CheckersGPT: Learning World Models through Language Modeling

Abhinav Joshi[*] **Vaibhav Sharma**[*] **Ashutosh Modi**
Indian Institute of Technology Kanpur (IIT Kanpur)
`{ajoshi, svaibhav, ashutoshm}@cse.iitk.ac.in`

Abstract

Although Large Language Models (LLMs) have been trained using just the next token prediction objective, these have shown impressive performance on various tasks. Consequently, it has attracted research interests in this regard. While one line of work in the past has suggested that LLMs learn surface-level statistics from the dataset, another line of work emphasizes that the learned representations are effective for simulating the underlying world model, considering the causal relationship for the next token prediction. This phenomenon is often referred to as the emergence of a world model in sequence prediction tasks. Recent work has demonstrated this phenomenon in a simulated setting of board games like Othello and Chess. In this paper, we analyze the game of Checkers to find out the emergence of a world model in a language model. By training a GPT-style autoregressive language model using only the next character prediction objective, we find that the model does show a hint of learning a world model representation of the board positions. We perform our analysis on two datasets: 1) synthetic dataset, which comes from the checkers game tree, and 2) human gameplay dataset. With multiple models trained with different layer sizes, we find that increasing the parameter size does help learn better world model representation decoded by linear probes.

1 Introduction

Though the Large Language Models (LLMs) have shown a remarkable ability to perform well on a wide range of tasks (Radford et al., 2019; Brown et al., 2020), the underlying process behind predicting the desired next token remains a black box. Since these language models are trained on a huge corpus of human-written text, it remains challenging to validate whether the network models the causal relationships or relies on spurious correlations occurring in the large corpus. Some initial findings suggest LLMs rely on surface-level statistics, stating LLMs are *'stochastic parrots'* (Bender et al., 2021) and *'causal parrots'* (Zečević et al., 2023). On the other hand, some of the recent work highlights LLMs learn feature representations that help create a proxy for an internal representation of the world. This property of the emergence of the 'world model' (Ha and Schmidhuber, 2018) in the learned representations has attracted significant research interest in recent years (Li et al., 2021; Toshniwal et al., 2022; Li et al., 2023; Nanda et al., 2023; Karvonen, 2024). The former line of work deals with model interpretations from an end-to-end perspective, whereas the latter heavily relies on understanding the complex properties via learned hidden representations. In this work, we focus on the latter part and analyze the learning process of GPT-style autoregressive models (Radford et al., 2018). Recent works in this line have shown these representations encoding complex properties present in the language (Bau et al., 2020; Goh et al., 2021; Geva et al., 2021; Burns et al., 2023; Elhage et al., 2022; Gurnee et al., 2023). More recently, researchers have explored the learning behavior using a simulated setting coming from a board game like Othello (Li et al., 2023; Nanda et al., 2023) and Chess (Toshniwal et al., 2022; Karvonen, 2024). We extend this setting to the game of Checkers (also see App. A) by training a GPT-style architecture (Radford et al., 2018) over the task of the next character prediction. We consider gameplay trajectories (sequence of moves in the game of checkers) coming from human gameplay as well as create a synthetic dataset using a random legal move in the game, i.e., considering the random sequences obtained from the game tree of checkers. A noteworthy feature of this training setup is that no information is provided to the network about the game of checkers, i.e., no legal moves feedback, no game design, and no information about the quality of the trajectory (more strategic or less strategic). Inter-

