Exploring the Limits of OthelloGPT's Emergent Representations

Julian Baldwin

Abstract:

Previous investigations suggest that transformer models can learn linear, emergent representations of high-level concepts. This report aims to explore linear representations as an avenue for interpretability and steering of language model behavior. Specifically, we build on prior work studying OthelloGPT: a small transformer model trained to predict legal moves in the simple board game Othello. We recreate linear probes to accurately extract board state, and demonstrate that simple, targeted interventions using directions learned by these probes can achieve near perfect edit performance: predictably steering the model to output only legal moves matching the edited position. We describe a novel technique for a "global intervention" where a full board state is substituted into the residual stream during inference. Linear probes are also used to extract and edit representations beyond board state, with interventions on next turn color coherently altering the model's board representation and outputs. Finally, challenges in intervening on an artificial feature of player directional bias highlight the difficulty of representational entanglement, even with highly accurate probes.

Contents

Background

OthelloGPT is the nickname given to an early 2023 paper by Kenneth Li et al[1], which gave compelling evidence that, rather than just memorizing surface statistics, transformers can learn "emergent world representations" from sequence tasks. They trained a small transformer (20 million parameters, fewer than GPT2-Small) on the task of predicting legal moves given a sequence of randomly generated moves from the board game Othello; despite never directly having access to the board state, the authors were able to use probes—classifiers using the model's residual stream[2] as input—to extract an accurate board representation from the model's activations.[3] Probes can be misleading[4] but the OthelloGPT authors make their case for a causally significant representation much more compelling by using the probes to intervene on the board representation during inference: flipping pieces in the model's board representation causes the model to output moves that are only legal in the modified position.

One caveat is that the OthelloGPT intervention technique is fairly involved. Their most successful probes were non-linear, so in order to reliably shift the model's activations to a different board state they use gradient descent directly on the residual stream until the probe gives the desired output. This optimization process must be applied repeatedly after each layer of the transformer during inference for the intervention to be successful.

In March 2023, Neel Nanda significantly strengthened Li's overall result by showing that these advanced techniques were unnecessary, and the board representation could be recovered from the model just using linear probes (projecting the model's residual stream onto 64x3 dimensions to predict the board state). This meant interventions could be performed by reversing the direction corresponding to a particular board square in a single layer.[5]

Together, these two results paint an exciting picture for mechanistic interpretability: transformers are able to learn interpretable, linear representations of features, and we can verify these hypotheses by making predictable interventions on models. However, the original OthelloGPT paper only shows their intervention in limited situations: swapping black/white for a single piece at a time. Nanda's interventions demonstrated a clear causal effect, but his exploration was self-admittedly brief and he did not achieve ideal edit performance; the probabilities assigned to newly legal moves after the intervention had significantly lower probability than the original legal moves.

[1] Emergent World Representations: Exploring a Sequence Model Trained on a Synthetic Task[2] Residual stream refers to the skip connections in the layers of a transformer model. Some authors have argued the residual stream can be thought of as the central feature of a transformer, accumulating outputs from layers and allowing components of the transformer (attention and MLP) to "communicate." [3] See this

article for an intuitive overview of the original OthelloGPT paper, written by the first author. [4] Neel Nanda gives a good, quick explanation of the difficulties of probes in his write-up on OthelloGPT; essentially: are they capturing some essential feature in the model or just computing it themselves? The challenges of interpreting probes have been long-known in the NLP community: see this post by John Hewitt about probing for linguistic structure in natural language models.
[5] This type of manipulation in the residual stream is akin to a simple form of activation steering.

Motivation

Nanda's interventions are compelling case studies, but many questions remain about OthelloGPT's linear board representation; just how far can we push these representations? Can we make arbitrary interventions, even those violating gameplay semantics (for example, swapping multiple pieces or modifying empty board squares to black/white)? In cases where these interventions fail, can we correlate those failures?

Beyond just board state, do we find disentangled representations if we probe for two different features? Can we intervene on one without impacting the other? What are the powers and limitations of probing residual stream activations and intervening as a general methodology for interpreting model representations?

By investigating these questions, this project aims to advance general understanding of learned representations in transformers. Hopefully these findings, along with the contributions of many other researchers working to rigorously understand small models, to improving measurement of model capabilities and detection of risks through interpretable decomposition.

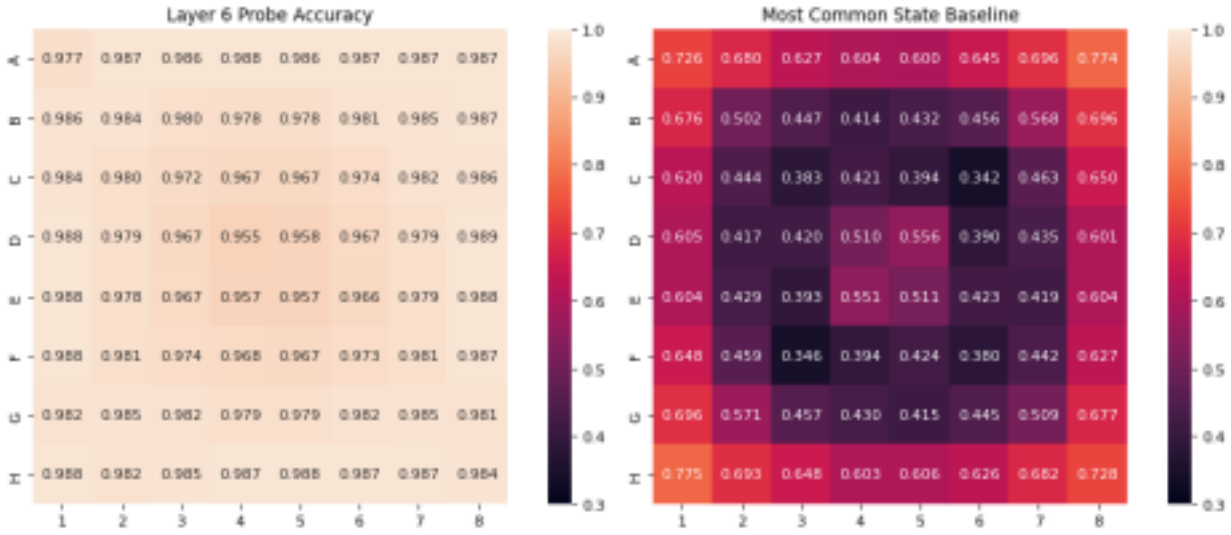
Recreating Linear Board Probes

Technical setup

I used an identical model architecture to the original OthelloGPT paper: a small transformer model with 8 decoder layers.[6] The model has a vocabulary of 61 tokens—60 tokens representing each of the open squares on the Othello board and 1 token for passing. I trained the model for 40 epochs on 5 million games of synthetic Othello sequences[7], where each game was generated by having both players repeatedly select uniform random moves. After training, the model was able to output legal moves in 99.8% of positions, even in games not seen during training.

In the original OthelloGPT paper, the probes were nonlinear, using a single hidden ReLU layer to predict the complete board state: an output of 64x3 logits (64 squares on the board and 3 possible states for each: black, white, or empty). The probes took the residual stream after a particular transformer layer as input, in this case a 512 dimension embedding. Neel Nanda had the key insight that a more native way for the model to represent the board is with a slightly modified set of states for each square: empty, the color about to place a disc, and the color who just placed a disc ("mine and theirs" versus black and white). Nanda demonstrated linear probes by training only on alternating moves in a game sequence, but for my own probes I directly

convert each board state from the original black/white representation to the model's perspective of mine/theirs.[8]



[6] Find the full OthelloGPT codebase here.
[7] This is less training data than the original paper (which used 20 million synthetic games), but 5 million games seemed sufficient to match the accuracy of the original OthelloGPT model, and I wanted consistency with other models I was training on modified datasets.
[8] This is more subtle than just switching every other state, as players will occasionally have no legal moves and are forced to pass, particularly in end game situations.

