Examining the Interaction of Interpretable Features and Training Dynamics in Othello-GPT

Joshua Engels [*1] **Julian Yocum** [*1] **Sean Wang** [*1]

Abstract

Model interpretability continues to grow as a subject of research and has seen some success in developing an understanding of the inner workings of models, for example in recovering a "world model" in a transformer trained on valid Othello moves. While these methods have revealed interesting properties after training, little work has examined interpretability over the course of training. In this project, we perform multiple investigations that examine the interaction of training and interpretability. Specifically, we examine the development of a world model in Othello-GPT as we vary dataset size, problem complexity, and the number of training epochs. We find that feature interpretability (the quality of linear probes trained on Othello-GPT) increases monotonically across all of these axes. A second training dynamic we investigate is how interpretable features can be intentionally promoted by the choice of loss function. We find that we can add an additional loss function that explicitly encourages the development of a world model and that this results in better interpretability in the final trained model without affecting validation loss convergence.

1. Introduction

Project video link: https://youtu.be/Iv74qm68ab8

One recent area of research in understanding neural networks is mechanistic interpretability (Olah et al., 2020): understanding model behavior by directly examining model internals (e.g. activations and weights) at a low level. A common mechanistic interpretability method is *probing* (Alain & Bengio, 2016), which involves training a (usually linear) classifier on a set of internal model activations to recover features of interest. Features are a general term corresponding to useful and understandable activation patterns in a model's hidden layers; simple examples might include edge detectors in CNNs, a "cat" direction on ImageNet, or a "German" direction in language models.

To the best of our knowledge, all probing work so far has focused on probing already trained models. In contrast, we are interested in how training and probing interact. Specifically, we ask how model interpretability varies with the training set size and complexity if we can improve model interpretability with probing, and if we can use probing to train models using less data.

We focus on a line of work (Li et al., 2022) (Nanda, 2023) that trains a transformer model to make valid Othello moves, and then trains probes on internal activations to recover the board state, Othello-GPT's "world model." We propose a series of experiments that examine 1. under what circumstances this world model develops at all, 2. how this world model develops during training, and 3. how encouraging or hindering the development of this world model affects training.

2. Literature Review

2.1. General Interpretability

Broadly, interpretability research aims to reverse engineer features within models that can be interpreted by humans. In addition to probing techniques, recent work by Anthropic demonstrates that features can be discovered in an unsupervised manner using auto-encoders (Bricken et al., 2023). In addition to feature discovery, there are some limited results on methods to promote the interpretability of features of models through architectural choices. For example, Anthropic finds that replacing the activation function with softmax linear units increases the number of interpretable neurons in MLPs (Elhage et al., 2022a).

There has also been some limited research on how features develop during the training process. The authors

[*]Equal contribution [1]Massachusetts Institute of Technology, Cambridge, MA. Correspondence to: Joshua Engels <jengels@mit.edu>, Julian Yocum <juliany@mit.edu>, Sean Wang <swang07@mit.edu>.

of (Hu et al., 2020) find that the early training dynamics of a model resemble a simple linear model; in other words, during the early part of training the model finds a "simple" solution to the problem. Additionally, in Anthropic's initial work describing superposition (Elhage et al., 2022b), the authors find that feature development can follow a phase change pattern, where validation loss drops sharply and features separate all of a sudden in a piecewise fashion.

2.2. Othello-GPT

We build off of recent work proposing Othello-GPT (Li et al., 2022) (Nanda, 2023). This line of research models the game Othello as a sequence of moves and trains a GPT(Brown et al., 2020) model with 8 layers, 8 attention heads, and a 512 dimensional hidden dimension to predict the next move in the sequence. Intuitively, the model should learn to assign an equal probability to the possible valid next moves in any given Othello game, and 0 to all other possible moves.

The authors of (Li et al., 2022) find that they are able to apply non-linear probing with one hidden layer and ReLU activation function to recover the board state from the hidden layers of the trained model with high accuracy. In follow-up work, (Nanda, 2023) finds that a *linear* probe is actually sufficient if, instead of training a probe for determining if there is a white or black piece on a cell, one trains a probe that asks if there is a piece of the current color or the opposite color corresponding to the next move.