[*] Equal Contributions

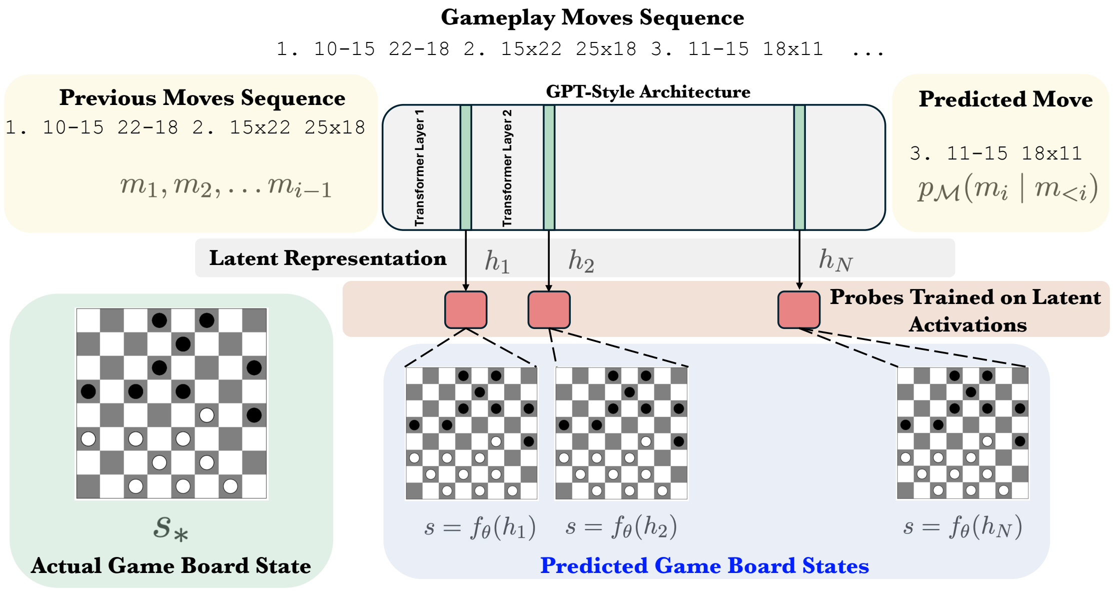


Figure 1: The Figure shows the GPT-style architecture trained over gameplay move sequences (represented as a PDN string) for Checkers. The evidence of the emergence of the 'world model' is obtained by probing the latent activations h_i from different layers for predicting the actual board state, where θ denotes the parameters for linear probes. Note that the figures show the next move prediction as an objective for representation purposes; in actual experiments, we considered the next character prediction for training the models. (see §3 for more details)

estingly, the training setup we follow does not even consider the next move prediction as the learning objective but rather the next character prediction (multiple characters form a move in the game), and the network only considers the sequence of characters as an instream to learn about the game. The closed setup of the game with definite board states helps facilitate simulating the underlying 'world model', i.e., the rules of the game decide the legal moves. We explore if the next character prediction tasks help learn useful representation forming the proxy for the board game, i.e., 'world model' for checkers. We train multiple networks with different layer sizes and observe the learning behavior across the training. Further, for analyzing the emergent behavior of the trained model, we consider using probes (Alain and Bengio, 2018; Belinkov, 2022) trained over latent representations learned by the model. Figure 1 provides a brief overview of the experimental setup. We observe that the deeper layers of the generative models tend to predict the game board state better, showcasing a piece of evidence for emergent behavior. In a nutshell, we make the following contributions:

- We create a simulated setting of decision-making tasks using the game of Checkers. Following previous works on OthelloGPT (Li et al., 2023; Nanda et al., 2023) and ChessGPT (Karvonen, 2024), this adds a new setting, facilitating the interpretability research.
- We curate a human gameplay dataset and a synthetic version of the dataset, containing 22,607 and 16 million gameplay trajectories, respectively.
- We analyze the learned representations in detail and provide empirical evidence to support the emergent behavior of GPT-style autoregressive models. We release the codebase and the dataset for the experiments `https://github.com/Exploration-Lab/CheckersGPT`

2 Related Work

The study of representations as a proxy for world models has been an active area of research interest in recent times. Li et al. (2021) study sequence generation models over synthetic tasks and report finding latent features encoding a proxy of world models. Some of the other works on similar lines highlight language models encoding ground concepts (Abdou et al., 2021; Patel and Pavlick, 2022; Burns et al., 2023), following previous works that study the linguistic features learned by sentence representations (Conneau et al., 2018; Tenney et al., 2019). Moreover, other recent works like McGrath et al. (2022); Lovering et al. (2022) analyze the representation learned by AlphaZero (Silver et al.,

Dataset Source	Game	# Games	# Tokens
Human	Othello	132,921	59
Synthetic	Othello	16,000,000	59
Human	Chess	16,000,000	1023
Human	Checkers	22,607	399
Synthetic	Checkers	16,000,000	399

Table 1: Statistics for synthetic and human gameplay datasets. The stats for Othello and Chess are taken from Li et al. (2023) and Karvonen (2024), respectively.

2017) encoding chess concepts. Some initial works in the simulated setting of board games study the game of Chess for GPT-style architectures (Toshniwal et al., 2022), which was extended for the Othello game in Li et al. (2023); Nanda et al. (2023); Hazineh et al. (2023). More recently, Karvonen (2024) provides a detailed analysis by training a GPT-syle architecture on the Chess gameplay sequences.

3 Methodology

To study the learning behavior of GPT-style architecture trained on a next token prediction task, we create an experimental setup by considering the sequence of gameplay moves in checkers (see Figure 1). We consider a character as a token and train the model to predict the next character in the gameplay sequence strings. Note that in the gameplay string, multiple sets of characters define a move in the gameplay. We intentionally train the models in such a fashion so that it doesn't provide any prior knowledge about the game of checkers (no game rules or board states). We further validate if such a training objective can learn a world model for the game of checkers while optimizing to predict the next token based on the previous tokens.

