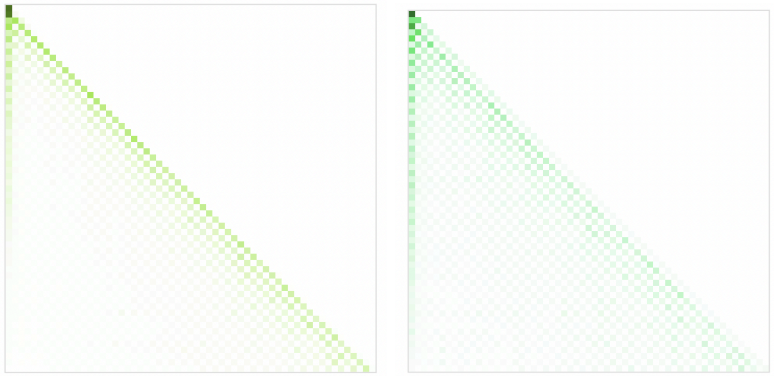
- **The OV part of the circuit.** For all eight heads in layer 0, the combination

  resid_pre 0 @ W_V [head_idx] @ W_O [head_idx] @ '(some cell) is blank'

  has the same qualitative shape as the plot immediately above (where we specialized to head 3), spiking iff (some cell) was just played.

  So again, resid_pre fundamentally keeps track of which token was just played and we can recover this info by projecting it in various directions. [11]

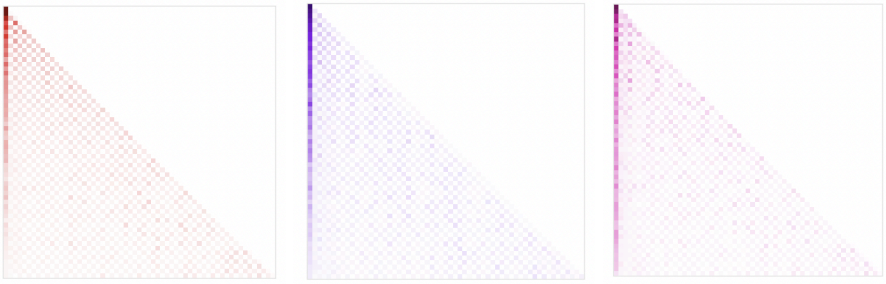
- **The attention patterns.** Above, we observed that heads 0.2, 0.3 move info that a cell was played (one, two) tokens ago to the current position. Visually, we can see this by plotting the attention patterns of heads 0.2 and 0.3 across a game. Here they are for game 0, but I checked that the patterns look qualitatively similar in other games.



  The attention patterns for head 2 (left) and head 3 (right) are dominated by diagonals offset by (1,2) resp., so move info from seq_pos to (seq_pos+1, seq_pos+2).

  How about for earlier moves? For this, we observe that the attention patterns for heads 0.0 and 0.7 (left two patterns shown below for game 0) attend to *all* previous tokens played by the other player (i.e. to every other token starting one move back, similar to head 2 w/o the decay in the top plot on page 12), while the attention pattern for 0.6 (rightmost pattern shown below for game 0) attends to all previous tokens played by the current player (i.e. to every other token starting two moves back, similar to head 3 w/o decay.)

[11] It could be interesting to check if the W_{OV} [different head] directions are aligned or if this info is recorded redundantly in a subspace of resid_pre.



These would let us move info about whether or not a cell was played n+1, n+2 moves ago for all n to the current position.

Summary

To summarize, here's how the model appears to tell that a tile was played in the past.

- For each tile and all attention heads in layer 0, resid_pre @ W_V [head_idx] @ W_O [head_idx] @ '(some cell) is blank' spikes at the sequence position where (some cell) is played.
- Attention heads 0.0, 0.2, 0.7 can move this signal for tokens played an odd number of moves ago to the current sequence position.
- Attention heads 0.3, 0.5 can move this signal for tokens played an even number of moves ago to the current sequence position.
- I checked that heads 0.2 and 0.3 resp. *in fact* move the signal for tokens played one or two moves ago resp. to the current sequence position, but leave a check of the general case to future work.