RWKV for efficient learning of chess representations in the sequence-to-sequence paradigm

Richard Zhang
Mathematics, University of Waterloo, Waterloo, Canada
ry7zhang@uwaterloo.ca

Abstract

Recent advances in language models have sparked significant interest in understanding how they develop capabilities, particularly in relation to internal world representations formed during training. Chess, with its well-defined rules and structured game environment, serves as an excellent test case for investigating this phenomenon. In this paper, we represent chess games as sequences of moves and train both Transformer and RWKV models on this task. Our findings align with existing literature, demonstrating that the Transformer architecture is capable of predicting chess moves and developing internal representations of the game. Moreover, we also show that RWKV models surpass Transformers in this domain, achieving superior performance in both move prediction and world modeling metrics, confirming the capabilities of the RWKV architecture for world-modelling as well as their potential to underlie chess evaluation nets.

1 Introduction

The transformer architecture [1] and attention mechanism [2] have achieved undeniable success on tasks in the sequence-to-sequence paradigm. Autoregressive language models based on the architecture (GPT-3 [3], Gemini [4], Qwen [5]) have achieved state-of-the-art results across various benchmarks such as coding, story writing, translation etc. There is growing interest in these models' ability to reason ([6], [7]) as well as their capacity to function within environments (LLM agents, function calling landmark papers). Thus, it appears their capabilities far exceed the limited domain of sequence modeling they were originally trained on.

At the same time, not much is known about how they achieve these results. Widely, large neural nets are regarded as impenetrable black box, leading to doubts about whether they possess the capability for generalization and representation learning, or whether their behavior is merely due to advanced forms of compression and the reflection of surface statistics. Their susceptibility and tendency to hallucination is seen as lending credence to this theory. A notable example is the tendency of LLMs to generate plausible-looking chess moves, only to fall apart by making illegal moves shortly thereafter. This is often cited as evidence that sequence modeling does not cause the model to learn the hidden representations which engender their training data.

Nevertheless, it seems implausible that language models completely fail to capture the true hidden dependencies in the sequences they are trained on given their surpassing success across disparate domains. The transformer architecture, in particular, is designed to capture both short- and long-range dependencies via learned key, query, and value matrices, in a way akin to how humans take action in the world—by attending to certain stimuli and responding appropriately. Recently, techniques like attention activation maps [8], transformer circuits [9], and layer probes [10] have provided insight into certain parts of these models' inner workings, uncovering interpretable patterns that correlate with recognizable features beyond the surface statistics of training data, providing evidence for a deeper understanding of the phenomena generating the data.

For example, experiments have revealed internal representations for simple concepts, such as tracking boolean states during synthetic tasks [11] or concepts like color and direction [12]. However, due to the complexity and diversity of language, drawing conclusive results on higher-order concepts has proved challenging[13].

1.1 RWKV

Taking a step back, transformers can be viewed as a specific instance of sequence-to-sequence models. A question that can be raised about such models, in general, concerns the advantages and limitations of their approach compared to RNNs. Indeed, transformers emerged as an alternative to RNNs, excelling in managing both local and long-range dependencies, while supporting parallelized training—qualities that RNNs lacked. In part due to the vanishing gradient problem, RNNs have struggled with long-range dependencies. Modifications such as LSTMs and GLUs improved their performance, but the non-parallelizability in the time dimension still posed scalability challenges.

However, RNNs offer some advantages. One of the challenges facing transformers is their limited context length. To generate a new token, the entire history must be passed into the model again, making inference slow and memory-expensive (Time and memory costs both on order of $O(n^2)$). In contrast, RNNs require less memory in the forward pass by using the update operation $h_t = f(h_{t-1}, x_t; \theta)$, which is furthermore closer to how we conceptually understand sequence generation.

Furthermore, recent advances have made steps towards overcoming these difficulties. One approach has been to replace self-attention layers with linear complexity layers. Representative works in this domain include RWKV [14] and Mamba [15]. Both are conceptually recurrent models which address the drawbacks of recurrent architectures while maintaining its advantages. Essentially, these models allow for parallelized forward passes, like transformers, but can also unroll iteratively in the time dimension during inference, like RNN's.

Both variants have achieved comparable or surpassing performance to Transformers on tasks in diverse domains as image, audio, and text. RWKV-based models in particular achieve comparable performance to Transformer analogues with lower computational costs while maintaining stable scalability[16]. In this paper, we will explore RWKV (Section 3.3) as an alternative to the Transformer architecture, comparing its capabilities in modelling with the Transformer architecture on a specialized sequence task.

1.2 Chess

Chess, possibly due to its simple state representation and rich complexity, has long been a popular testing ground for machine learning. In 1996, IBM's Deep Blue defeated the reigning world champion, Garry Kasparov. Since then, advancements in computation and AI algorithms have continued to outpace human abilities. In October 2015, the original AlphaGo became the first computer Go program to defeat a human professional without handicap, marking a paradigm shift in game-playing algorithms. The chess community was later impacted by AlphaZero, which achieved superhuman capabilities through self-play reinforcement learning, winning 155, and only losing 6 games against Stockfish in a 1,000-game match.

AlphaZero employed a deep neural network with two main components: a policy network and a value network, both of which accept a game state as input. The policy network output a probability distribution over all possible moves, guiding the Monte Carlo Tree Search (MCTS) in selecting actions while the value network was responsible for estimating desirability of the state.

This event marked the end of hand-crafted evaluation networks, with Stockfish being a prominent example. In subsequent years, techniques like LC0 and Stockfish NNUE built upon these advancements. Interestingly, AlphaZero's play often mirrored human-like opening strategies, structures, and heuristics—contrasting with engines that sometimes make odd anti-positional moves justified by deep calculations.

Notably, this has spawned interest in interpreting these models, possibly with an end to potentially advancing human chess understanding of the game. In fact, there has been some success in this respect - in contrast to large language models, various techniques have discovered patterns which correspond to higher-level human concepts within them. In [17], linear probes detected heuristics such as king safety and outposts in human play. In [8], attention maps from the Lc0 transformer network revealed evidence of not only board state knowledge but also limited search capabilities.

We note that despite the recent presence of the transformer architecture, such models are not in general sequential, and they generally predict off of the game state, and not move sequences.

1.3 Overview

Indeed, the board state is by far the most natural and efficient way of training a chess model for the purposes of playing games. However, we argue that the rule-based and stateful nature of chess makes the sequential representation an intriguing one for investigating the emergence of world models in sequence models.

This paper explores the intersection of sequence modeling and chess models, with the hope that insights gained from this comparatively simplified setting could be applicable to natural-language settings. On the other hand, the transformer architecture, which has been successfully applied in sequential tasks, has also seen success in general chess models, as demonstrated by [18]. This parallel suggests that the capacity of an architecture in the domain of sequential tasks could also present evidence of its capability for the purpose general state-dependent models (i.e. chess-playing agents such as NNUE and Lc0).

Inspired by the sequential and stateful nature of chess, we aim to build and examine chess models within the sequence-to-sequence paradigm. Our targets are the Transformer and RWKV architectures. We begin by training a transformer model on tokenized PGN games from diverse sources, followed by an RWKV model. Due to the limited complexity of chess in comparison to standard language tasks, we decided that smaller networks should be viable for our purposes. To this point, the NanoGPT project was quite perfect for our purposes and we employed it with only minor adjustments and allowances for the architecture and task at hand.[1]

Previous works [20] trained GPT models on pure text sequences without any a priori knowledge of rules or board structure. In our paper, we take a different approach by using a vocabulary composed of all legal chess moves, distinguished by piece (7756 moves plus one padding token, which is ignored for loss computation). We contend that the resulting setup does not differ significantly and is rather more suitable for our purposes (Section 2.1).