Figure 1.1.1: Probe accuracy for layer 6 compared to a simple baseline (most common state), which shows accuracy if the probe predicted the most frequent state for each board square.

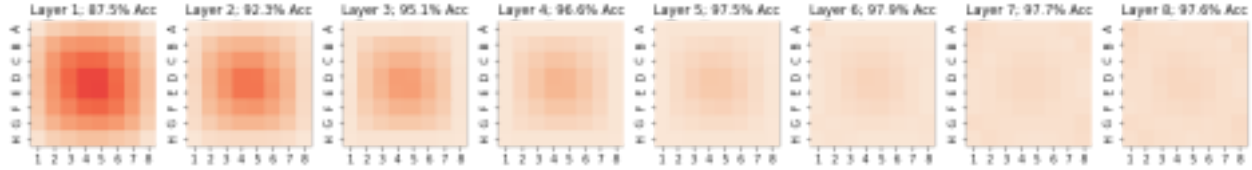


Figure 1.1.2: Probe accuracy for each layer; total accuracy included in plot titles. Probe accuracy peaks in layer 6, though layers 5-8 are all similar with >97% accuracy.

I trained separate linear probes on each transformer layer, using activations from running the model on all sequences from 10,000 games, with the modified board states as ground truth. The probes consist of a transformation matrix to project from the model's 512-dimensional residual stream to the board representation.

Linear interventions

With these linear probes, can we achieve equal performance to the complex optimization approach used in the original OthelloGPT paper? Their technique used gradient descent on the residual stream to achieve a desired probe output, and needed to be applied repeatedly from the middle layers all the way through inference to the model's output. With the linear representation, we can make our desired changes to the probe's output with a simple vector

addition. Given the weight matrix for a particular probe, we simply read off the direction in the embedding space corresponding to a given board square and state (I'll refer to this as a steering vector) and add that direction to the residual stream. All the steering vectors are normalized, so it takes some basic tuning to find a coefficient for the vector addition that successfully propagates the intervention through the rest of the model.

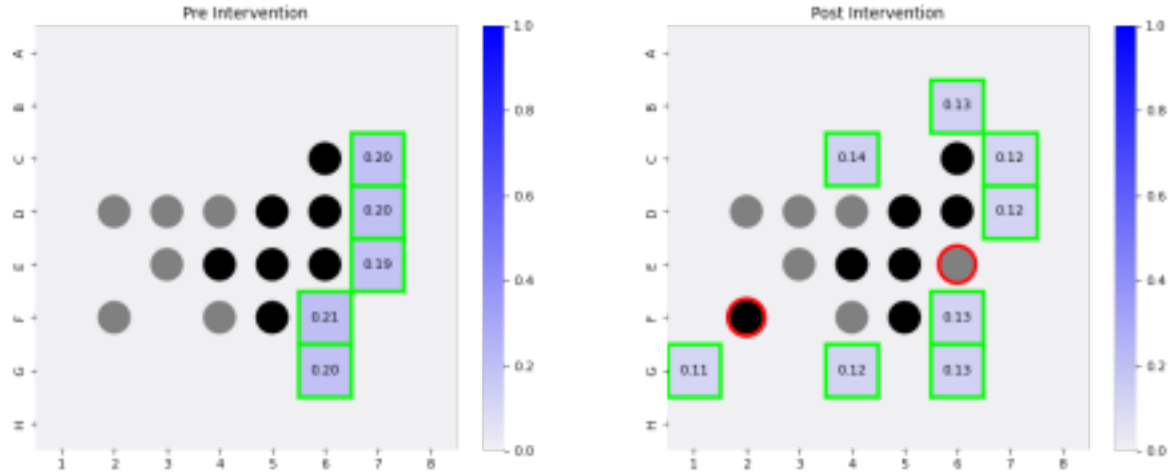


Figure 1.2.1: Example intervention, modifying the state of two discs (E6 and F2, shown in red). Legal moves in each position are highlighted in green. Heatmap is shaded proportional to the probability given to that board token after softmax.

In this example, we intervene in layer 4 to flip E5 from black to white, and F2 from white to black. The model's final predictions are near perfect edit performance—illegal moves in the post intervention board state are fully eliminated and new legal moves given equal probability—and it came from a single layer intervention, and required minimal tuning of coefficients. Even for a more complex intervention like flipping an empty piece to black or white (something that is strongly outside the training distribution for the model; within the training distribution, it is a perfect assumption that a board squares will be empty if it is not in the move sequence), near ideal performance can be achieved with steering vectors in only a single layer.

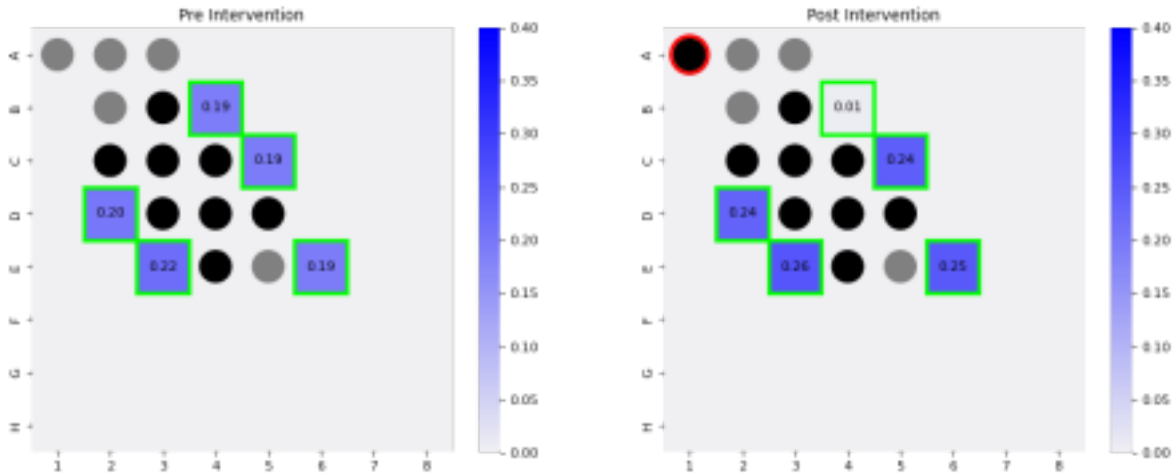


Figure 1.2.2: Example of unintended consequences from local intervention. Flipping A1 from

white to black should have no impact on the legal moves. However, because the probe's learned directions for A1 and B2 are not fully orthogonal, B2 is also flipped to black in the model's internal representation. Thus, the model erroneously gives low probability to B4.

However, despite the simplicity of local linear interventions, they can have inadvertent impacts. In the above example, the model's outputs are wrong with respect to our intended intervention, but consistent with its internal representation. It's unclear whether this is the result of the model's internal representation not being fully disentangled (each board square having an orthogonal direction to each other board square) or because the probe, trained to maximize accuracy, is picking up on spurious correlations in the dataset (such as B2's state being a weak proxy for A1's state). While this exact example could likely be corrected with more precise tuning of different steering vectors, performing more involved interventions (say, 10+ pieces being flipped simultaneously) with only localized changes is clearly infeasible.

