Beyond Evaluation: Learning Contextual Chess Position Representations

Ben Hull https://bluehood.github.io/

Abstract

This paper introduces `ChessLM`, a novel Transformer-based model designed to learn rich, contextual vector representations (embeddings) of chess positions. Moving beyond traditional chess engines focused on move evaluation, our approach is inspired by the success of self-supervised learning in Natural Language Processing. We adapt the Vision Transformer architecture and train the model using two self-supervised tasks on a large corpus of chess games: Masked Piece Prediction, which requires the model to infer masked pieces based on board context, and Move Difference Prediction, which involves predicting the number of moves between two positions from the same game. Qualitative analysis demonstrates the model's ability to capture high-level structural and thematic similarities between positions across different game phases (opening, middlegame, endgame), such as shared pawn structures, material imbalances, and king safety patterns. However, the analysis also reveals limitations that would significantly impact our model's ability to generate position evaluations. Despite this, the learned embeddings are shown to be effective for retrieving strategically or tactically similar positions, enabling applications such as intelligent chess puzzle suggestions. This work contributes a novel method for learning chess position representations that encode thematic information, opening avenues for future research in applying representation learning to various chess-related tasks beyond traditional evaluation.

Keywords: Chess, Transformers, Machine Learning, Embeddings

1 Introduction

Chess has long been the subject of computational advancements, from Deep Blue famously beating Garry Kasparov in 1997 to modern-day engines, such as StockFish, which far surpass human abilities when playing chess. Much of the focus of chess computation has been on playing the game, or, more specifically, determining the next best move given a current position. However, this work takes a different approach, inspired by the success of Transformers in the Natural Language domain, we develop a model designed to learn complexities and themes of a position without explicitly training for next move prediction.

The primary objective of this work is to develop a robust model capable of generating meaningful vector representations, or embeddings, for chess positions. Such embeddings aim to capture the complex strategic and tactical themes inherent of a chess position. The availability of high-quality chess embeddings opens avenues for numerous downstream applications. These include enhancing chess engines or analysis tools through Retrieval-Augmented Generation (RAG), where similar historical positions and their analyses can provide additional context; enabling classification tasks, such as identifying specific tactical

themes or positional imbalances; and potentially facilitating regression tasks, like predicting game outcomes or evaluating the quality of moves based on the resulting position.

1.1 Previous Work

The task of representing chess positions computationally has evolved significantly alongside advancements in artificial intelligence and machine learning. Early approaches often focused on handcrafted features or direct evaluation functions used within traditional chess engines. However, the rise of deep learning has spurred interest in learning representations directly from data.

One line of research focused on learning vector representations for individual chess pieces rather than entire positions. For instance, `Chess2vec` [1] explored methods like Principal Component Analysis (PCA) on move counts to generate embeddings for pieces, aiming to capture their latent movement structures. While insightful for understanding piece relationships, this does not directly provide contextual embeddings for complex board states. Input representations for early neural network models often involved bitboards (representing the board as an 8x8x12 tensor for piece types and colours) or simpler vectorisations [2; 3].

Much of the deep learning work in chess, including early applications of Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs), concentrated primarily on position evaluation (predicting scores like centipawn values) or move prediction to improve playing strength [2]. DeepChess, for example, used a deep neural network for pairwise position evaluation to achieve high-level play [1]. While effective for creating strong engines, the learned representations were optimised for evaluation, not necessarily for capturing thematic or strategic similarity useful for tasks like position retrieval. Autoencoders have also been explored, such as using a convolutional autoencoder trained with a reconstruction objective (Mean Squared Error) to produce position embeddings [4].

More recently, Transformer architectures [5] have been applied to chess, demonstrating strong performance in various tasks. Several studies have successfully employed Transformers for building high-performance chess-playing engines, often trained via supervised learning on engine evaluations (distilling knowledge from engines like Stockfish) or through self-play reinforcement learning [6; 7]. These models typically focus on predicting optimal moves, action-values, or game outcomes, and often incorporate game history or specialised input encodings beyond the immediate board state. Other Transformer applications include generating realistic chess positions [8], generating move-by-move commentary [9], or even identifying player style through behavioural stylometry [10]. Some research also delves into interpreting Transformer attention maps to understand the learned chess concepts [11].