Our focus is on observing what the model learns by merely processing tokenized game transcripts. To this end, we first evaluate the model's ability to make legal moves. Our RWKV model succeeds in generating legal moves in about 85% of cases, depending on the dataset, while our Transformer model achieves a lower success rate of around 55%. The set of legal moves is entirely determined by the current board state, and we consider the model's ability to generate these moves as strong evidence that it has learned to construct a form of representation of the underlying world model. Given that only about 0.4% of possible moves are legal on any given turn (35 out of 7758), this result indicates that both models have made significant progress in constructing a world model, with errors possibly arising from data-related quirks in the training process, such as by common move sequences, which we conjecture will continue to diminish with further training. The complexity of chess, the length of games, and evaluation on random synthetic data lead us to rule out simple "memorization" as an explanation. Although our results fall short of the baseline accuracy of 99.8% reported in [20], we attribute this to the disparity in training time and possible methodological differences.

We also evaluate the models' hidden world representation through other more complex features, such as defended squares and available pieces, using various forms of simulation (self-play, mutual play, and play against human opponents). We find that the models reliably identify undefended pieces, and when a piece type is captured, the model becomes less likely to make moves involving that piece. This suggests that the model is tracking the status of pieces and adjusting its strategy accordingly.

[1]although [18] and [19] have shown that the chess-playing ability of transformer-based models are largely governed by their size, our focus more on the emergence of structure within these models and not on their ability.

Additionally, following the methods outlined in [10], [20], we conduct linear probe analyses of various layers of each model to examine the internal representations of the board state. We find that the RWKV model outperforms the Transformer model, even with fewer training iterations, in capturing meaningful internal representations of the game.

In summary, our contributions are as follows: (1) We reproduce the phenomenon of emergent world models in a different domain using the methods from [10], [20]. (2) We show that RWKV models also learn the world model in the course of sequence modelling. (3) We demonstrate the superior performance of RWKV models in both sequence prediction and world modelling, within the same experimental setup.

2 Data

Chess is a game played across many time controls and skill levels. Depending on the situation and players, the distribution of moves within a game can vary greatly. In our paper, we are interested to observe what a sequential model can learn about the game itself, and potentially draw parallels with our own understanding of the game, not necessarily in producing a strong model. For example, in Section 6.2 we conduct linear probes to discover how the model represents the "skill level" of the game, if at all. To this purpose, we curate a diverse collection of games from various which we believe provides reasonable and balanced coverage across the various forms of the game.

In order to reduce the chance of the models inferring incorrect game rules that may be artifacts of human play, as well improve model understanding of rare situations, we supplemented our dataset with a synthetic dataset consisting of random legal moves up to the sequence length. This has an effect similar to label-smoothing, but we hypothesized it may have the advantage of not unduly influencing the disposition of the model toward illegal moves.

Each game is molded into a supervised data point for training/inference by tokenization followed by padding. We chose an original tokenization scheme, which allots every possible half-move a token which encodes the starting and destination squares together with the piece type.

2.1 Tokenization

Auto-regressive sequential models predict the next token given a past history of tokens. Previous works trained GPT models on pure text sequences without any a priori knowledge of rules or board structure. In our paper, we take a different approach by using a vocabulary composed of all legal chess moves (7756 moves plus one padding token, which does not factor in loss computation).

We have three primary reasons for this choice. First, the legal start and end squares for a given piece are clearly direct dependent on each other, and do not constitute evidence of the model learning underlying game structures. Thus the ability of models to generate legal moves from text is not in question. Second, as humans, we understand the chessboard in terms of moves rather than individual characters or squares. Moves are the fundamental units of our chess vocabulary, and as a result, a model trained on moves may be more readily interpretable[2]. Third, by reducing the vocabulary size, we decrease the model's complexity, making it more capable and enabling faster experimentation and a tighter feedback loop.

In our scheme, every legal string of the form `{piece type}{start square}{end square}{possible promotion}` is considered as a token.[3]. After tokenization, the result has as many tokens as game half moves. As both the RWKV and the Transformer architectures train on all initial substrings in parallel up to their block size (3.1), the tokenized game strings are either cut short or padded with a special padding token which is ignored when computing cross entropy.

2.2 Datasets

Each dataset in following sections but for the last is ultimately stored as a collection of pgn files within a folder. An `IterableDataset` structure on these files allows multiple workers to iteratively

[2] 6.4

[3] Alternatively, this is just UCI notation with the piece prepended. This possibility was noted by [21] in passing.

read from distinct files. Separate folders or sub-setting the files is used to delineate training and validation datasets.

2.2.1 Lichess Elite

The Lichess Elite database [22] is a collection of chess games played by strong players (with ratings 2400+ against 2200+) from 2013 to 2024, excluding bullet games. Since the vast majority of the remainder played are in the blitz time control (3 - 10 minutes per side), this dataset provides exposure to the features of blitz chess, whose moves are vaguely characterized as being more intuitive and free-flowing, without sacrificing much on quality.

2.2.2 Binned

The binned dataset [23] is a collection of zipped pgn files containing Lichess games binned by distribution ranging from 1000 to 4000 elo. We provide a script to convert the files into a format readable by the `chess.pgn` library and check for errors. Both sequence models are trained on for a limited number of iterations on the low elo ranges, providing exposure to less skilled game sequences such as arising from beginner games. Additionally, this dataset is used for constructing skill probes (Section 6.2).

2.2.3 Tournament & TCEC

The two datasets make up the games of the highest "quality" (i.e. lower centipawn loss, advanced positional concepts). Tournament games [24], [25] are played at a longer time control, by generally more serious players. TCEC games are played by engines, and can be considered to be at a level surpassing any human. We did not attempt to investigate whether the model was able to incorporate these concepts, but merely included them for completeness.

2.2.4 Synthetic

The synthetic dataset consists of randomly generated legal moves up to a certain sequence length. The training and validation set both sample from their own set of 500 pgn files consisting of 100 games each. The validation dataset was also used to test the models' ability to play legal moves.

2.2.5 FEN evaluations

This final dataset is used to investigate the performance of RWKV on a different chess-related task (Section 7.2).

A FEN [26] string is a standard string-based description of a chess position. It consists of a board state, the current side to move, the castling availability for both players, a potential en passant target, a half-move clock and a full-move counter, all represented in a single ASCII string.

The dataset [27] is made of pairs consisting of an FEN string together with a stockfish evaluation of the position, with positive values representing white advantage, and negative values black advantage.

Guided by the methodology of [18], we expand the run-length encoding of FEN and tokenize remaining characters to produce integer arrays of length 80, to be fed into an embedding layer. The evaluations are categorized into separated into 20 bins, each containing a 50 point evaluation range, with evaluations of value > 450 or < -450 in absolute value sorted into bins #10 and #20 resp.