Global Board State Interventions via Probe Inverse

With the current probe training process, local edits will inevitably have some spillover with other representations in the model (adjacent squares, etc). However, because we have a linear transformation from activation space to probe states, we can leverage this to perform a "global edit". Specifically, we can take the inverse[9] of our probe's transformation matrix, then use it to calculate a full set of activations from a desired probe state.

This means setting a desired log probability for empty/mine/theirs for each square, which I selected by looking at average log probabilities for each square state in different board positions (probably the most hacky/unrigorous part of this method). A matrix multiplication of this desired state and the probe inverse generates a new 512-dimensional activation vector, which fully overwrites the model's residual stream after a particular layer, potentially destroying any prior computation not captured by the board probes. Remarkably, performing this global edit in the middle layers of the model gives coherent outputs closely matching the desired board state.

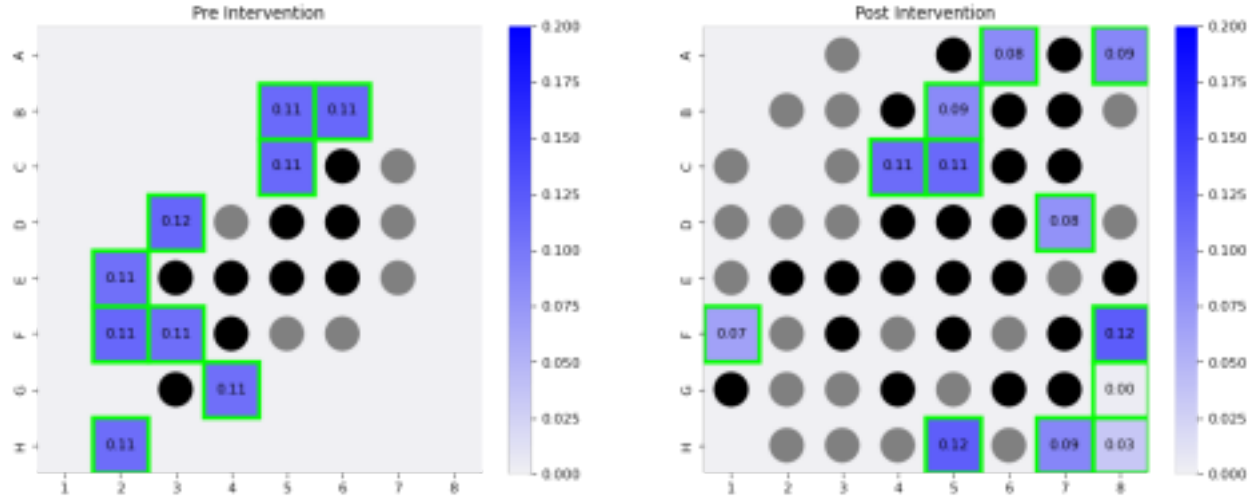


Figure 2.1.1: Global intervention on an input sequence of 10 moves, activations generated from

a different game's board state, inserted after layer 4. The post intervention probability distribution is not perfect, but the model still clearly separates legal from illegal moves.

[9] or pseudo inverse in this case.

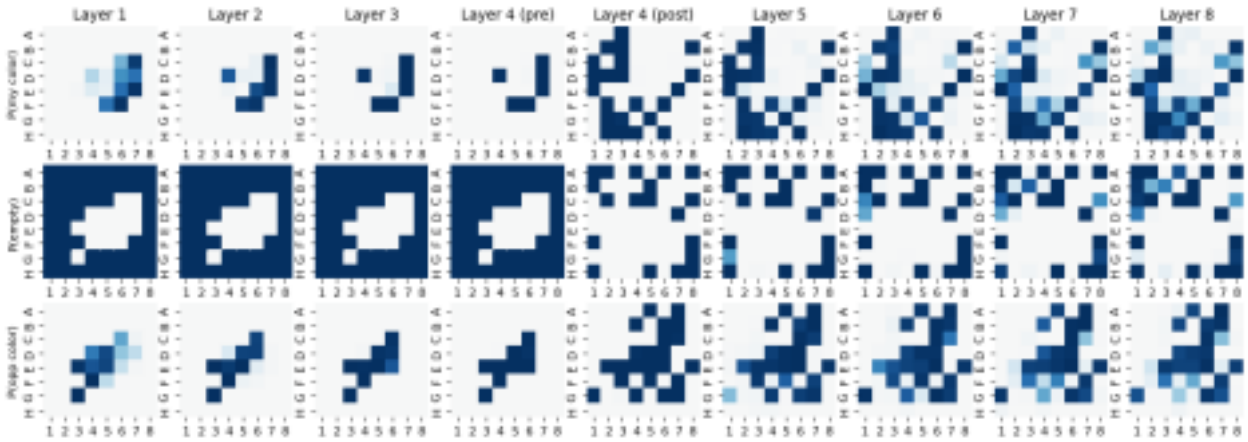


Figure 2.1.2: Visualization of the model's board representation before and after the global edit, according to board probes. We can clearly see the model gradually form an accurate representation of the input sequence, which is overwritten by our global edit after layer 4. After the intervention, the board state prediction gains some noise (likely a result of attention inputs from the true sequence conflicting with the generated residual stream), but remains very close to our desired global edit.

I conceptualize this process as giving the model extreme cognitive dissonance between its internal representation (residual stream) and updates from attention in later layers. I would have expected a full substitution of activations to be so out of distribution that it reduces the model to incoherence, so the relative coherence of the output gives intriguing evidence about the locality of the board representation. Perform a global edit too early in the forward pass and the board representation will mostly revert back to the true sequence; do it too late and the generated activations are missing essential aspects of calculating legal moves.

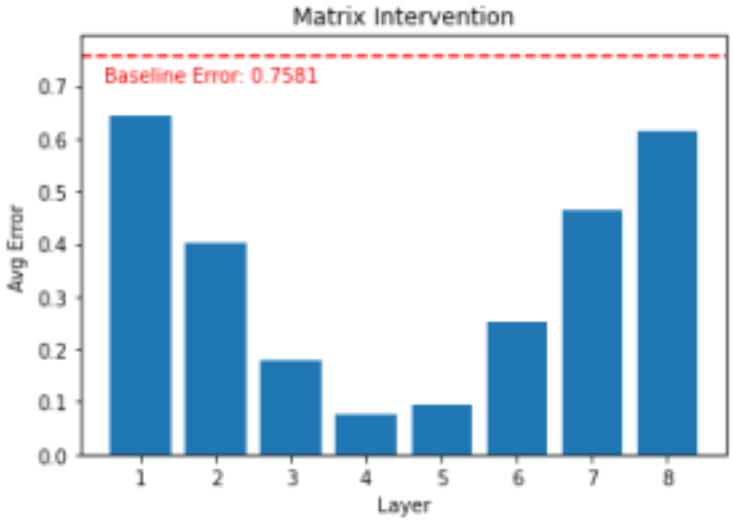


Figure 2.1.3: Aggregated error from systematically testing roughly 28,000 global interventions, shown by layer of global edit. Baseline is calculated as error (probability given to illegal moves) if no intervention takes place.