3.1 Dataset

To obtain the set of sequences coming from a Checkers game, we considered two sources: 1) Synthetic dataset, and 2) human gameplay dataset. The synthetic sequences are generated via uniformly sampling next moves from a Checkers game tree, whereas the Human gameplays are obtained from an online Checkers platform.[1] Figure 2 shows the distribution of token lengths in the obtained human gameplay dataset. To have a similar distribution for the synthetic version, we clip the sequence in the range of 100 to 400. Previous approaches (Li et al., 2023; Nanda et al., 2023; Karvonen, 2024) used a similar strategy to create synthetic and human gameplay versions of the datasets for board games like Othello and Chess. Table 1 compares the created Checkers game sequence resource with the existing works. The gameplay sequences in the created datasets are represented as a Portable Draught Notation (PDN) string. App. Figure 5 provides a detailed explanation of the gameplay sequence format.

[1]https://www.fierz.ch/download.php

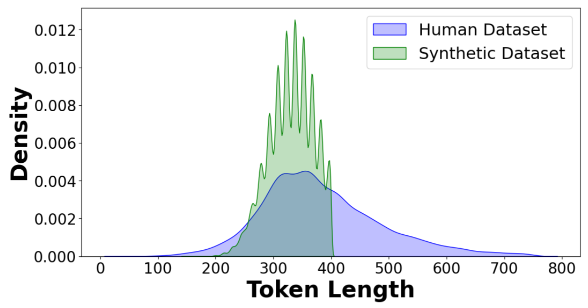


Figure 2: Distribution of the token length in the synthetic and human gameplay datasets. We stop the game tree at the token length of 400 and generate the synthetic dataset.

3.2 Model Training

We focus on studying the learning behavior of the model trained on the next token prediction task. Rather than providing the task as the next move prediction in a sequence, we reduce the inductive bias by training the architecture on the next character prediction instead. Moreover, the model has no idea about the underlying mechanism from which these strings are generated. We use a vocabulary size of 17 characters to encapsulate the gameplay PDN string, which is "`-./0123456789;x`" white space "_" and new line "\n." Note that the squares (positions in the Checkers board) are also encoded in the string, reducing the inductive bias of encoding a single board position as a token. We consider a GPT-style architecture (Radford et al., 2018) for all of our experiments and train them in an autoregressive fashion to predict the next character in the sequence. For a sequence of N tokens $\mathbf{t}_1, \mathbf{t}_2, \ldots \mathbf{t}_N$ defining the gameplay trajectory. The model is trained to predict the token $\mathbf{t}_i$ using all the previous tokens $\mathbf{t}_1, \mathbf{t}_2, \ldots \mathbf{t}_{i-1}$. We employ the autoregressive style training using a causal mask and minimize the cross-entropy loss between the actual sequence token and the predicted logits. App. Table 2 provides the architecture-specific details for the trained models. To notice the effect of parameter size, we use the same set of hyperparameters for training different variants of the same architecture, i.e., 1-layer, 2-layer, 4-layer, 8-layer. App. Figure 5 provides some qualitative examples of the input format provided to the model. For both datasets, we used a 99/1 train/test split to train the model in an autoregressive fashion.

3.3 Evaluation

We monitor the learning behavior of the models using various objectives.

1) Loss: We monitor the training/test losses across the training to validate if the model is able to minimize the learning objective.

2) Valid Moves Accuracy: Another metric that helps capture the learning behavior is the percentage of valid moves generated by the model. We consider the Checkers game engine to explore the game tree and validate the trajectory generated by the trained models. Note that the model predicts the game sequence at the character level, whereas the game tree validates the sequence of legal game moves. We sample N sequences from the trained model and check if the sequence contains strings processable by the game engine, i.e., all the generated sets of characters signify a gameplay sequence with all legal moves. We consider the generated string invalid even if one of the predicted characters is invalid. We report the percentage of valid strings from the sampled K strings.

Probing Internal Model Representations: To gain a deeper insight into the latent representations learned by the model, we make use of the probing literature (Alain and Bengio, 2018). Probing provides a way to quantify the quality of learned representations (Belinkov, 2022). We consider the underlying world model representation of the game board state and train linear probes over the layer activations to predict the current board state. The linear probe for square s at layer l of the model is formulated as follows:

$$P_{s,l} = \text{Softmax}(W_{s,l} \cdot A_l),$$

where $P_{i,l}$ is the probability distribution over the 3 classes for square s, as predicted by the probe at layer l, $W_{i,l}$ is the weight matrix for the linear probe associated with square i at layer l, A_l is the 512-dimensional activation vector from layer l of the model. We trained separate linear probes for each model layer as well as each board square. Each probe at layer l takes the input as the activation from the model's l^{th} layer.