3 Model Training

3.1 Network Input and Outputs

Both GPT and RWKV variants were trained as auto-regressive decoder-only models, accepting a sequence of tokens (Section 2.1), and outputting the corresponding next token prediction for all initial subsequences of the input. In our experiment, we adopted a sequence length of 140, representing 70 moves, sufficient to represent about 95.8% of games in our Lichess Elite database their entirety. The tokens are projected using learnable embeddings into a hidden dimension of 768. Learnable

Table 1: Model sizes.

Name	Total	Heads	Lay-ers	Embed size
Transformer	91.04M	12	12	768
RWKV	91.13M	12	12	768
Transformer (FEN) (Section 7.2)	85.1M	12	12	768
RWKV (FEN)	85.2M	12	12	768

positional embeddings are added. The embeddings are then passed into a series of 12 copies of an architecture-dependent block, responsible for the majority of computation. Finally, the output is fed into a final linear layer, to output the logits corresponding to each prediction. This final layer, which we refer to as the head, is the same matrix used to perform the embeddings, utilizing a method known as weight-tying [28]. One justification for this operation is that one-hot encoded vector inputs can be regarded as logits with all the density distributed to a single index.

3.2 Sequential Transformer

We describe the main block used in our transformer variant.

Given a sequence of tokens $(x_1, ..., x_n)$ where $x_i \in \mathbb{R}^d$, self-attention returns a sequence $(z_1, ..., z_n)$ where z_i in $\mathbb{R}^d$. Using projection matrices W^Q, W^V, W^K in $\mathbb{R}^{d\times d}$, a logit e_{ij} for each pair of tokens (i, j) is computed using scaled dot-product attention:

$$e_{ij} = \frac{(x_i W^Q)(x_j W^K)^T}{\sqrt{d}} \tag{1}$$

The attention weights are obtained via softmax:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})} \tag{2}$$

Finally, the output is computed as a weighted sum of value projections:

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V) \tag{3}$$

Our model utilizes multi-headed attention, wherein each “head” computes n=12 heads in parallel, each with their own unique key, value and query matrices.

This process is repeated for each head, in the self-attention layer, and the outputs are concatenated before being passed through a final linear projection.

3.3 Sequential RWKV

The RWKV block features and channel mixing and time mixing component, and can be understood as iteratively updating the input embedding x together with a state in both cases.

$$\begin{aligned}
x' &= \text{layer\_norm}(x, W_1, b_1) \\
dx, x_{\text{last},2} &= \text{tmix}\big(x', x_{\text{last},2}, c_{\text{last\_num}}, c_{\text{last\_den}}, c_{\text{decay}}, c_{\text{bonus}}, \text{mix}_k, \text{mix}_v, \text{mix}_r, W_k, W_v, W_r, W_{\text{out}}\big) \\
x &= x + dx \\
x' &= \text{layer\_norm}(x, W_2, b_2) \\
dx, x_{\text{last},1} &= \text{cmix}\big(x, x_{\text{last},1}, \text{mix}_k, \text{mix}_r, W_k, W_r, W_v\big) \\
x &= x + dx \\
&\text{(repeated n\_block times)}
\end{aligned} \tag{4}$$

The channel mixing component computes the residual updating as follows:

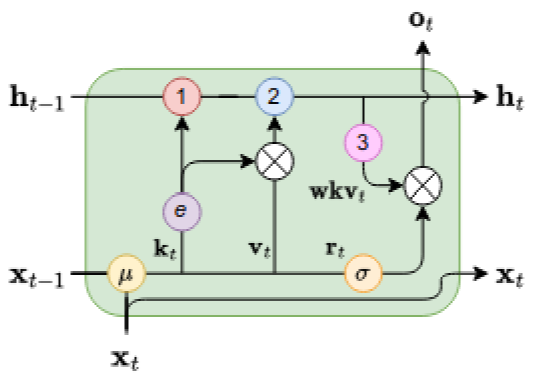
$$\begin{aligned}
k &= W_k \cdot (x \cdot \text{mix}_k + x_{\text{last}} \cdot (1 - \text{mix}_k)), \\
r &= W_r \cdot (x \cdot \text{mix}_r + x_{\text{last}} \cdot (1 - \text{mix}_r)), \\
v_k &= W_v \cdot (\max(k, 0))^2, \\
\text{output} &= \sigma(r) \cdot v_k, \quad \text{where} \quad \sigma(r) = \frac{1}{1 + e^{-r}}
\end{aligned} \tag{5}$$

It first interpolates between the current input and the last input using learned parameters. The interpolated input is run into a 2 layer feed forward network with squared relu activation, and finally multiplied by a sigmoid function. In terms of recurrent neural networks, we can understand this as the channel update being gated by r.

The time mixing component is computed as follows:

$$\begin{aligned}
k &= W_k \cdot (x \cdot \text{mix}_k + \text{last}_x \cdot (1 - \text{mix}_k)), \\
v &= W_v \cdot (x \cdot \text{mix}_v + \text{last}_x \cdot (1 - \text{mix}_v)), \\
r &= W_r \cdot (x \cdot \text{mix}_r + \text{last}_x \cdot (1 - \text{mix}_r)), \\
wkv &= \frac{\text{last}_{\text{num}} + \exp(\text{bonus} + k) \cdot v}{\text{last}_{\text{den}} + \exp(\text{bonus} + k)}, \\
num &= \exp(-\exp(\text{decay})) \cdot \text{last}_{\text{num}} + \exp(k) \cdot v, \\
den &= \exp(-\exp(\text{decay})) \cdot \text{last}_{\text{den}} + \exp(k). \\
rwkv &= \sigma(r) \cdot \text{wkv} \\
\text{output} &= W_{\text{out}} \cdot \text{rwkv} \ .
\end{aligned} \tag{6}$$

Similarly, the input is first interpolated between the current and its predecessor. Analogously to Transformers, key, value and *receptance* vectors via projection. We note that wkv is basically the value v weighted by k, together with contributions from the v's from previous time steps, weighted less the further back they are. Again, the output is gated by the receptance. As an RNN cell, it can be visualized as follows:



However, an advantage is that while the update equation to wkv is recursive, it can be also represented as

$$\mathrm{wkv}_i = \frac{\sum_{j=1}^{i-1} e^{-(i-1-j)w+k_j} v_j + e^{u+k_i} v_i}{\sum_{j=1}^{i-1} e^{-(i-1-j)w+k_j} + e^{u+k_i}} \tag{7}$$

and thus each time step can be computed in parallel.

4 Training

Our study involved training multiple models tailored to three distinct objectives: (1) predicting game sequences, (2) conducting model probes, and (3) predicting evaluations based on game states. Below, we detail the training procedures and configurations for the sequence-to-sequence models, which are central to this paper. The methodologies for objectives (2) and (3) are described in their respective sections.

4.1 Training Protocol

Both models were trained for a total of 240 epochs, with approximately 60 epochs allocated to each of the Classical, Lichess Elite, and lower-skill binned datasets. Dataset transitions were performed manually to better control the evolution of the models during training. While this manual approach resulted in slight variations in the number of iterations spent on each dataset for the two models, we consider these differences negligible due to the diminishing rate of convergence in later epochs. Subsequently, both models underwent an additional 40 epochs of training on synthetic datasets, consisting of 500 files per epoch, with each file containing 100 games. This step was motivated by observed model difficulties in handling anomalies, such as rare openings and illogical moves.

Note that we use the epoch unit merely as a book-keeping device representing around 800 batches. Due to the size of our datasets, all except the synthetic dataset were iterated over at most once. Consequently, dropout was not utilized during training.