9

To get a more systematic view of global edit performance in different layers, I ran thousands of interventions across different randomized board positions. Specifically, the model was evaluated on all combinations of partial game sequences from two different games, with activations in a given layer being overwritten by the board state of the second game sequence. Error was aggregated by the layer of the intervention, which interestingly shows layers 4 and 5 as the cleanest points for global changes. While probe accuracy in predicting board state is highest in layer 6, this causal analysis indicates that earlier layers might be more suitable for gaining a deeper understanding of the model's computation and its utilization of the board representation, as these layers are more predictably impacted by interventions.

10

Other Representations in OthelloGPT

Probing for next color direction

While the board state representation has been my main focus, it seemed worthwhile to investigate other features the model may have learned to represent in its embedding space; can we extend the overall methodology of probing for some feature we might expect the model to represent for this task, then attempt causal interventions to confirm the feature is used in model computation?

One logical feature is the next player to move (black or white). Black always starts and players alternate placing pieces, so it might seem that calculating the next color is as trivial as computing the parity of the length of the input sequence. However, a player may have no legal moves and be forced to pass leaving one color placing two pieces in a row. These situations are

rare in human play, but appear occasionally in our randomly generated dataset, meaning it's reasonable the model would learn an explicit turn representation to handle this complexity.

To test this, I trained a probe to predict the next color to play, using nearly an identical setup to the board state probes. Rather than predicting a 64x3 board state, the turn probes project the residual stream onto two classes: black to play and white to play. I again trained a separate probe on each layer using activations from 10,000 games, and the probes quickly achieved >99% accuracy. As we might expect, the "black to play" direction and "white to play" direction are nearly the same vector, just reversed.

Interventions on turn are similar to the local steering interventions for board state, where to switch from white to black we simply add the "black to play" direction with a large coefficient. Intervening on turn in early layers is quite effective. The example below shows a typical turn intervention, with the steering vector applied in layer 1. The set of moves given high probability is nearly perfect in the post intervention output, but two erroneous moves are also given equal probability. A closer look at the board representation in the layers before and after the intervention shines light on this curious error.

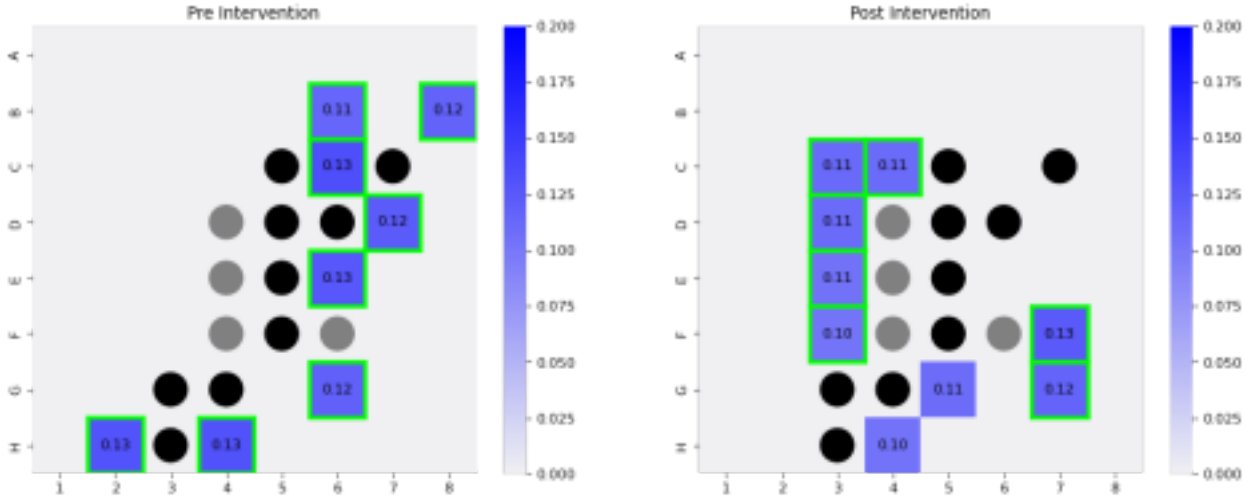
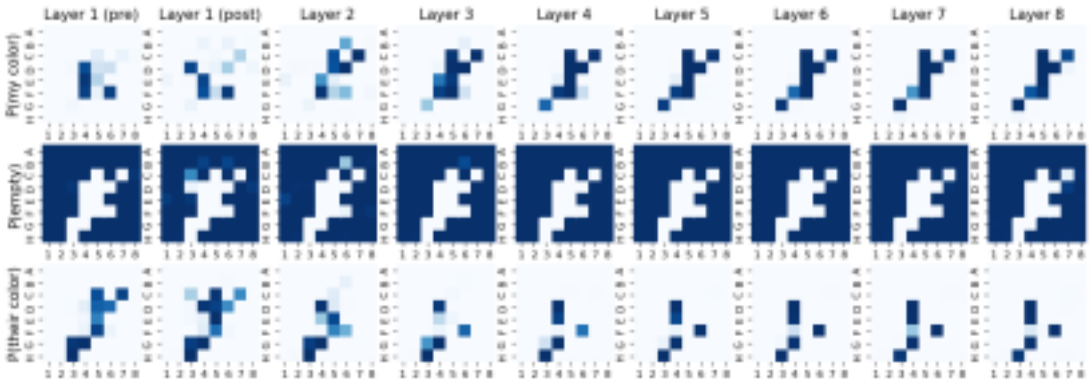


Figure 3.1.1: Example intervention on next color, swapping from white to black after layer 1.



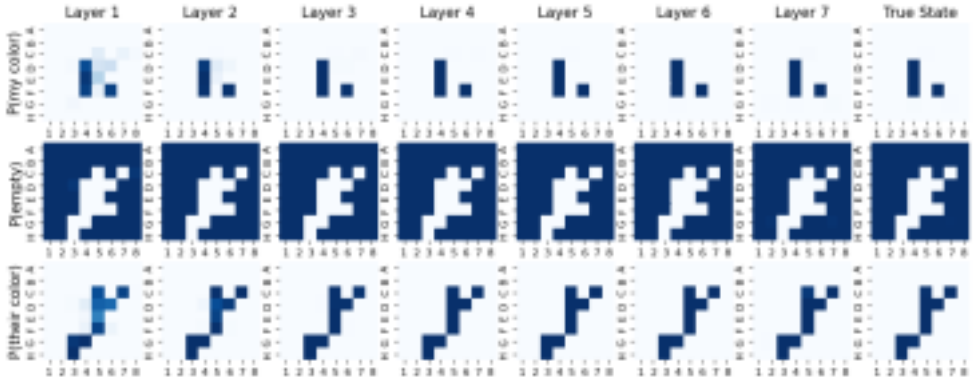


Figure 3.1.2: Board probe prediction across layers for the above intervention. The final column shows the true board state. Note how the model reconstructs the board representation after turn direction is flipped, but it does so imperfectly.

The next color intervention does not immediately impact the board representation, but clearly prompts a change in how the model is computing the board state through the middle layers. Interestingly, while the board state does update to reflect the new color order, it is not a perfect inversion of the true board. This suggests that the disrupted color order interferes with the model's typical process for incrementally calculating board positions. Yet, the model shows surprising resilience in recovering a reasonable board position, even when making a turn intervention as late as layer 3.[10]

[10] For a more detailed look at the impact of turn interventions in later layers, see Appendix C.