3. Methods

Synthetic Dataset Generation. To examine dataset-dependent features of model training, we construct additional synthetic datasets for Othello boards of size 4×4, 6×6, and 8×8. For each of these board sizes, we construct training datasets of size $N = 10^4, 10^5, 10^6$, and 10^7, where each dataset consists of randomly sampled valid Othello games. To generate a random game, we start with the usual Othello starting state in the center of the board, and simulate random valid moves until the board is entirely filled. Some games end before the board is filled because players cannot make a valid move; in this case, we ignore the game and resample another.

Training and Evaluation Process. We use the same training loop as the original Othello-GPT paper (Li et al., 2022) with a single minor modification: so that we can later evaluate the interpretability and validation loss at different points during training, we save a checkpoint of the model every 10 epochs, resulting in a total of 26 checkpoints across the 251 epochs (we start at epoch 1). We train an Othello-GPT model on all 12 combinations of Othello board size and training dataset size defined above, resulting in $3 * 4 * 26 = 312$ model checkpoints.

Once the models have finished training, we evaluate two primary metrics on the checkpoints: the standard metric of loss on a validation dataset, and a novel metric we term "interpretability score".

Evaluation Metric: Validation Loss. To determine the loss on the validation dataset, we first generate an additional 200 synthetic games the model was not trained on. For every move in each game, we generate the valid potential next moves and create a ground truth label y that is $1/n$ for the n valid moves and 0 for all other moves. The validation loss of the model is the sum of the model's cross entropy loss across all such y when prompted with the move history. We note that we also briefly examined a "top 1 accuracy" metric, which evaluated how often the model's top activating neuron was a valid move on the validation dataset, but abandoned this technique for cross-entropy loss because cross-entropy loss is more fine-grained.

Evaluation Metric: Interpretability Score. We now define the interpretability score. We turn the rough probe training and probe evaluation notebooks from (Nanda, 2023) into a single function that takes in a model operating on boards of arbitrary size and trains linear probes on layer 6 in the model (this layer is an arbitrary choice for consistency; our code works with any choice of layer). In the function, we train $num\_rows \times num\_cols$ linear probes to predict the board state corresponding to each cell. Thus, each linear probe has three outputs–blank, "my color", and "their color"–which we softmax so that they sum to 1. Finally, the function also evaluates the probe on a testing dataset. The "interpretability score" that is returned from the function refers to the overall accuracy of this evaluation, or in other words how often the highest activating class from each probe (averaged across all games and all moves) is the correct board state.

One interesting thing to note about this process is that we actually obtain slightly better probes than (Nanda, 2023). This is because we knew in retrospect the correct framing of the problem as the "current" vs. "opposite" color, so we changed the representation of the board state to follow these labels, as opposed to doing this in a post hoc way on separate probes trained on even and odd moves.

To summarize, we generate a score from 0 to 1 describing how interpretable the model's internals are. A score of 1 means that we could train linear probes to reconstruct the exact board with 0 error, whereas a score of around 0.5 means that we could do no better than a random model.

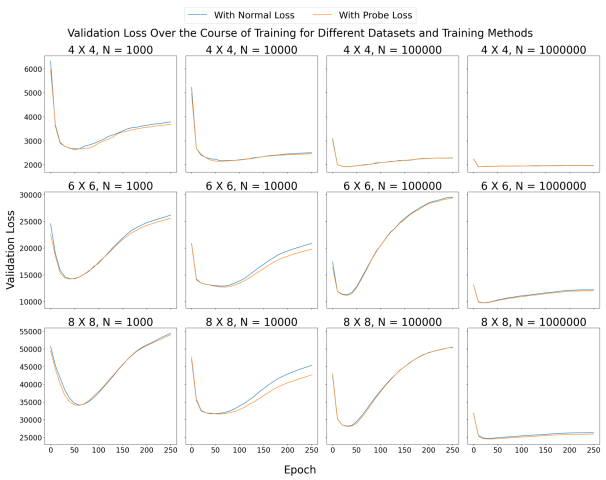

Figure 1. Validation loss versus number of epochs for each combination of datasets and training sizes; a lower loss is better. Training size increases from top to bottom, while dataset size increases from left to right. We observe that both training loss methods result in almost identical validation loss curves.