We employed the AdamW optimizer, with an initial learning rate of 5e-3, decaying to 6e-6 after 200,000 iterations. The learning rate decay schedule follows a cosine decay rule from the repository [29], [30] our model code was based on, defined as:

$$\mathrm{lr} = \mathrm{min\_lr} + \eta(t) \cdot (\mathrm{lr} - \mathrm{min\_lr}) \tag{8}$$

where

$$\eta(t) \tag{9}$$

is computed as

$$\begin{array}{cc} \frac{t}{\mathrm{warmup\_iters}} & t < \mathrm{warmup\_iters}, \\ 0 & t > \mathrm{lr\_decay\_iters}, \\ 0.5\Big(1 + \cos\Big(\pi \frac{t - \mathrm{warmup\_iters}}{\mathrm{lr\_decay\_iters} - \mathrm{warmup\_iters}}\Big)\Big) & \mathrm{otherwise} \end{array} \tag{10}$$

The batch size was set to 120 for the GPT model and 100 for the RWKV model, reflecting the slightly larger size of the latter. In hindsight, we acknowledge potential flaws in our choice of hyperparameters. A more gradual learning rate decay, a lower initial learning rate, and a consistent batch size might have yielded better results.

4.2 Training Comparison

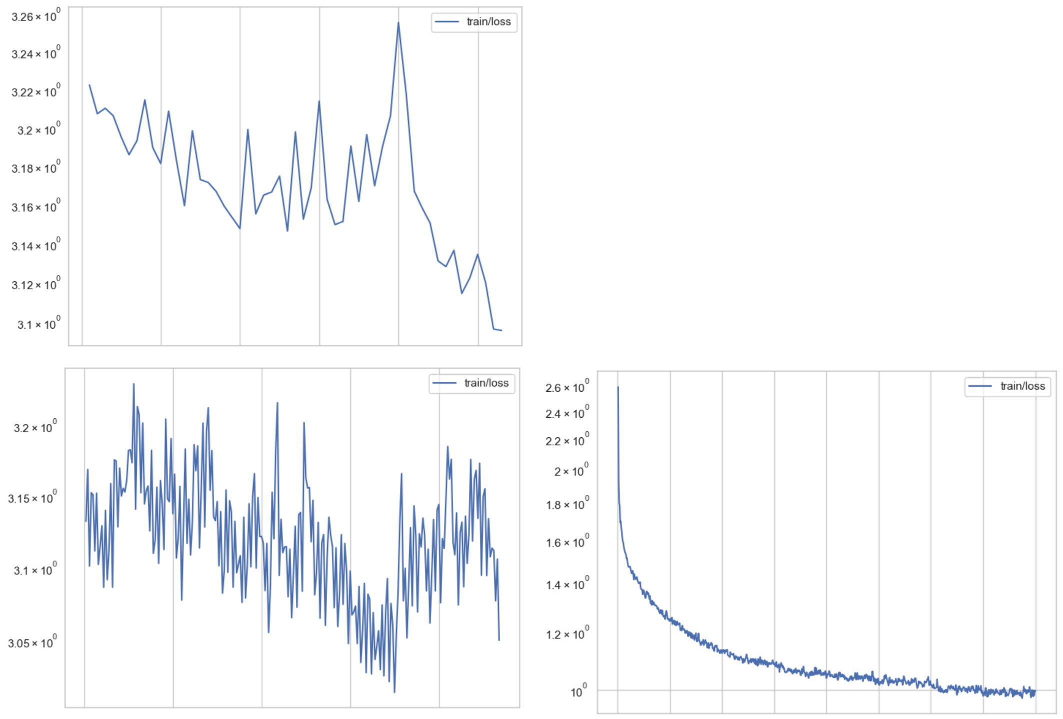


Figure 1: Transformer training progress (0-80, 80-240) (left) vs RWKV full training progress (~250 epochs) (right)

We noted that despite training a similar number of iterations on the same datasets, and almost identical hyper-parameters, the RWKV model was able to steadily improve while the Transformer model struggled to find footing. As noted in Section 4.1, our dataset sizes were sufficient for us to train on without repetition. Thus, training data is unlikely to be repeated, and validation loss is not needed. Ultimately, the cross-entropy loss of the RWKV model reached an order of 8 - 90 $\times$ 10^{-2} depending on the dataset, while the the GPT model remained in the range of 2-6, with higher losses for the synthetic dataset.

5 Results

We began our analysis by testing the capability of the model to play full games. We developed a simplified chess GUI through which models or humans could relay moves using the `python-chess` library. Model moves were selected by interpreting 7798-dimensional output of the model head on the game sequence as a probability distribution via the sigmoid function and sampling from it. Since models still had a tendency to predict illegal moves occasionally, we implemented a rule whereby the model was allowed 5 generation attempts, and would resign when all 5 were exhausted.

We then conducted an initial qualitative analysis by gauging its response against humans, followed by recording various statistics of games produced by automated play.

5.1 Human play

In the opening, the RWKV was a strong player, adept at a variety of openings, almost always securing an advantage in sample games. Illogical opening moves were punished: It would capture a piece left undefended, as well as punish positional mistakes. Often, its position would even be winning by the middlegame.

However, it frequently failed to follow through when said advantageous position lead to winning tactics. For instance, it might initiate a sequence that simultaneously put the king in check and left an opponent's piece undefended, but then neglect to capture the exposed piece once the opponent was forced to defend their king. Such a deficiency may be viewed as part of a larger tendency to making random blunders. Another deficiency was that, it occasionally struggled when multiple captures occurred, sometimes forgetting to capture back in these scenarios. As games continued, such weaknesses became more prominent, and the late middlegame and endgame phases proved the greatest challenge to it. While it could perform simple checkmates, such as with two queens, it lacked the ability to navigate complex endgame scenarios effectively.

Early in training, the model struggled to find legal moves when placed in a position with limited options, such as when in check. Due to the rules set in the program as described above, this led to early resignations. As training progressed, such glaring deficiencies diminished, leaving only its tendency for strange blunders and its weakness in the endgame.

The Transformer based model was similar in that it performed best in the opening, and became more problematic as the game progressed. However, while RWKV's weaknesses stemmed from difficulties in capitalizing on its advantages and completing tactical sequences, the Transformer model struggled with more fundamental chess concepts. It frequently failed to detect hanging pieces and often left its own pieces en prise, demonstrating a lack of tactical awareness. Although it played some opening moves that adhered to standard principles, it often deviated into unnatural and illogical continuations early in the game. By comparison, RWKV resembled a strong human player with a tendency to make occasional, inexplicable mistakes, while the Transformer remained at the level of a complete beginner, exhibiting only a superficial understanding of chess rules and making moves impulsively. From human play alone, RWKV appeared to have developed a significantly more advanced understanding of the game than the Transformer model.

5.2 Model v Model

We substantiate our findings by collecting statistics from simulated games played between our models. Given two models, we ran 100 games using the program described in Section 5, tracking game results, game lengths, and game accuracy.