We hypothesize that if a linear probe with little capacity can learn to predict the state of the board (which is not a linear function of the input), it demonstrates the capability of the model to transform the input into a linear representation of the board states in the latent activations. Essentially, the higher the accuracy of linear probes, the better the quality of learned representations, highlighting the learning of the underlying world model. Though the board contains 64 squares, only 32 are active board places where a piece can exist. We only consider the active 32 squares for reporting the probing accuracy. For each square, the probe is designed to classify it into one of the three possible states (Black Piece, White Piece, and Empty square). Recently, Nanda et al. (2023) reported the OthelloGPT representing (Mine, Yours, Empty) rather than (Black, White, Empty) in the form of linear representations. We follow a similar format for our experiments, where the probes predict the current game state relative to the current player at each turn, i.e., for odd turns, the model considers Black pieces as Mine and White pieces as Yours, and vice versa for even turns. Moreover, in our setup, since the model is trained on a next character prediction task rather than the next move prediction, multiple characters form a single move. For computing the probing accuracy, we consider the features corresponding to the last token before the next move. For white's move, we consider the representations corresponding to `'.'`, i.e., the white move starts after `<turn#>`, and for black's move, we consider features corresponding to `<space>` token before black's moves (see App. Figure 5 for a reference PDN string). We perform a 50/50 train/test split for probing experiments and report the test accuracy.

4 Results and Discussion

We perform all the experiments by considering different layer-sized models over the synthetic and human gameplay datasets. Overall, we found the training done on human trajectories to be much more stable than the ones done on randomly sampled trajectories from the game tree, with the models trained on the synthetic version of the dataset having low valid moves accuracy. App. Table 3 and Table 4 show the number and percentage of generated legal moves, respectively. We observe that

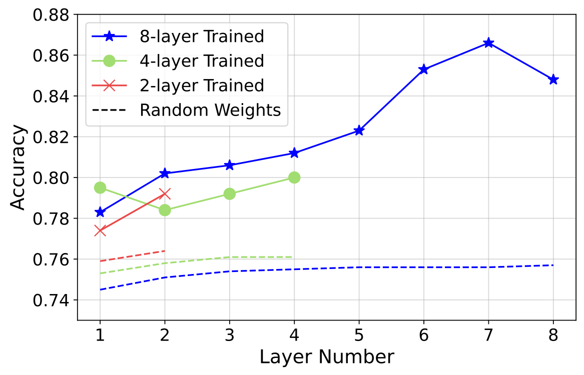


Figure 3: Probe classification accuracy for various models with different layer sizes. The dashed line represents the probe accuracy models with random weights.

the models with 4 and 8 layers learn better than the ones with a smaller number of layers and are able to generate sequences with 90% and 99% legal moves, respectively. However, we found that the model trained over the synthetic sequences generates a very low number of valid moves (in ranges of 100 even after 600K training steps), when compared to models trained on human gameplay sequences.

Probing Experiments: Figure 3 provides probing accuracy for various models with different layer sizes. We observe that for trained models, the probe accuracy for predicting board states using latent activations increases significantly compared to models with randomly initialized parameters. Moreover, we observe that the board states become more predictable for deeper layers in the larger network (8-layer), highlighting the quality of learned representations forming the underlying world model. App. Table 5 shows the probe accuracy for different layers of the models. Surprisingly, we observe that though the model trained on a synthetic dataset generates a very low number of valid moves, the latent representations learned by the model are good enough to predict the board state with a comparable probing accuracy. Note that, for validation of the model's learning, we considered the metric of valid moves percentage, where all the trajectories with a single wrong character are ignored, making it a rigid evaluation scheme. Previous works like Karvonen (2024), have considered top-k% to report the accuracy of the valid move; however, in Checkers specifically, multiple sets of moves are possible with different numbers of characters, i.e., when taking a capture x move, the number of characters in a move change (in contrast to chess where the number of characters is fixed). Given the setting, it becomes difficult to process the validation metric in such a fashion. App. Figure 6 shows the predictions obtained for the best model with 8 layers along with the corresponding ground truth board states.

5 Discussion

The notion of emergence argues that as the complexity of a system increases, new properties start to appear (Anderson, 1972). This idea of emergence has recently gained attention due to the increasing sizes of datasets as well as a number of parameters in LLMs, i.e., exhibiting 'emergent abilities' (Ganguli et al., 2022; Srivastava et al., 2023; Wei et al., 2022). Though there has been some evidence of these emergent abilities, Schaeffer et al. (2023) argues that the design of evaluation metrics plays a crucial role in observing the emergent behavior when increasing parameter size. In this work, we studied another variant of emergence, more specifically, if a simple objective of next character prediction can lead to learning the underlying 'world models', i.e., does the model learn the governing dynamics of the actual system (Checkers in our case) to satisfy the next token prediction objective. Though our findings point towards models learning the underlying 'world models,' it is to be noted that the evaluation measure we use may not be a good measure and only provides a weak signal. As recently highlighted by Vafa et al. (2024), a better evaluation metric is required for justifying the presence of implicit world model representations. Moreover, the settings we used had huge constraints; it would be interesting to consider a more open-ended setting encapsulated using a written text in the future.

6 Conclusion

In this work, we developed and studied a simulated setting, considering the board game of Checkers, and asked if a next token prediction objective in a GPT-style architecture helps learn the underlying world model for a task. With a detailed set of experiments over a varying range of models, we find that the deeper models (with about 25M parameters) show evidence of learning representations that encapsulate a proxy for the underlying world model. We believe this study will help facilitate future research on the emergent behavior of autoregressive models, by providing a firm playground for understanding representations learned during training over Checkers game sequences.