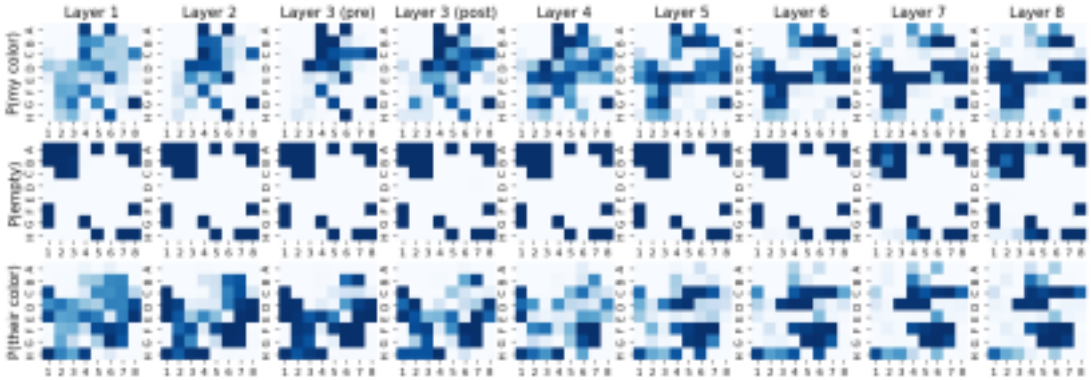


Figure 3.1.3: Turn intervention after layer 3 in a more complex board position, with the final column again showing the true board state. Despite having to totally reconstruct its board representation after layer 3, the model reaches a relatively coherent result.

Introducing an artificial feature (player type)

To further explore model representations, I generated a modified dataset with a new, artificial feature I'll refer to as player type. Rather than just selecting uniform random moves during game generation, a random player type is selected with a bias towards one of the corners of the board. This bias is implemented as the player selecting their preferred move—the one closest to their corner[11]—80% of the time, and choosing a random legal move the remaining 20% of the time. I generated 5 million games to match the original synthetic dataset, and trained a new model with the same architecture on this biased dataset. The playertype model outputs legal moves 99.8% of the time on its validation set, and 98.6% of the time on the unbiased dataset.[12]

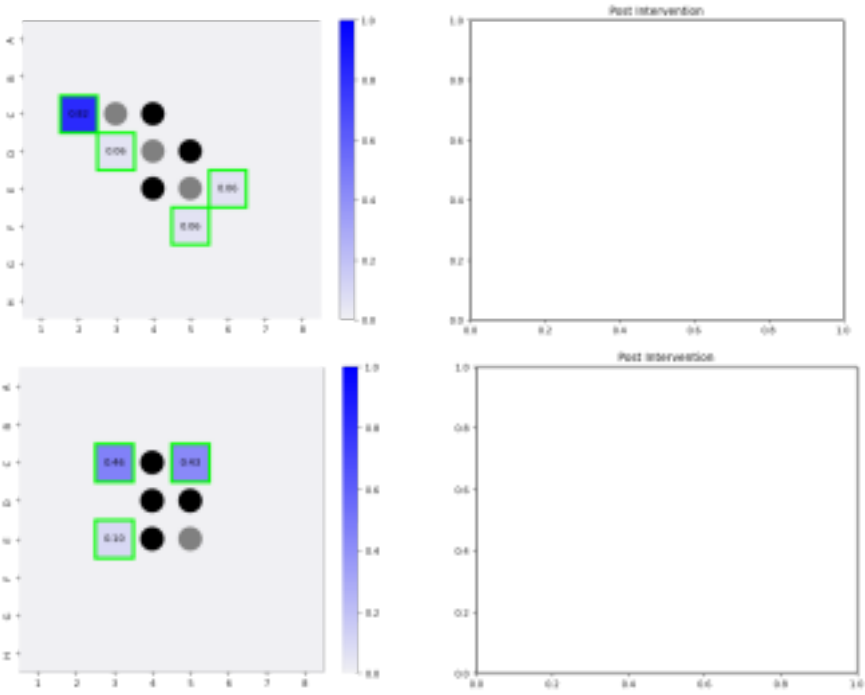


Figure 3.2.1: Predictions from player type model for first 3 moves of game, which is biased towards the top left corner. The model clearly identifies the top left bias and makes consistent predictions.

[11] To avoid ties for preferred (which using a distance metric like Manhattan distance would cause), each player has a preference ranking of all board squares, starting at their corner and incrementing by row. This means the top-left player type is essentially biased up and then left. For example, square A8 is considered more “top left” than square B1.
[12] The unbiased (original) synthetic dataset is essentially randomly exploring the game tree of Othello, so this is a fairly good metric for how well the player type model has encoded the base rules of Othello.

With the trained model, we can probe for representations. Linear board state probes trained on the playertype model achieve high accuracy, despite the model only being trained on a highly biased dataset, and local interventions are still effective.[13] We can probe for player type using the same setup as board state or turn, but now predicting 4 classes: one for each preferred corner. The player type probes achieve high accuracy (peaking at 99% in the middle layers), which isn’t particularly surprising when the model has a board representation and player type is so correlated with board state; more tiles in a corner means the sequence is more likely to be biased towards that corner. However, we can help quantify how much just this correlation contributes to probe accuracy by training player probes on the original, unbiased model. These baseline player probes peak at 93% accuracy, leaving a significant gap that’s suggestive the model could be directly representing player type.

To test this, we can try intervening on player type. This has mixed results. With some tuning of coefficients I was able to get some success in changing the most preferred move, but the model never fully deprioritized the previous preferred move. I was unable to find a tuning to fully reverse the model’s preference (i.e. flipping from top left preference to bottom right preference). My best attempts also required repeatedly applying the steering vector on multiple layers, as a single layer intervention would revert to the original outputs.

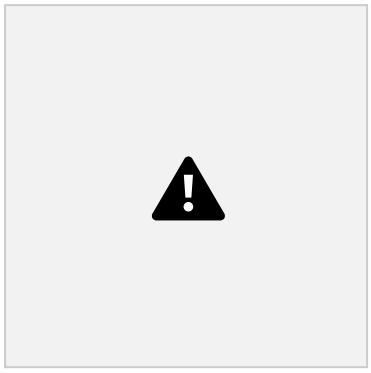


Figure 3.2.2: Left and center charts show model predictions with and without a linear steering intervention on player bias from top left to top right. Right shows log probabilities with and without intervention to more clearly highlight the impact of the intervention.

The intervention is successful in making the top-1 prediction the closest square to the top right,[14] but the model still prefers the top left move over other legal moves. Trying a higher coefficient on the steering vector reduces the model to incoherence (all moves given similar probability, regardless of legality).

Visualizing the internal board representation during a player intervention is revealing. Unlike the turn intervention, the player intervention has an immediate impact on the board state. It appears that either the model or the probe has not fully disentangled the concept of player type from the board state. In the layers after the intervention, the board representation “recovers” to the original true board state, which could explain why attempting player type interventions only in early layers has no apparent effect on the model’s outputs. This representational entanglement is reasonable given the consistent correlation between player type and board layout in the training data. Since a player's type does not change mid-game, concentrations of their pieces in a preferred corner perfectly align with their type bias.