- The first column tallies game results: we note that while the RWKV model has learned to checkmate, (whether by accident or not), all of the games played by the Transformer ended prematurely, indicating that it had not managed to learn legal moves (Table 2). A 1 represents a victory for the first player, a -1 a loss, and 0 a draw (neutral result).
- The second column tallies game lengths. The x-axis represent half-move number. We note that while distribution of game lengths is uniform in rows 2 and 3, probably a result of the random termination due to the Transformer model failing to yield legal moves. More interestingly, beyond a certain move number, all RWKV vs RWKV games end in a rule based result, indicating that the RWKV model has gained some measure of endgame ability. This seems to suggest that it has internalized important aspects of the game, and that some of its early terminations may be due to the extra complexity in the opening.
- The final column is a measure of the actual game-playing ability of the engines. We tallies the mean centipawn loss across all games on any given move. Again, the x-axis represents half-move number. That is, we measure the differences between the evaluation of the best move and that of the move made by the model. For our evaluation oracle, we employed Stockfish 14 @ 0.2 seconds per move. We note that the x-axis of the last two simulations are much shorter, due to the early resignation of the Transformer model.

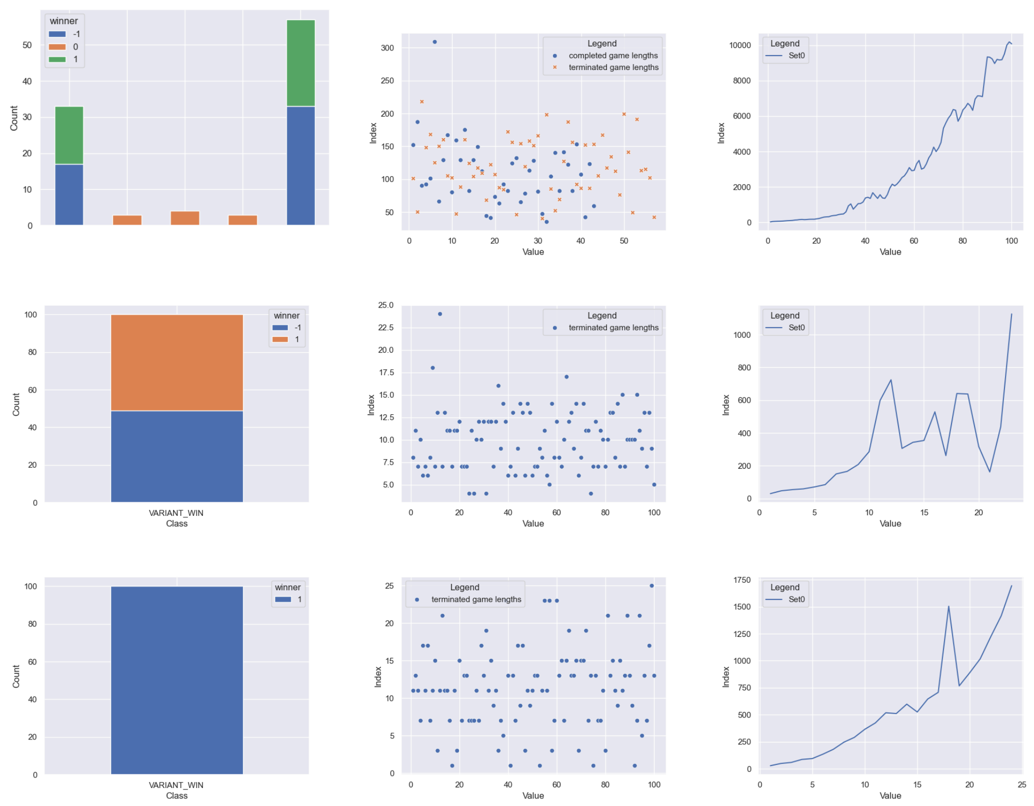


Figure 2: (Top): RWKV v RWKV. (Middle): Transformer v Transformer. (Bottom): RWKV vs Transformer. In the top left bar chart, the game result each bar represents are in order: checkmate, fivefold repetition, insufficient material, stalemate, and win by resignation (Opponent exceeded illegal move allowance)

6 World Representation

As a first step towards investigating whether our model learns to represent the board state, we first evaluate it on its ability to produce legal moves. We computed how often its top prediction was legal across games from the Lichess Elite and Synthetic (Section 2.2). Our results bore out the statistics observed from the simulations of the previous section: the RWKV model was significantly more able to make legal moves. Moreover, we note that it's ability was little hindered by the randomness of the game involved, despite only training on synthetic games for 40 epochs. Although our synthetic dataset is also randomly generated, due to the due to combinatorial explosion of possible chess positions, there is little chance of overlap and its ability cannot be accounted for by simple memorization.

We furthermore note that the Transformer model performed better on the human-play dataset, which is possibly less surprising. When considered together with the the results of Section 6.2, one might conjecture that the transformer model was learning something more akin to style rather than building an internal world model.

Table 2: Top-1 legality frequency

Variant	Legal% (Lichess Elite)	Legal% (Synthetic)
Transformer	39.8	28.2%
RWKV	68.9%	67.6%

6.1 Square Probe

Next, we wanted to know whether our models' internal activations contained a representation of the current board state. To study this question, we train probes that predict the board state from the network's internal activations after a given sequence of moves. For simplicity's sake, we trained to determine the occupation status of a given square. Since a board's state is is a linear function of the occupation status of each of its squares (disregarding non-Markovian properties like the 3-fold rule), our results pertain to the capability of the models to represent the underlying game state.

There are 6 piece types in the game of chess ([K,Q,R,N,B,P]). Accounting for both sides and the possibility of an empty square, this gives 15 possibilities. Yet, a linear probe trained on the 4th activation layer of our RWKV model was able reconstruct the state of the square E4 on random middle game positions sampled from games in the Lichess Elite validation dataset (Section 2.2) with 87.3% accuracy. We regard this as evidence of the RWKV's capabilities for world modelling. By contrast, the transformer model managed 61.9% accuracy on its best layer.

Contrary to [10], it seems that in our instance, linear probes were sufficient to capture information of the board state. We have no suggestion as to why there might be such a difference between Othello and Chess.

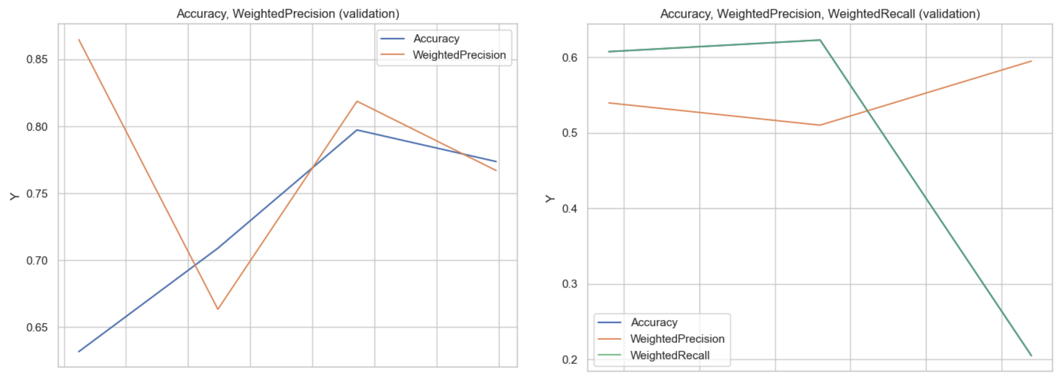
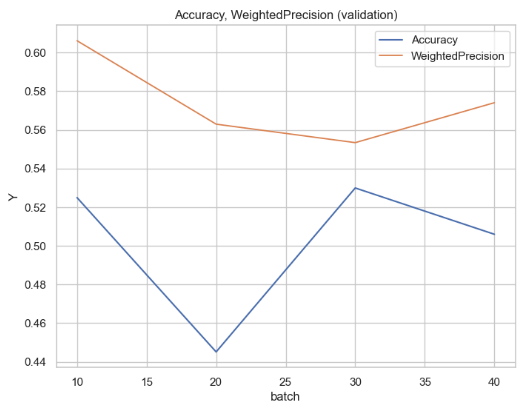