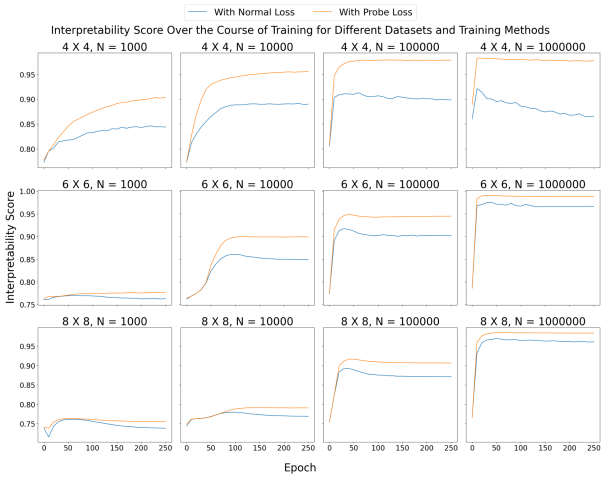

Figure 2. Interpretability score versus number of epochs for each combination of datasets and training sizes; a higher score is better. Training size increases from top to bottom, while dataset size increases from left to right. We observe that across all settings, the probing loss method is more interpretable.

Probing Loss. Finally, we investigate the extent to which we can influence feature development during training. We hypothesize that by choosing a suitable loss function, we can promote or penalize the model's internal feature development. A method that promotes feature development may promote the interpretability of the model and speed up the development of the "world model." We hypothesize that this may even act as an inductive bias and reduce the training time to reach accuracy thresholds (although as described below in our results section, we do not observe this effect in our experiments).

To promote a world model in Othello-GPT, we augment the original "next move" error loss function with an additional probing loss term by adding a trainable linear probe to layer 6. The linear probe's loss is set to be the cross entropy loss between the probe's predicted cell state and the actual cell state.

We add this new loss with a λ weight. In practice, we found that $\lambda = 1$ worked well in practice. This loss function can thus be written as

```
def loss(model, x)
    logits, probe_logits = model(x)
    return CrossEntropy(logits, y) +
           lambda * (-probe_logits * z)
```

where y and z are the valid next moves and board states respectively.

Like for the normal loss model, we train this alternative probing loss model on all 12 datasets, resulting in 312 more model checkpoints.

4. Results

Figure 1 plots the validation loss of both training methods (normal loss and probing loss) throughout the training process on each training metric and dataset size. Similarly, Figure 2 plots the interpretability score of both training methods throughout the training process on each training metric and dataset size.

The results in Figure 1 show that the validation losses between the two methods are almost precisely the same through the training process. In other words, adding the probing loss during training does not make the convergence behavior of our model worse, but it also does not make it better. The one exception we saw was on the 8×8 board, where we saw a very slight convergence improvement from using the probing loss (we reach the minimum loss in about 10 fewer epochs). In terms of the general validation loss curves, we see what we expected: the minimum loss decreases as we increase the dataset size, and the loss curve overall reaches a minimum before increasing as the training starts to overfit.

Our results in Figure 2 show that across all datasets, training the probe loss increases the interpretability of the resulting model, both during and after training. Additionally, for both the normal loss and probing loss training techniques, the interpretability score increases as we increase the number of training epochs, as we increase the dataset size, and as we increase the number of training samples.

5. Limitations and Future Work

Our immediate next steps aim to extend our experiments in ways we simply did not have time for during our class project. We first plan to try with larger λ values to see if this causes a change in the convergence plot; we expect that eventually when λ is large enough, the convergence plot will have to change in either a positive or negative direction.