Our work with `ChessLM` diverges from these prior approaches in several key aspects. Firstly, the primary objective is not direct evaluation or move prediction for gameplay, but rather learning rich, contextual embeddings that capture thematic and strategic similarities between positions, regardless of the immediate evaluation score. Secondly, we adapt the Vision Transformer (ViT) architecture [12], treating the board as a spatial input, differing from models processing sequences of moves or using standard CNNs. Most importantly, `ChessLM` employs novel self-supervised pre-training tasks specifically designed for learning positional context: Masked Piece Prediction (MPP) and Move Difference Prediction. These tasks encourage the model to understand piece relationships, board structure, and game

dynamics in a way distinct from minimising evaluation error, maximising move prediction accuracy, or minimising reconstruction loss. While inspired by self-supervised learning in NLP [13], our tasks are tailored to the chess domain. By focusing on learning thematic representations through these unique self-supervised objectives and a ViT-based architecture, `ChessLM` contributes a new methodology for generating chess position embeddings suited for downstream tasks like similarity search and retrieval-augmented analysis, complementing existing work focused on evaluation and playing strength.

1.2 Background: Transformer Architectures

The core technology underpinning our model is the Transformer architecture, first introduced by Vaswani et al.[5]. Transformers have revolutionised sequence processing tasks, particularly in Natural Language Processing (NLP), due to their effective use of self-attention mechanisms. Unlike recurrent or convolutional networks, self-attention allows the model to weigh the importance of different elements in the input sequence relative to each other, regardless of their distance. This capability is crucial for capturing long-range dependencies and complex interactions between words and their meanings.

Inspired by the success of models like BERT (Bidirectional Encoder Representations from Transformers)[13], which demonstrated the power of pre-training large Transformer encoders on self-supervised tasks (like Masked Language Modeling), we adapt this approach to chess. BERT learns rich contextual representations of language by predicting randomly masked words in a sentence. Similarly, our approach leverages self-supervised tasks tailored to chess to create a model with understanding of board positions.

1.3 Model Architecture

The model developed for this project, named `ChessLM`, adapts the Vision Transformer (ViT) [12] concept for chess board analysis. The architecture processes the 8x8 board as follows:

- **Input Embedding:** Each square on the board is initially represented by its piece value (e.g., 1 for White Pawn, -4 for Black Rook). A linear layer projects this single value into a higher-dimensional space. Learnable positional embeddings are added to each square's representation to provide spatial information. Additionally, a learnable embedding representing the current turn (either white or black represented as 0 or 1) is added.
- **Task Tokens:** Following common practice in multi-task Transformer models, special learnable tokens are prepended to the sequence of square embeddings. These tokens serve as aggregators of global board information, and their corresponding output vectors are used as input to the task-specific prediction heads.
- **Transformer Encoder:** The core of the model consists of a stack of 6 Transformer encoder layers. Each layer employs multi-head self-attention (with 8 heads) and a feed-forward network. We use pre-layer normalisation for improved training stability. A dropout rate of 0.1 is applied within the encoder layers for regularisation.
- **Task-Specific Heads:** Two separate heads branch off from the output corresponding to the task tokens:

  - `mpp_head`: A Multi-Layer Perceptron (MLP) that takes the output representation and predicts the identity of masked pieces (12 possible piece types, excluding empty squares).
  - `moves_head`: An MLP that takes the concatenated output representations from two different board states from the same game and predicts the number of moves required to transition between them. That is, the number of half moves that would need to be made to get from one position to the other.

The model comprises of approximately 4.5 million total parameters, all of which are trainable. This architecture is designed to process the entire board context simultaneously via the self-attention mechanism, enabling it to learn complex relationships between pieces and squares. The model outputs an embedding vector of dimension 256.

1.4 Datasets

Training leverages two distinct datasets, pre-processed into structured formats to facilitate the self-supervised tasks. These datasets are derived from a large corpus of chess games and positions, available from the Lichess database[14] and the computerchess[15] database.