Figure 3: RWKV E4 probe on time-mixing activations across layers (3, 6, 9 12) vs Transformer probe on attention activations across layers (4, 8, 12)

6.2 Skill Probe

We were also interested whether the models internalized higher order features of the game such as move quality / player skill. We began by organizing our binned dataset (Section 2.2.2) into 4 tiers of [low, medium, high, top], corresponding to Lichess games where both players were in the range of 400-1800, 1900-2400, 2500-3200,2500-3200, 3300-4000 resp. We note that the dataset is imbalanced, due to the distribution of elos, but our examples were sufficient that our model saw a bit of everything. Then again, we trained a linear probe on specific activations within our models. The result was a bit surprising in that the Transformer model, which had underperformed RWKV model in all tests so far, seemed to have learned to encode skill-related information on its best layer to a degree comparable with the RWKV model. From the poor performance of other layers, it is evident that such performance is well above base line. To round out our analysis, we have included the confusion matrices for skill classification based on the activations on layer 8 of both models (Section A).



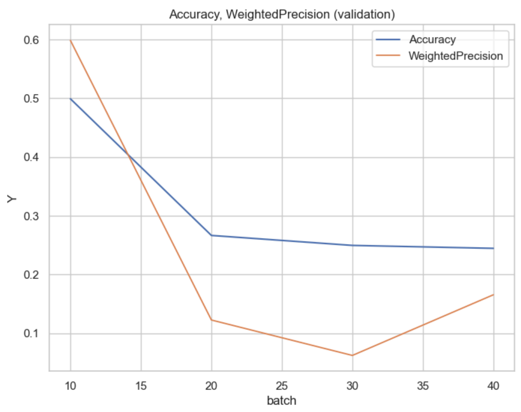


Figure 4: RWKV skill probe on time-mixing activations across layers vs Transformer probe on attention activations across layers (3, 6, 9, 12)

6.3 Commentary

Our work continues previous work [10], [20], [31], on investigating the the internal structures of sequence-to-sequence models. We applied pre-existing methods to the yet untested RWKV architecture and found behavior resembling that of transformer based models.

We found that the RWKV model was reasonably adept at learning rules and encoding the hidden game state, but that game skill was only partially represented in the layers we investigated. By contrast, the transformer model struggled to build a world representation, a conclusion substantiated by simulated games, legal move metrics, and internal probes about the board state.

We acknowledge some methodological weaknesses in our probe experiments. Since the distribution square occupation/game skills is unknown, it would have been better to present results from probes trained on randomly initialized model activations for a baseline comparison. Nevertheless, the performance of the less concomitant layers may serve as a useful baseline approximation.

7 Closing remarks

7.1 Further directions

Our work builds upon prior research in chess, interpretability, and sequence modeling, integrating ideas from each of these domains to deepen our understanding of how models develop internal "world" representations.

Our investigations into the representation capabilities of our models have left much on the table. One interesting line of inquiry is to studying the positional encodings: i.e. whether they are transposition invariant. Indeed, [19] noted that the positional encodings of their transformer architecture were integral to its performance, a better understanding of how to optimize the positional embeddings could be interesting. Indeed, Chess itself seems a promising test ground for interpretability: one could question how models understand procedural concepts such as endgames as well as structural concepts like positional features. Another possibility could be to adopt methodologies such as with richer input representations, rope embeddings, and simultaneous learning of policy from the LC0 architecture, and test the capabilities of a RWKV based evaluator, motivated by its performance in this study. We also note that training sequential models for game playing may still have their purposes, as they seemed particularly suited to capturing style. Finally, the broader question remains: to what extent can sequence-to-sequence models, develop reasoning capabilities without explicit search such as Monte Carlo? Previous works [18], [8] have shown evidence of this in Chess, but not in an explicitly the sequential paradigm. Our paper has demonstrated that the RWKV architecture is capable of developing reasoning capabilities to some degree within the sequential paradigm, but we remain cautious about the conclusiveness of these results and the extent to which they generalize to other domains or tasks.

Table 3: FEN evaluation metrics

Variant	Final CE-Loss	Accuracy	Weighted Precision
Transformer	2.7	18.1%	5%
RWKV	2.0	30.4%	27.9%

7.2 FEN Prediction

Motivated by the out-performance of the RWKV model, we decided to undertake a preliminary experiment to investigate whether it retained its advantage when used to construct a more traditional form of chess model. Previous works using transformers for the purpose of board evaluation models include [18], [19]. In particular, the latter represented the board state in a very simple way, using FEN strings (Section 2.2.5), to great success.

In our experiment, we also performed supervised classification of FEN strings, where the targets were binned stockfish evaluations of the corresponding positions. However, ours' was a much simplified setup, only aiming to compare the performance of Transformers and RWKV applied to a state-based prediction task. In particular, [18] employed various policies in addition to state-value prediction, as well as label-smoothing or loss ablation, whereas we do not. This, in addition to multitudinous other factors, likely contributes to the discrepancy between the accuracy of our models.

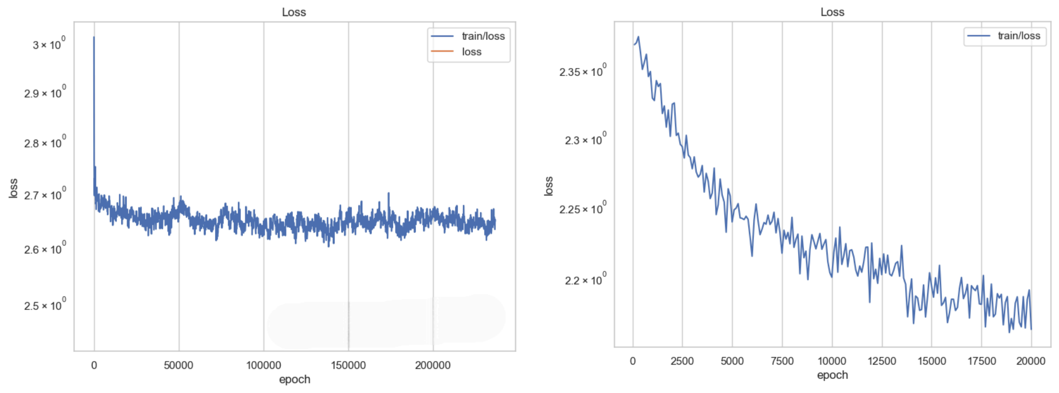


Figure 5: Transformer training progress vs RWKV training progress

Compared to the task of move prediction, both models encountered greater difficulty with evaluation prediction, as reflected by higher cross-entropy loss values (refer to Table Table 3). From the resulting confusion matrices (Figure Section A), we observe differences in the capabilities of the two models:

- The Transformer-based model struggled to evaluate the board, and did not improve significantly with training, despite training for longer.
- The RWKV model demonstrated a stronger ability to discern broad categories of evaluations. Specifically, it learned to distinguish whether positions were balanced or favored one side. However, the RWKV model was less able to differentiate between fine-grained evaluations (e.g., slight advantages or disadvantages), which aligns with the difficulty of this task even for domain experts in chess.