[13] See Appendix E

[14] As previously mentioned, the player is biased towards top squares, then towards right squares, so it would prefer A3 to C5 (even though C5 appears visually closer to the top right corner)

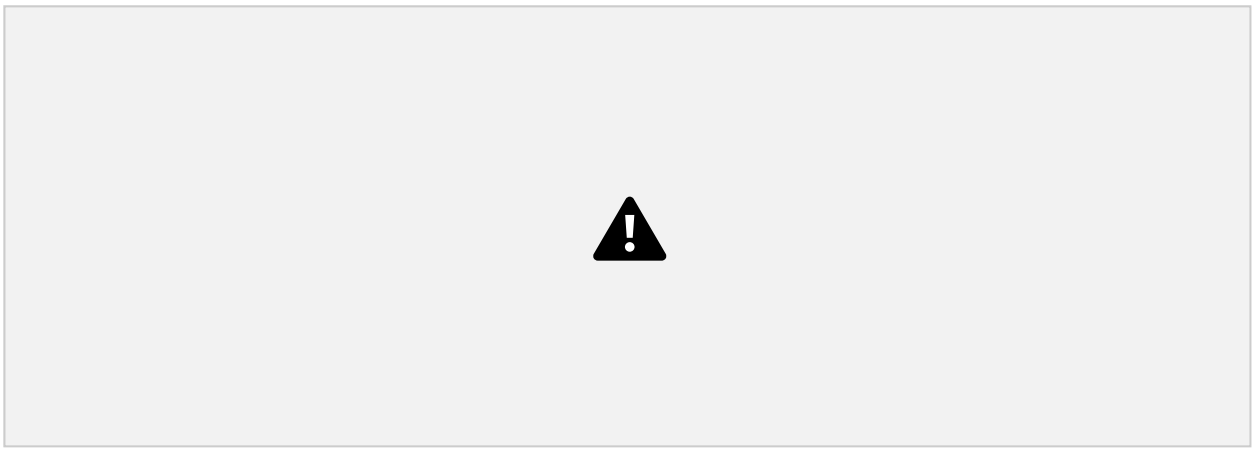


Figure 3.2.3: Board representation during an attempted player type intervention from top left, the true bias, to bottom left. Intervention immediately causes massive interference with board representation which is gradually corrected and the model gives an eventual standard output.

Future Work

This investigation reinforces the power of linear probing and causal interventions in deeply understanding transformer models. The larger linear representation hypothesis offers a tantalizing vision for decomposing and manipulating models at the level of interpretable concepts. However, this report also shows how straightforward probes can mislead, and have clear weakness in identifying fully disentangled representations. Additionally, many of the key takeaways are still just observatory, and more rigorous quantitative evaluation metrics for intervention performance could enhance these results. A truly exciting result would be a definition precise and general enough to extend beyond just the toy task of Othello, and open up large-scale intervention testing to map limits of model representations without simply recording limitations of the toy world.

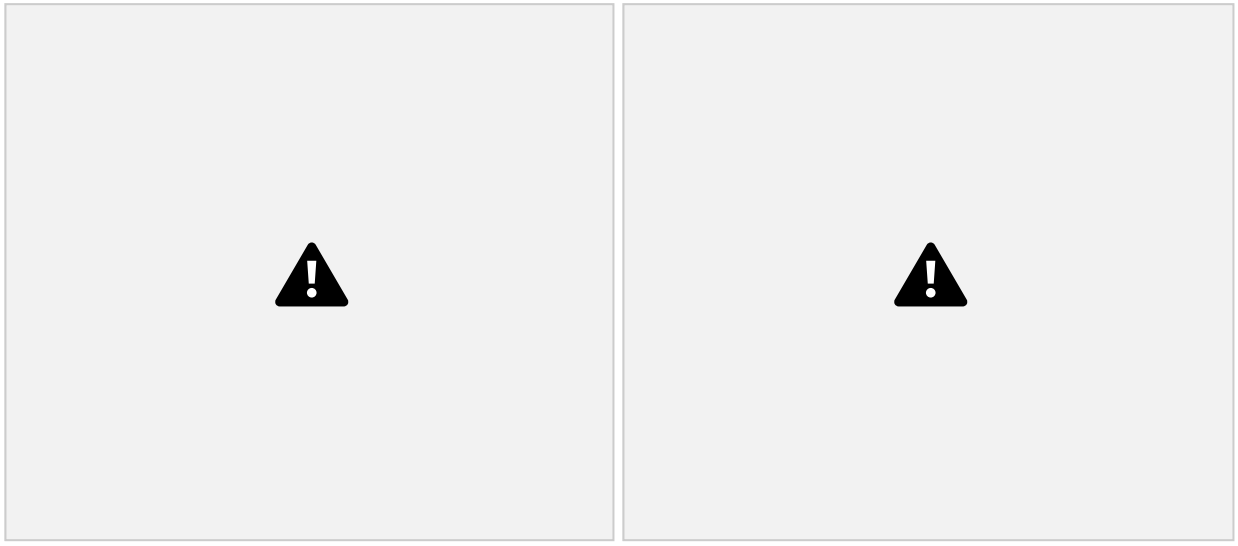
Expanding beyond residual stream interventions to a more granular, circuit-based-approach could also be valuable. Neel Nanda's insight for a linear board representation was inspired by neuron level study of the original OthelloGPT model, and tools like OthelloScope give the beginning of more exhaustive study of how model's compute board states.

Building on the player type experiments, further work could examine how robust model representations are to biased or corrupted data. If world representations are fundamentally less interpretable when trained on less friendly data, can fine-tuning help recover the ability to intervene or just mask the alien complexity?

Acknowledgements

I am deeply grateful to David Reber, my research mentor for this project, for invaluable advice and guidance throughout the research process. I'd also like to thank Victor Veitch for detailed feedback on ideas, and for granting access to the DSI cluster which allowed me to train and test models. I am grateful to Miles Wang and Elias Baldwin for insight and writing feedback, and also to Zachary Rudolph for organizing and running the XLab research fellowship at UChicago, without which this project would not have been possible.

Appendix A: Additional Probe Baselines, Zero-Shot Transfer

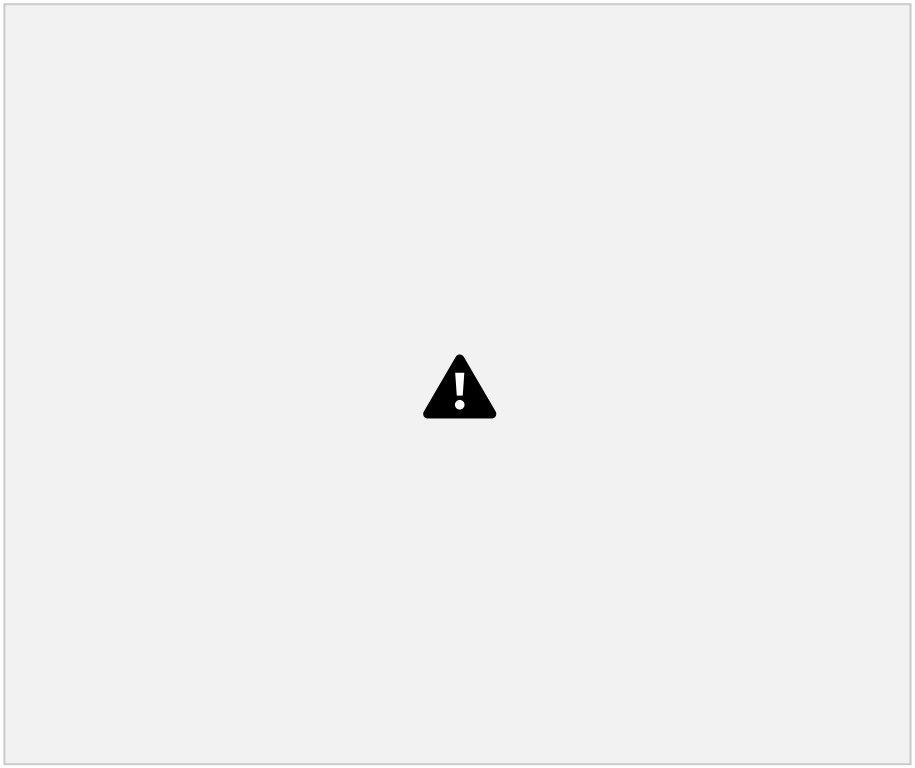


Most common per turn baseline still predicts the most common state for each board square, but is able to take turn into account (i.e. on turn 5 selects the most common state for turn 5 based on training dataset). The probes do not directly have access to the turn, so it isn't a proper baseline, but intuitively feels like a better comparison as sequence length is likely an easily extracted feature in the model's activations.