- **Masked Piece Prediction Dataset (MPPDataset):** This dataset provides samples specifically for the MPP task. Each sample consists of:
  - A board state, where the numbers -6 through 6 represent the pieces on the board, where a subset of pieces have been masked (their values replaced with a mask token).
  - The true values (identities) of the masked pieces. Target sequences are padded to a maximum length (6) to enable batching.
  - A mask indicating which target piece predictions are valid (i.e., correspond to originally masked pieces).
  - The turn (White or Black) of the position.

  This structure allows the model to receive a partially obscured board and learn to predict the missing pieces based on context.

- **Moves Difference Prediction Dataset (MovesDataset):** This dataset provides pairs of board states for the moves prediction task. Each sample contains:
  - The starting board state and the turn corresponding to that state.
  - The ending board state and the turn corresponding to that state.
  - The number of moves (plies) required to transition from the start state to the end state.

  This format enables the model to compare two positions and learn to estimate the distance between them.

1.5 Training Regime

To encourage the model to learn comprehensive representations of chess positions, we employ a multi-task learning strategy combining two self-supervised objectives, mirroring techniques used in large language model pre-training:

1. **Masked Piece Prediction (MPP):** Analogous to BERT's Masked Language Model task, a random subset of pieces on the input board are masked (replaced with a mask token). The model's objective is to predict the original identity of these masked pieces based on the surrounding context (the remaining pieces and whose turn it is). This task allows the model to understand typical piece configurations, legal placements, and the relationships between pieces. For MPP 10% of the pieces were masked.

2. **Moves Prediction:** This task involves presenting the model with two distinct board states (a start and an end position) from actual game sequences. The model must predict the number of moves (plies) separating these two positions. This objective encourages the model to learn about piece mobility, game dynamics, and the plausible evolution of a position over time.

The model was trained end-to-end using the AdamW optimiser[16] with a weight decay of 0.01. A Cosine Annealing Learning Rate schedule is applied over $N = 30$ epochs to adjust the learning rate dynamically. Gradients are clipped to a maximum norm of 1.0 to prevent exploding gradients during training. Data is processed in batches of 32 with shuffling enabled for the training sets. The losses from both tasks are implicitly combined as they share the encoder and optimizer, contributing jointly to the gradient updates.

The model was trained on a single P100 GPU for approximately 10 hours on a dataset of $100,000$ games, with 5 samples taken from each game, for a total of $500,000$ training samples. 90% of the dataset was used for training and 10% for training validation.

2 Results

The trained model can now create embeddings with 256 dimensions representing the board state of a given position. However, without a standardised benchmark to test the model against for chess position understanding, we are limited to a manual qualitative evaluation of the results.

We first create a dataset consisting of one million positions derived from real games and chess puzzles. Each entry contains the FEN representation of the board, alongside the embedding generated for that position.

We then use cosine similarity to compute the similarity between a given position and all other positions in the dataset. This will allow us to select the top three most similar positions based on our embedding model. There are many such comparisons we can analyse, however, we shall discuss three here, ranging from opening, middle and endgame positions. Further examples of compared positions can be seen in Appendix 1.

2.1 Opening

The following figure (Figure 1) shows the position of an opening after nine moves. We have also included the top three most similar positions, as calculated by our embedding model.