Limitations

We only consider the probing accuracy to get a hint of model learning 'world models.' For making the claim more concrete, previous works (Li et al., 2023; Nanda et al., 2023; Karvonen, 2024) have also proposed various intervention schemes applied over the trained probes, where they intervene over the learned representation to observe the changes in model's predictions, highlighting the formation of world models. In the future, it would be interesting to try similar strategies for the proposed setting of checkers.

Another major limitation of this work is the limited setting for the number of experiments; we only considered one architecture with different layer sizes to gain deeper insights into the training behavior of the models. In the future, it would be interesting to validate the same for a range of models with high parameter sizes.

Due to the limited number of game trajectories available online as well as limited computing, we performed all the experiments in a small dataset/parameter size regime. It would be good to study the training process with a larger number of parameters, specifically for the synthetic dataset where we could not get the model to generate valid game trajectories.

Ethical Considerations

In this work, we focused on a simulated setting of the game of Checkers. To the best of our knowledge, the proposed setting and the model does not have any negative consequences on the society at large. Moreover, the proposed approach and model are restricted to research only; we do not advocate the usage of the model in real-life online Checker gameplay.

References

Mostafa Abdou, Artur Kulmizev, Daniel Hershcovich, Stella Frank, Ellie Pavlick, and Anders Søgaard. 2021. Can language models encode perceptual structure without grounding? a case study in color. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 109–132, Online. Association for Computational Linguistics.

Guillaume Alain and Yoshua Bengio. 2018. Understanding intermediate layers using linear classifier probes. *Preprint*, arXiv:1610.01644.

P. W. Anderson. 1972. More is different. *Science*, 177(4047):393–396.

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. 2020. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? . In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2023. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. Toy models of superposition. *Transformer Circuits Thread*.

Deep Ganguli, Danny Hernandez, Liane Lovitt, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova Dassarma, Dawn Drain, Nelson Elhage, Sheer El Showk, Stanislav Fort, Zac Hatfield-Dodds, Tom Henighan, Scott Johnston, Andy Jones, Nicholas Joseph, Jackson Kernian, Shauna Kravec, Ben Mann, Neel Nanda, Kamal Ndousse, Catherine Olsson, Daniela Amodei, Tom Brown, Jared Kaplan, Sam McCandlish, Christopher Olah, Dario Amodei, and

Jack Clark. 2022. Predictability and surprise in large generative models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 1747–1764, New York, NY, USA. Association for Computing Machinery.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Gabriel Goh, Nick Cammarata †, Chelsea Voss †, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. 2021. Multimodal neurons in artificial neural networks. *Distill*. Https://distill.pub/2021/multimodal-neurons.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing. *Transactions on Machine Learning Research*.

David Ha and Jürgen Schmidhuber. 2018. World models.

Dean Hazineh, Zechen Zhang, and Jeffrey Chiu. 2023. Linear latent world models in simple transformers: A case study on othello-GPT. In *Socially Responsible Language Modelling Research*.

Adam Karvonen. 2024. Emergent world models and latent variable estimation in chess-playing language models. *Preprint*, arXiv:2403.15498.

Belinda Z. Li, Maxwell Nye, and Jacob Andreas. 2021. Implicit representations of meaning in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1813–1827, Online. Association for Computational Linguistics.

Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Charles Lovering, Jessica Forde, George Konidaris, Ellie Pavlick, and Michael Littman. 2022. Evaluation beyond task performance: Analyzing concepts in alphazero in hex. In *Advances in Neural Information Processing Systems*, volume 35, pages 25992–26006. Curran Associates, Inc.

Thomas McGrath, Andrei Kapishnikov, Nenad Tomašev, Adam Pearce, Martin Wattenberg, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. 2022. Acquisition of chess knowledge in alphazero. *Proceedings of the National Academy of Sciences*, 119(47).

Neel Nanda, Andrew Lee, and Martin Wattenberg. 2023. Emergent linear representations in world models of self-supervised sequence models. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 16–30, Singapore. Association for Computational Linguistics.