These plots show accuracy per board square for each layer's probe predicting zero-shot on activations after layer 6. Accuracy is higher across all the probes than predictions based on original layer activations, suggesting that layer 6 is indeed where the board state is most crisply represented? The probe's learned directions for each square are not identical across layers, but looks more like random noise than some structural difference.

These show the inverse: the probe from layer 6 transferring zero-shot to the activations after layers 1-8. The layer 6 probe matches the early layer probes performance on early layer activations, but we see a clear drop in accuracy and more noisy pattern in layers 7 and 8. Perhaps the board representation changes as the model computes the final legal moves?

Appendix B: Cosine Similarity of Board Directions

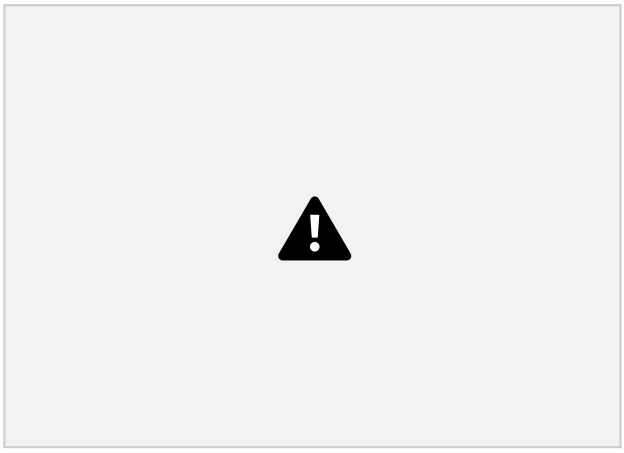


Cosine similarity for each pair of board squares (numbered 0-63, left to right by row). While an

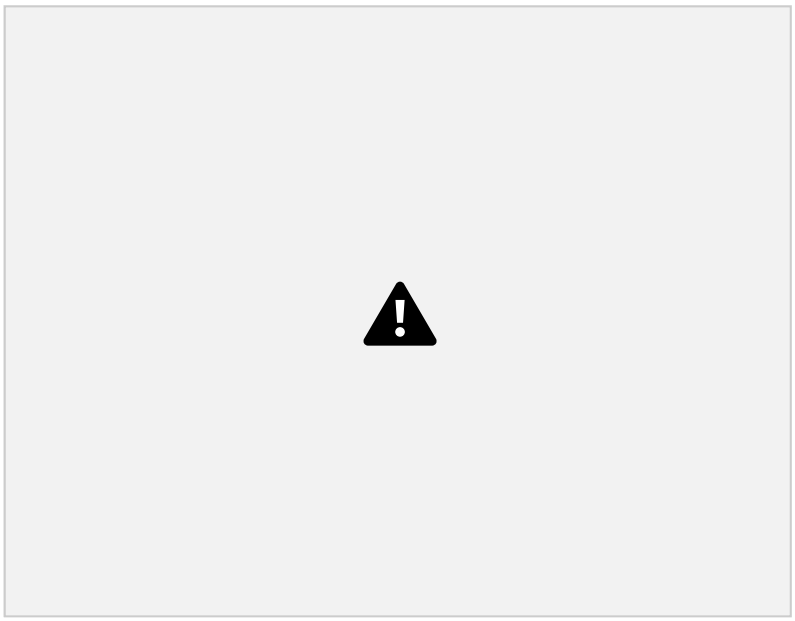
idealized representation of the board might have each direction fully orthogonal to all others, there is clear structure with adjacent board squares having higher similarity.

These charts highlight cosine similarity for two specific directions: A1 and D3 (shows the same information as a single row in the above heatmap, just reshaped to more clearly see the structure of correlation between adjacent squares). We can intuitively see how the probe for D3 gets weak signal for the state from D3 from adjacent squares and diagonals.

Appendix C: Aggregated Error for Turn Interventions



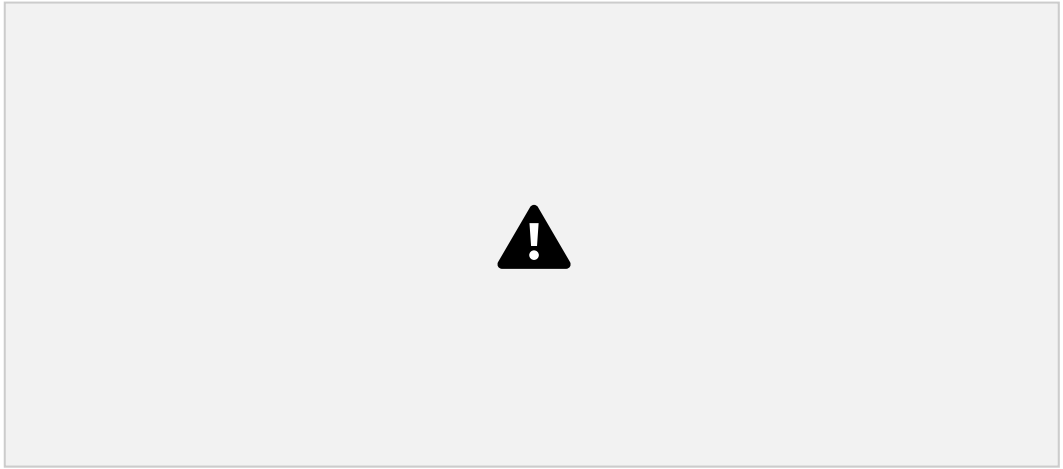
The above chart shows the result of attempting many turn interventions (flipping white to black or black to white when appropriate) in layer 1, with error aggregated by the length of the input sequence (the turn number of the particular game). Shifting the model's representation in the first several moves of the game is particularly unsuccessful, but further into games it works consistently (baseline error is roughly 0.7).