Drawing inspiration from existing work on Othello-GPT regarding intervention, we can also modify the interpretability score to take into account the causal role of the world model. For example, we may measure how often applying a weighted update in the opposite direction of the linear probe causes the model to make the "wrong" predictions that are consistent with this update. Additionally, we plan to apply the interpretability score function to layers other than the layer to which we apply the probe loss. Both of these methods will result in a more robust interpretability score, increasing confidence in our results.

One limitation of our work is that we only study the world model of Othello-GPT and not the algorithm it uses to turn this world model into a prediction. We hypothesize that this may be a possible reason that we did not see an improvement in loss even though we saw an improvement in the interpretability of Othello-GPT's world model. One set of next steps we are interested in is mapping out the circuit that Othello-GPT uses to run this algorithm, and encouraging the development of this algorithm during training.

Another avenue of future work is to look at different board sizes to see if the results for 10 by 10 or 12 by 12 boards continue to follow the trend we observed. Furthermore, all of the data used in our research was synthetically generated - using agentic game data, e.g. from humans or trained machine learning agents, may produce different results.

Finally, we are interested in methods that penalize feature development to train a model that learns to obfuscate its feature representation. Demonstration of the ability of models to obfuscate their representation in this way could have profound implications for the feasibility and story of interpretability. To accomplish this task, we propose a method inspired by the generative adversarial network: we will again connect a probe, as in our probe loss experiments, but now train the model for a step, then freeze it and update the probe, and then unfreeze the model and backpropagate a loss from the new update of the probe.

6. Teammate Contributions

The work of this project was divided among three teammates. All contributed equally to writing the paper and recording the video, and a breakdown of responsibilities further is included below.

Josh Engels wrote the probe training and evaluation code (the "interpretability score" function), wrote the code to generate simplified state representations in terms of "mine" and "their" pieces, created the final experiment pipeline and ran it, and wrote the figure plotting code.

In order to promote interpretable features, Julian Yocum extended the original Othello-GPT model to have a linear probe head and then altered the training loop to include the second probe loss term. He also enabled checkpointing of the model every epoch.

While the original Othello-GPT repository was only made for 8×8 boards, this project investigated the effect of varying the dimensions. In order to do this, Sean Wang modified the code base so that it was compatible with more general $N \times N$ boards. Additionally, he developed the pipeline for the generation of the synthetic datasets.

7. Conclusion

In this paper, we explored the interpretability of Othello-GPT models. In particular, we addressed these three questions: how does interpretability depend on dataset size and board complexity; how interpretability changes over the course of training; and what are the effects of using interpretability during training. We find that it is possible to make Othello-GPT models more interpretable through training without interfering with convergence, suggesting that we may be able to promote interpretability without harming convergence. On the other hand, promoting interpretability did not *increase* the convergence rate or decrease the final loss, although our experiments do not rule out that this phenomenon will not happen on different datasets or settings.

References

Alain, G. and Bengio, Y. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Hatfield-Dodds, Z., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Elhage, N., Hume, T., Olsson, C., Nanda, N., Henighan, T., Johnston, S., ElShowk, S., Joseph, N., DasSarma, N., Mann, B., Hernandez, D., Askell, A., Ndousse, K., Jones, A., Drain, D., Chen, A., Bai, Y., Ganguli, D., Lovitt, L., Hatfield-Dodds, Z., Kernion, J., Conerly, T., Kravec, S., Fort, S., Kadavath, S., Jacobson, J., Tran-Johnson, E., Kaplan, J., Clark, J., Brown, T., McCandlish, S., Amodei, D., and Olah, C. Softmax linear units. *Transformer Circuits Thread*, 2022a. https://transformer-circuits.pub/2022/solu/index.html.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., and Olah, C. Toy models of superposition. *Transformer Circuits Thread*, 2022b. https://transformer-circuits.pub/2022/toy_model/index.html.

Hu, W., Xiao, L., Adlam, B., and Pennington, J. The surprising simplicity of the early-time learning dynamics of neural networks, 2020.

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

Nanda, N. Actually, othello-gpt has a linear emergent world model, Mar 2023. URL <https://neelnanda.io/mechanistic-interpretability/othello>.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.