Roma Patel and Ellie Pavlick. 2022. Mapping language models to grounded conceptual spaces. In *International Conference on Learning Representations*.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2023. Are emergent abilities of large language models a mirage? In *Advances in Neural Information Processing Systems*, volume 36, pages 55565–55581. Curran Associates, Inc.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *Preprint*, arXiv:1712.01815.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Johan Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew M. Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, Cesar

Ferri, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Christopher Waites, Christian Voigt, Christopher D Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, C. Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danielle Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth Donoway, Ellie Pavlick, Emanuele Rodolà, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, Fatemeh Siar, Fernando Martínez-Plumed, Francesca Happé, Francois Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Xinyue Wang, Gonzalo Jaimovitch-Lopez, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Francis Anthony Shevlin, Hinrich Schuetze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, Jackson Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B Simon, James Koppel, James Zheng, James Zou, Jan Kocon, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Joan Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jörg Frohberg, Jos Rozen, Jose Hernandez-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua B. Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh Dhole, Kevin Gimpel, Kevin Omondi, Kory Wallace Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Lucas Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros-Colón, Luke Metz, Lütfi Kerem Senel, Maarten Bosma, Maarten Sap, Maartje Ter Hoeve, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ramirez-Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L Leavitt, Matthias Hagen, Mátyás Schubert, Medina Orduna Baitemirova, Melody Arnaud, Melvin McElrath, Michael Andrew Yee, Michael Cohen, Michael Gu, Michael Ivanitskiy, Michael Starritt, Michael Strube, Michał Swędrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimee Xu, Mirac Suzgun, Mitch Walker, Mo Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, Mukund Varma T, Nanyun Peng, Nathan Andrew Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha S. Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter W Chang, Peter Eckersley, Phu Mon Htut, Pinyu Hwang, Piotr Miłkowski, Piyush Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan Le Bras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Russ Salakhutdinov, Ryan Andrew Chi, Seungjae Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman, Samuel Stern Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima Shammie Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven Piantadosi, Stuart Shieber, Summer Misherghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsunori Hashimoto, Te-Lin Wu, Théo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Venkatesh Ramasesh, vinay uday prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yufang Hou, Yuntao Bai, Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Shubham Toshniwal, Sam Wiseman, Karen Livescu, and Kevin Gimpel. 2022. Chess as a testbed for language model state tracking. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 11385–11393. AAAI Press.

Keyon Vafa, Justin Y. Chen, Jon Kleinberg, Sendhil Mullainathan, and Ashesh Rambachan. 2024. Evaluating the world model implicit in a generative model. *Preprint*, arXiv:2406.03689.

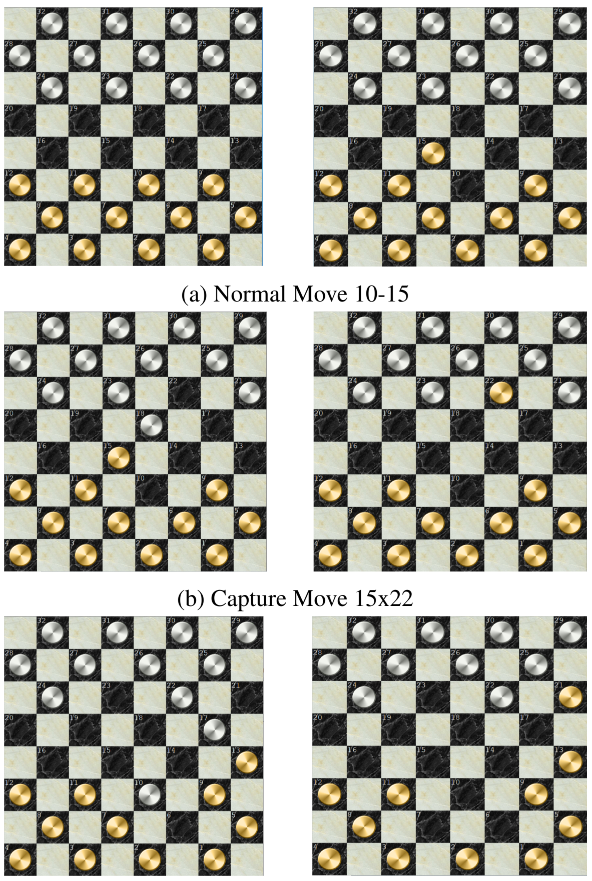
Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Matej Zečević, Moritz Willig, Devendra Singh Dhami, and Kristian Kersting. 2023. Causal parrots: Large language models may talk causality but are not causal. *Transactions on Machine Learning Research*.

Appendix

A The Game of Checkers

Checkers[2] is a board game played by two opponents on opposite sides of the game board. One player has dark pieces; the other has light pieces. A move consists of moving a piece forward to an adjacent unoccupied square. If the adjacent square contains an opponent's piece, and the square immediately beyond it is vacant, the piece may be captured (and removed from the game) by jumping over it. Only the dark squares of the checkerboard are used, and a piece can only move forward into an unoccupied square. When capturing an opponent's piece is possible, capturing is mandatory in most official rules. In almost all variants, a player with no valid move remaining loses. This occurs if the player has no pieces left or if all the player's pieces are obstructed from moving by opponent pieces. The gameplay sequence is captured in the PDN format[3], which is of the form "`1. 10-15 22-18 2. 15x22 ...`". Figure 5 describes the structure of the gameplay sequences in detail. The moves can be broadly classified into a normal move represented with "-" where a piece moves from one position to another, a capture move represented with "x" where an opponent piece is captured, and a multi-capture move "x x" where multiple pieces are captured. Figure 4 shows an example on the checker's board.