In light of the confusion matrix, we argue that the RWKV model's progress may even be somewhat underrepresented by the presented accuracy. Evaluation metrics like accuracy and cross-entropy loss may disproportionately penalize small errors, such as misclassifying evaluations that are close in magnitude. For example, incorrectly categorizing a slightly advantageous position as balanced may not represent a significant conceptual error, yet it contributes equally to the loss as larger misclassifications. A metric such as MSE or HL-Gauss [32] dist may be more faithful, and moreover paint the RWKV variant more favorably as well. Nevertheless, the given metrics are sufficient to represent the disparity between the transformer and RWKV.

7.3 Conclusion

In this paper, we explored the potential of the RWKV architecture as an alternative to the Transformer for sequence-to-sequence modeling in the domain of chess game sequences, focusing on their ability to form internal world models in pursuit of this task. Our experiments demonstrated that RWKV-based models offer distinct advantages over Transformers in learning world representations and predicting sequence outputs.

In our analysis, RWKV models achieved significantly higher success rates in generating legal moves, with an accuracy of up to 85% compared to the Transformer's 55%. RWKV models also displayed superior ability to internalize game dynamics, such as tracking defended squares and capturing the status of pieces, as evidenced by probe-based evaluations. Additionally, RWKV excelled in self-play and FEN-based evaluation tasks, showing understanding of complex concepts such as checkmating patterns and tactical sequences.

Leveraging the stateful and nuanced properties of chess, we investigated the little-tested capabilities of the RWKV architecture for world modelling, using methods from previous literature. We constructed square and skill probes, to measure how well specific features of the generating world were encoded within these models. Our findings showed that the RWKV model was reasonably adept at learning rules and encoding the hidden game state, while the transformer model struggled to do so.

Furthermore, we extended our experimental setup to also construct board-based evaluation models, to test whether the the capabilities of the RWKV carried over to the typical responsibilities of chess models. We showed that the RWKV model retained its advantage in the new task as well. Although constraints prevented us from achieving high accuracy, this initial test may indicate an interesting direction for the future.

Despite these promising results, our study has limitations. We focused on relatively small-scale models and simplified training setups, leaving questions about scalability to larger networks and datasets. Our inexperience may have contributed to methodological errors which caused the accuracy of our models to suffer. That is, while the RWKV out-performance Transformers in chess was clear in our experiment, the generalizability of these findings is unclear.

Our research builds on previous findings from the fields of chess, sequence modeling, and interpretability, thus reinforcing the importance of exploring models that can capture complex world representations. Several avenues of further research present themselves. As noted in Section 7.1, future work could investigate the potential of alternative architectures for chess evaluation networks, extend sequence models for game-playing tasks, and delve deeper into how internal world representations form in sequence models. More broadly, the game of chess serves as a promising test bed for studying how sequence-to-sequence models might develop sophisticated reasoning capabilities without resorting to explicit search mechanisms.

References

[1] A. Vaswani *et al.*, "Attention Is All You Need." Accessed: Dec. 21, 2024. [Online]. Available: http://arxiv.org/abs/1706.03762

[2] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate." Accessed: Dec. 21, 2024. [Online]. Available: http://arxiv.org/abs/1409.0473

[3] T. B. Brown *et al.*, "Language Models are Few-Shot Learners." Accessed: Dec. 21, 2024. [Online]. Available: http://arxiv.org/abs/2005.14165

[4] G. Team *et al.*, "Gemini: A Family of Highly Capable Multimodal Models." Accessed: Dec. 21, 2024. [Online]. Available: http://arxiv.org/abs/2312.11805

[5] J. Bai *et al.*, "Qwen Technical Report." Accessed: Dec. 21, 2024. [Online]. Available: http://arxiv.org/abs/2309.16609

[6] S. Hao *et al.*, "Training Large Language Models to Reason in a Continuous Latent Space." Accessed: Dec. 21, 2024. [Online]. Available: http://arxiv.org/abs/2412.06769

[7] OpenAI, "Learning to Reason with LLMs." Accessed: Dec. 21, 2024. [Online]. Available: https://openai.com/index/learning-to-reason-with-llms/

[8] E. Jenner, S. Kapur, V. Georgiev, C. Allen, S. Emmons, and S. Russell, “Evidence of Learned Look-Ahead in a Chess-Playing Neural Network.” Accessed: Dec. 20, 2024. [Online]. Available: http://arxiv.org/abs/2406.00877

[9] E. Nelson, N. Neel, and O. Catherine, “A Mathematical Framework for Transformer Circuits.”

[10] K. Li, A. K. Hopkins, D. Bau, F. Viégas, H. Pfister, and M. Wattenberg, “Emergent World Representations: Exploring a Sequence Model Trained on a Synthetic Task.” Accessed: Dec. 20, 2024. [Online]. Available: http://arxiv.org/abs/2210.13382

[11] B. Z. Li, M. Nye, and J. Andreas, “Implicit Representations of Meaning in Neural Language Models.” Accessed: Dec. 21, 2024. [Online]. Available: http://arxiv.org/abs/2106.00737

[12] M. Abdou, A. Kulmizev, D. Hershcovich, S. Frank, E. Pavlick, and A. Søgaard, “Can Language Models Encode Perceptual Structure Without Grounding? A Case Study in Color.” Accessed: Dec. 21, 2024. [Online]. Available: http://arxiv.org/abs/2109.06129

[13] J. Brinkmann, A. Sheshadri, V. Levoso, P. Swoboda, and C. Bartelt, “A Mechanistic Analysis of a Transformer Trained on a Symbolic Multi-Step Reasoning Task.” Accessed: Dec. 21, 2024. [Online]. Available: http://arxiv.org/abs/2402.11917

[14] B. Peng *et al.*, “RWKV: Reinventing RNNs for the Transformer Era.” Accessed: Dec. 20, 2024. [Online]. Available: http://arxiv.org/abs/2305.13048

[15] A. Gu and T. Dao, “Mamba: Linear-Time Sequence Modeling with Selective State Spaces.” Accessed: Dec. 21, 2024. [Online]. Available: http://arxiv.org/abs/2312.00752

[16] Y. Duan *et al.*, “Vision-RWKV: Efficient and Scalable Visual Perception with RWKV-Like Architectures.” Accessed: Dec. 20, 2024. [Online]. Available: http://arxiv.org/abs/2403.02308

[17] T. McGrath *et al.*, “Acquisition of Chess Knowledge in AlphaZero,” *Proceedings of the National Academy of Sciences*, vol. 119, no. 47, p. e2206625119, Nov. 2022, doi: 10.1073/pnas.2206625119.