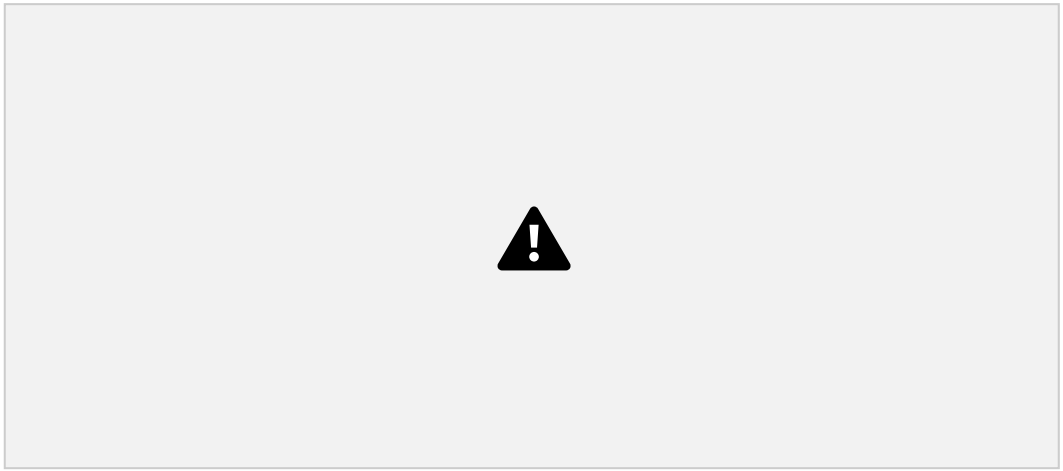


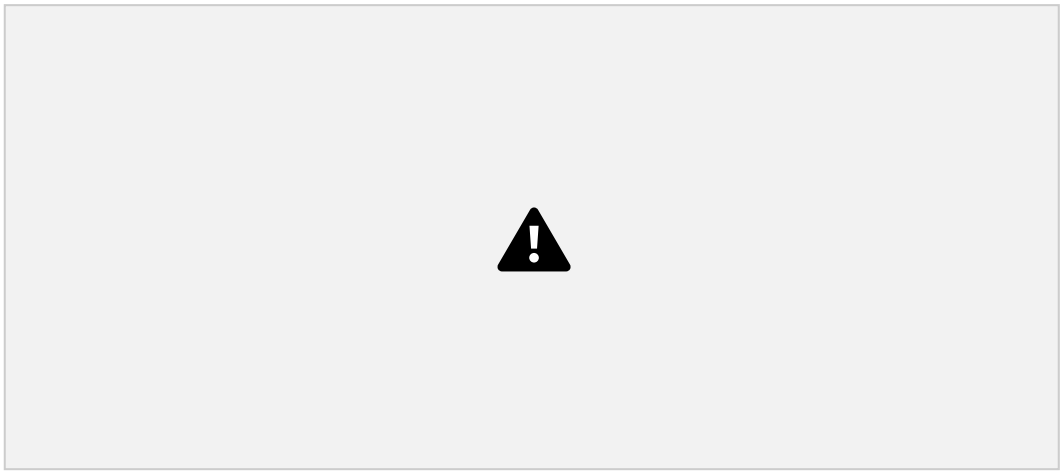
Shows the effectiveness of turn intervention using different scales (coefficient of the steering vector) on different layers. A scale of 2x in layer 1 gives the lowest average error, but larger scales are necessary to successfully shift model outputs in later layers. This is relatively unsurprising, as the turn interventions seem to work by shifting how the model computes the board state, and so it becomes more and more out of distribution to upend the board representation in later layers, necessitating a stronger steering vector.

Appendix D: Player Probe Predictions Across Unbiased Games

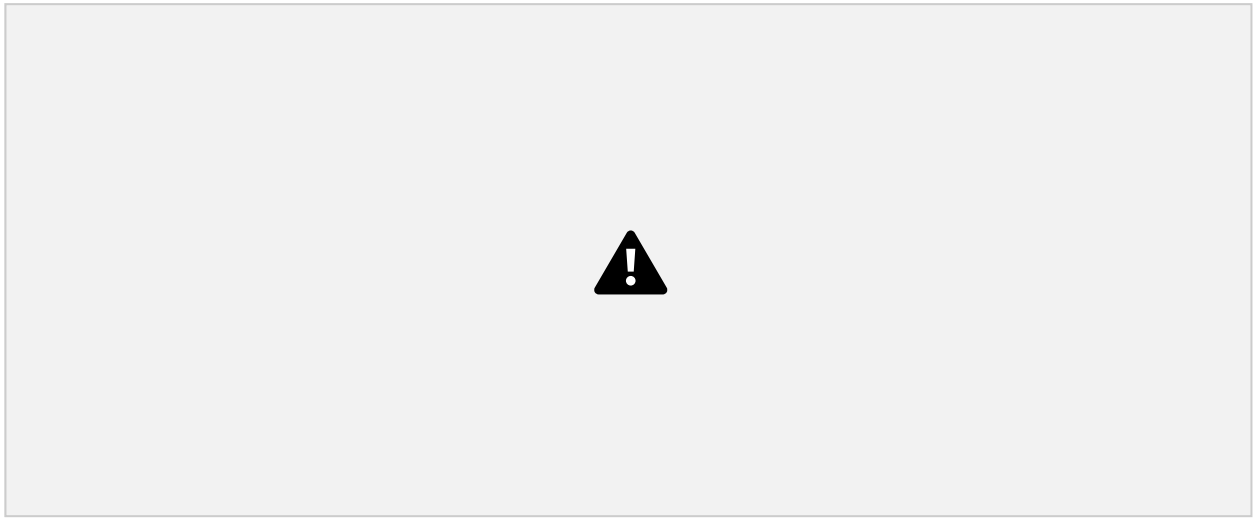
Results from player type probes when running the player type trained model on unbiased games from the original OthelloGPT dataset. These games have no true player type, so it is logical that the probes give ambiguous and varying results at different points in the sequence. The resulting charts give a view into probes generalizing to ambiguous situations, and are visually pleasing.



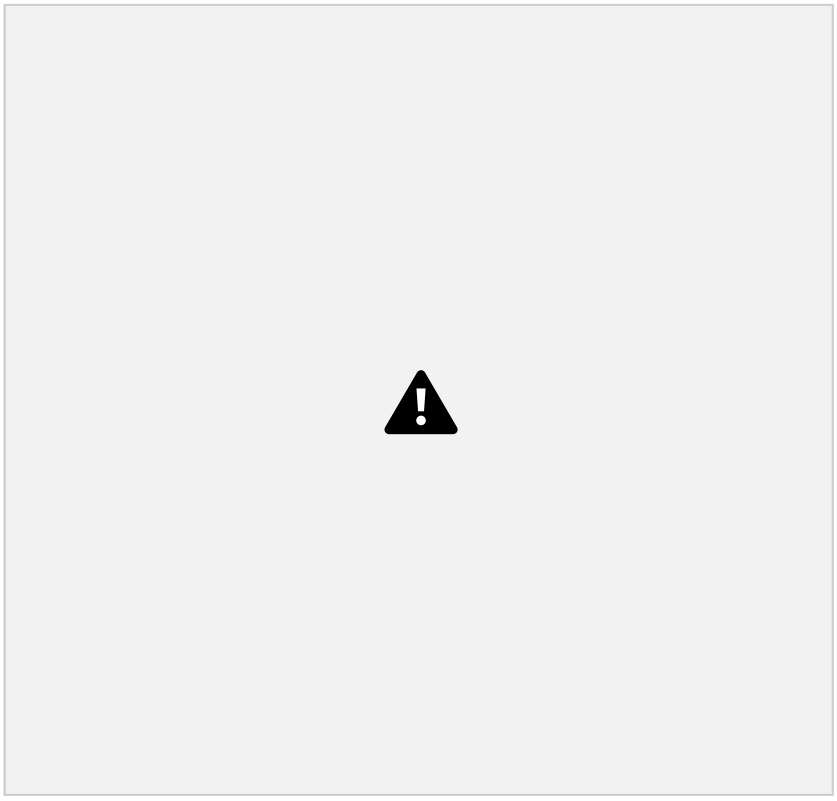




Appendix E: Board State Intervention on Player Type Model



Model predictions with and without intervention on C2, flipping black to white. Model maintains top left bias before and after intervention.



This scatterplot shows log probability given to each board square with and without intervention, with key board squares highlighted. Three distinct clusters can be observed pre and post intervention: illegal moves, legal moves, and the preferred move.