(a) Normal Move 10-15

(b) Capture Move 15x22

(c) Multi Capture Move 7x14x21

Figure 4: Different types of moves

B Hyperparameters

The full set of hyperparameters is given in the Table 2. We trained the model on 4 NVIDIA A40 with a total training time of close to 14 hours. We used a 50/50 split to train the linear probes and a 99/1 split to train the model. We used the AdamW optimizer (Loshchilov and Hutter, 2019), with a maximum learning rate of 3e-4 and a minimum learning rate of 3e-5. We used the block size of 399 for our autoregressive task. We chose the model dimension of the transformer to be 512 with 8 heads and 8/4/2/1 layers. We trained the model for 600,000 iterations.

Hyperparameter	**Value**
Optimizer	AdamW
Learning Rate Schedule	Cosine
Max Learning Rate	3e-4
Min Learning Rate	3e-5
Weight Decay	0.1
Betas	0.9, 0.95
Batch Size	100
Block Size	399
Training Iterations	600,000
Dropout	0.0
d_model	512
n_heads	8
n_layers	8,4,2,1
Parameters (8 layers)	25M
Parameters (4 layers)	12.6M
Parameters (2 layers)	6.3M
Parameters (1 layers)	3.16M

Table 2: Hyperparameter values for the model training.

[2]https://en.wikipedia.org/wiki/Checkers
[3]https://en.wikipedia.org/wiki/Portable_Draughts_Notation

```
Game Sequence Format:
<turn-#>player-1-moves<space>player-2-moves<turn-#>player-1-moves<space>player-2-moves. . .

1. 10-15 22-18 2. 15x22 25x18 3. 11-15 18x11 ...
1. 11-15 24-20 2. 8-11 28-24 3. 9-13 22-18 ...
1. 9-14 22-18 2. 11-15 18x11 3. 8x15 25-22 ...
1. 12-16 24-20 2. 11-15 20x11 3. 7x16 22-18 ...
1. 10-15 21-17 2. 6-10 17-14 3. 9x18 23x14 ...
```

```
Special Characters:
'-': denotes the move from one board position to another board position
'x': denotes the moves when a piece is captured.
```

Figure 5: Input string formats for the GPT-style architecture training in the autoregressive format. The game sequence is a string of characters, and the network predicts the next character based on the previous characters. `red text` is the template style denoting the separation between the turns and the moves by the two players. The `blue text` denotes moves by player-1 followed by `orange text` showing moves by player-2. Note that the network only predicts the next character for all the experimentation, and no other information about the game rules is provided to the network.

Iteration	**500**	**1K**	**5K**	**10K**	**50K**	**100K**	**300K**	**600K**
8 Layer	189	1034	3482	3913	4595	4576	4733	4808
4 layer	152	667	3412	3735	4527	4553	4551	4802
2 layer	125	256	2	1	1	7	367	4693
1 layer	3	0	0	0	0	0	0	0

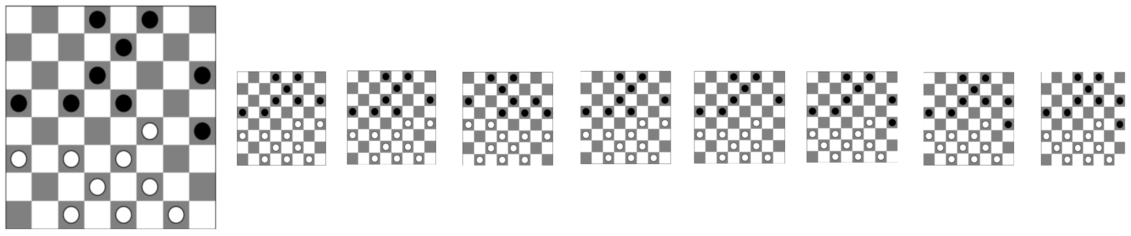
Table 3: The table shows the total number of valid moves generated by the models across training iterations. We generate 100 gameplay sequences from the obtained checkpoints and sum up all the legal moves obtained for the sequences. For a single trajectory, we only generate the sequence until the model generates an invalid character and increase the count accordingly; for example, if the first 20 moves are legal in the current trajectory, we increase the number of legal moves count by 20.