[18] A. Ruoss *et al.*, “Amortized Planning with Large-Scale Transformers: A Case Study on Chess.” Accessed: Dec. 17, 2024. [Online]. Available: http://arxiv.org/abs/2402.04494

[19] M. Daniel, “Transformer Progress - Leela Chess Zero.” Accessed: Dec. 21, 2024. [Online]. Available: https://lczero.org/blog/2024/02/transformer-progress/

[20] K. Adam, “Chess-GPT’s Internal World Model.” Accessed: Dec. 20, 2024. [Online]. Available: https://adamkarvonen.github.io/machine_learning/2024/01/03/chess-world-models.html

[21] S. Toshniwal, S. Wiseman, K. Livescu, and K. Gimpel, “Chess as a Testbed for Language Model State Tracking.” Accessed: Dec. 20, 2024. [Online]. Available: http://arxiv.org/abs/2102.13249

[22] “Lichess Elite Database – Learn from the strongest!.” Accessed: Dec. 21, 2024. [Online]. Available: https://database.nikonoel.fr/

[23] “ezipe/lichess_elo_binned_debug at main.” Accessed: Dec. 23, 2024. [Online]. Available: https://huggingface.co/datasets/ezipe/lichess_elo_binned_debug/tree/main

[24] “Download PGN Files.” Accessed: Dec. 23, 2024. [Online]. Available: https://www.pgnmentor.com/files.html

[25] “TWIC Archive | The Week in Chess.” Accessed: Dec. 23, 2024. [Online]. Available: https://theweekinchess.com/twic

[26] “Forsyth–Edwards Notation.” Accessed: Dec. 21, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Forsyth%E2%80%93Edwards_Notation&oldid=1196548074

[27] “Chess Evaluations.” Accessed: Dec. 21, 2024. [Online]. Available: https://www.kaggle.com/datasets/ronakbadhe/chess-evaluations

[28] O. Press and L. Wolf, “Using the Output Embedding to Improve Language Models.” Accessed: Dec. 22, 2024. [Online]. Available: http://arxiv.org/abs/1608.05859

[29] Andrej, “karpathy/nanoGPT.” Accessed: Dec. 22, 2024. [Online]. Available: https://github.com/karpathy/nanoGPT

[30] P. Bo, “BlinkDL/nanoRWKV.” Accessed: Dec. 22, 2024. [Online]. Available: https://github.com/BlinkDL/nanoRWKV

[31] "TransformerLensOrg/TransformerLens." Accessed: Dec. 23, 2024. [Online]. Available: https://github.com/TransformerLensOrg/TransformerLens

[32] E. Imani and M. White, "Improving Regression Performance with Distributional Losses." Accessed: Dec. 23, 2024. [Online]. Available: http://arxiv.org/abs/1806.04613

[33] A. Radford, R. Jozefowicz, and I. Sutskever, "Learning to Generate Reviews and Discovering Sentiment." Accessed: Dec. 20, 2024. [Online]. Available: http://arxiv.org/abs/1704.01444

[34] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, "RoFormer: Enhanced Transformer with Rotary Position Embedding." Accessed: Dec. 21, 2024. [Online]. Available: http://arxiv.org/abs/2104.09864

[35] "https://arxiv.org/pdf/2412.06769v1." Accessed: Dec. 21, 2024. [Online]. Available: https://arxiv.org/pdf/2412.06769v1

[36] L. Sutawika *et al.*, "EleutherAI/lm-evaluation-harness: v0.4.3." Accessed: Dec. 21, 2024. [Online]. Available: https://zenodo.org/doi/10.5281/zenodo.12608602

[37] D. Monroe and P. A. Chalmers, "Mastering Chess with a Transformer Model." Accessed: Dec. 21, 2024. [Online]. Available: http://arxiv.org/abs/2409.12272

[38] "Visualizing Chess Game Length and Piece Movement♟." Accessed: Dec. 22, 2024. [Online]. Available: https://kaggle.com/code/ironicninja/visualizing-chess-game-length-and-piece-movement

[39] "LCZero 0.31-dag-f7fb268-BT4-6147500-it332 vs Stockfish dev-20241208-cf10644d - TCEC - Live Computer Chess Broadcast." Accessed: Dec. 23, 2024. [Online]. Available: https://tcec-chess.com/

A Confusion matrices

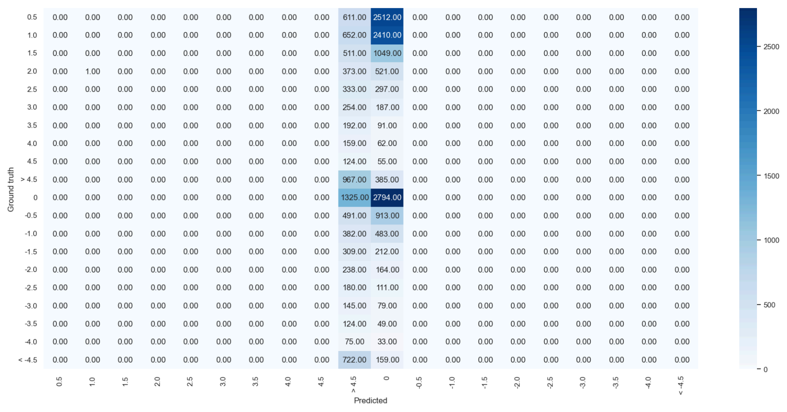


Figure 6: Transformer value prediction

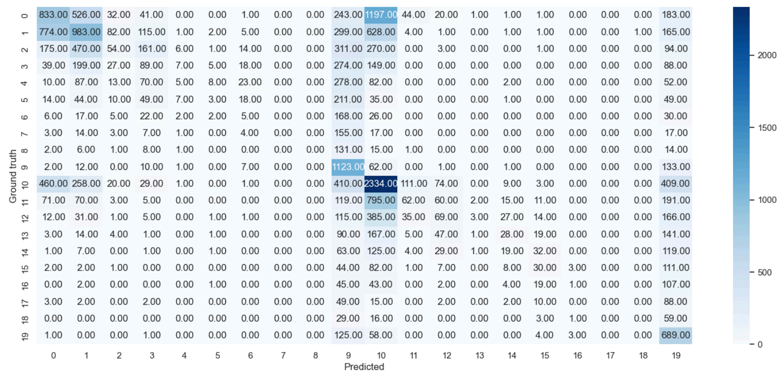


Figure 7: RWKV (The labels are in the same order as for the transformer matrix)

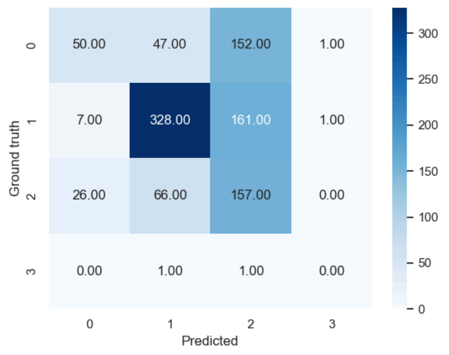


Figure 9: RWKV skill probe on layer 8.[4]

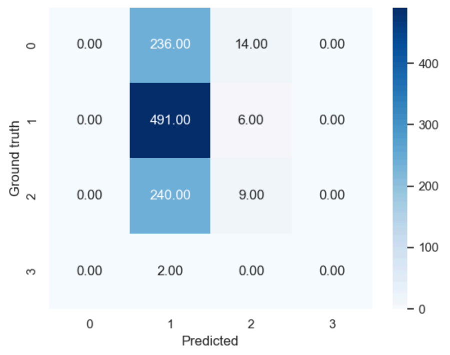


Figure 10: Transformer skill probe on layer 8.

B Code

Code resides at https://github.com/manyhue/seq_chess_probe.git

[4]The labels correspond skill levels of [high, low, medium, top]