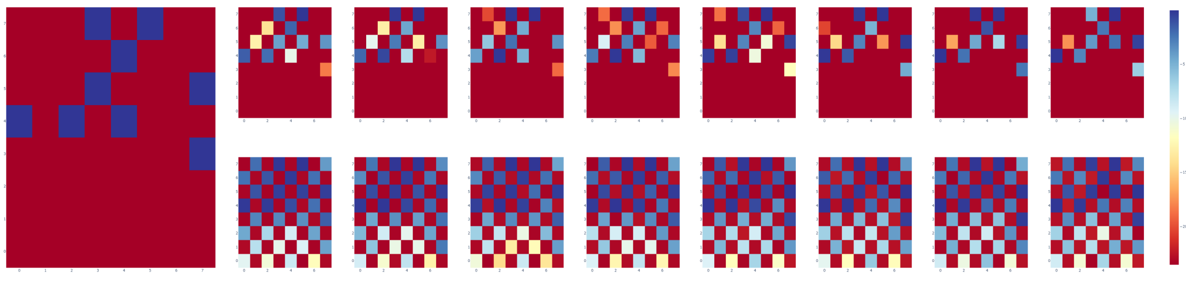
Iteration	**500**	**1K**	**5K**	**10K**	**50K**	**100K**	**300K**	**600K**
8 Layer	0%	0%	33%	54%	86%	91%	99%	100%
4 Layer	0%	0%	34%	54%	87%	84%	90%	100%
2 Layer	0%	0%	0%	0%	0%	0%	6%	93%
1 Layer	0%	0%	0%	0%	0%	0%	0%	0%

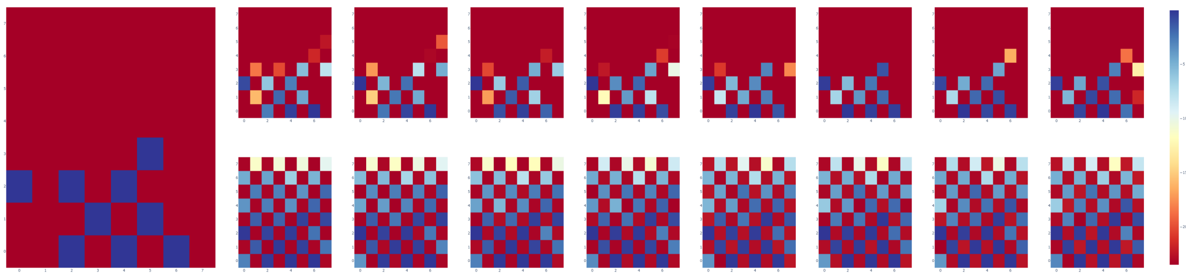
Table 4: The table shows the percentage of valid game sequences generated by different models over the range of learning steps (Iterations). We observe that for models with a higher number of layers, the models start to generate valid moves 99% of the time. Whereas the smaller model with Layer-2 learns slowly, reaching 93% after 600K training steps, the smallest model with only 1 transformer layer fails to generate trajectories with legal moves.

	Layer 1	**Layer 2**	**Layer 3**	**Layer 4**	**Layer 5**	**Layer 6**	**Layer 7**	**Layer 8**
8 Layer	0.783	0.802	0.806	0.812	0.823	0.853	0.866	0.848
8 Layer(Synthetic data)	0.738	0.741	0.749	0.802	0.870	0.872	0.865	0.835
8 Layer Random	0.745	0.751	0.754	0.755	0.756	0.756	0.756	0.757
4 layer	0.795	0.784	0.792	0.80	-	-	-	-
4 layer Random	0.753	0.758	0.761	0.761	-	-	-	-
2 layer	0.774	0.792	-	-	-	-	-	-
2 layer Random	0.759	0.764	-	-	-	-	-	-
1 layer	0.726	-	-	-	-	-	-	-
1 layer Random	0.761	-	-	-	-	-	-	-

Table 5: Probing classifier accuracy obtained for each layer's activation. The bottom row in each sub-row denotes the random accuracy (i.e., for a model with no training). We observe that as the number of layers increases, the improvements in the probing accuracies improve by a significant margin over the random.


(a) Board prediction of each layer


(b) Black piece prediction of each layer


(c) White piece prediction of each layer

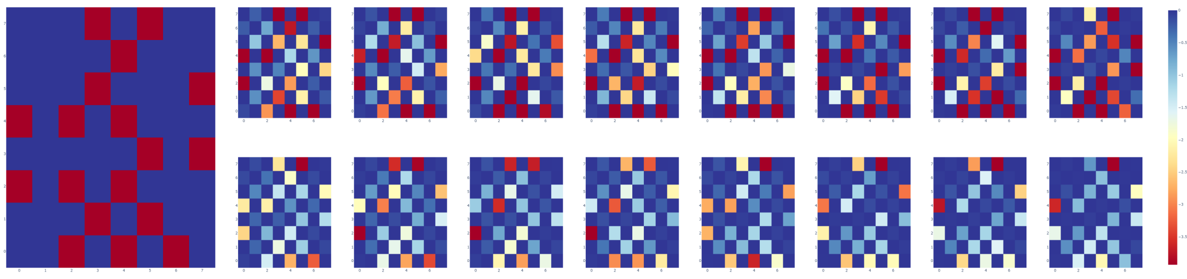

(d) Blank piece prediction of each layer

Figure 6: The figure shows the qualitative example for an instance of probes predicting the current board state. For each of the subfigures, the leftmost figure highlights the ground truth state with the predictions in the top row (clipped version for highlighting the prominent predictions) and the raw predictions in the bottom row.