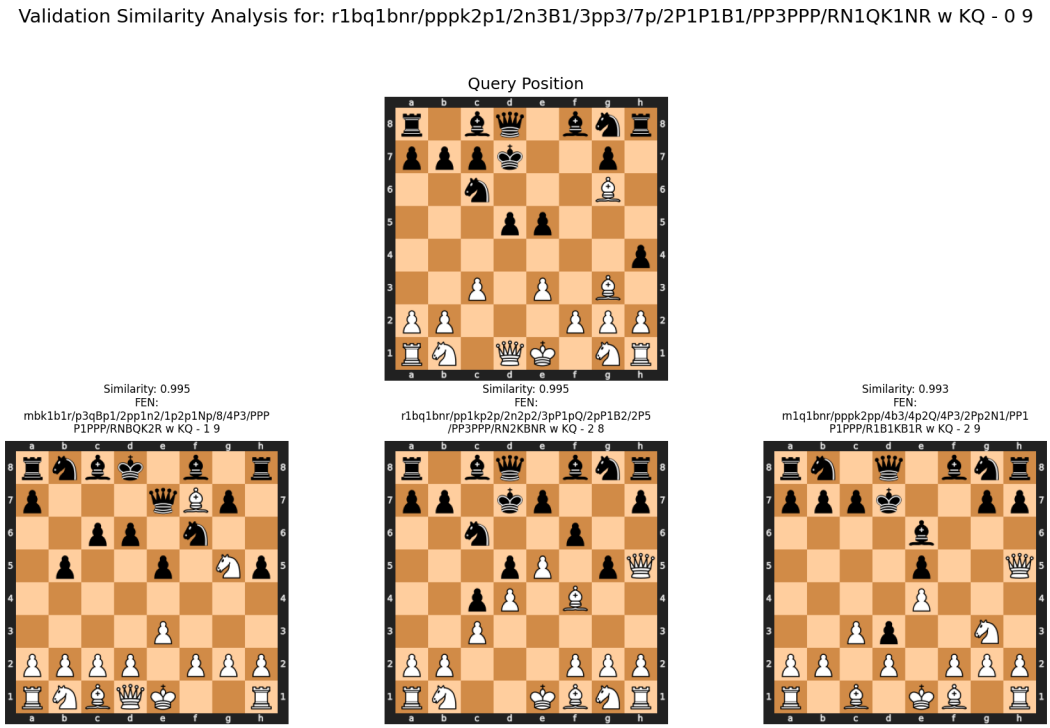


Figure 1: An opening position after nine moves with similar positions calculated using the embedding model.

The Stockfish evaluation for the query positon of +2.12 at depth 25 strongly suggests a significant advantage for White. The exposed Black king, having lost castling rights, is a major factor contributing to this. White's active Queen on d1 and Bishop on g3 exert considerable pressure on the central squares, particularly targeting the d5 pawn and potentially looking to exploit weaknesses around the Black king. The Bishop on g6 is also well-placed, limiting Black's king's mobility and potentially participating in future attacks. While White has two undeveloped pieces (the Knight on b1 and Rook on a1), Black's three undeveloped pieces (the Bishop on c8, Rook on a8, and Rook on h8) indicate a clear lead in development for White, allowing them to dictate the flow of the game and create threats more readily. The isolated black pawn on h4 could also become a target for White later in the game.

In the first position, with a Stockfish evaluation of +1.83, echoes the themes of the query position, indicating a notable advantage for White. The White Bishop on f7 is putting significant pressure on the black position, with the Black king having lost castling rights. This forces the Black king to remain in the centre, further hindering Black development and safety. The lead in development for White is also present here, likely stemming from similar opening moves. While Black possesses good central pawn control with pawns on d6, c6, and e5, White's well-placed Knight on g5 puts pressure on f7 and h7, a direct threat to Black's kingside. The ability for White to castle kingside immediately offers a significant advantage in terms of king safety and allows for better coordination of their rooks.

In the second position, the evaluation of +1.05 for White suggests a less decisive advantage compared to the previous two, but still indicates White is better off. Black has also lost castling rights due to the White Queen on h5 is a significant tactical detail. This active placement of the White Queen not only prevents Black from castling but also exerts direct pressure on the h7 pawn and potentially other targets on the kingside. The pawn structure indeed differs significantly, with both players having advanced central pawns (White on e5 and d4, Black on d5 and c4), creating a dynamic and contested centre.

This third position presents a contrasting scenario where Black holds an advantage (−1.01) after 9 moves, primarily due to the capture of a White Knight. Despite White having checked the Black king with their Queen on h5, the material advantage for Black is a significant factor. The similar development of pieces between White and Black suggests that the game has been relatively balanced in terms of piece deployment, up until the knight capture. However, Black's overextended pawn on c3 represents a clear weakness that White is likely going to target. This pawn is advanced and unsupported, making it vulnerable to attack, particularly from White's Bishop on f1. While Black has a material advantage, their king is still in the centre and potentially exposed to further checks from the White Queen, and the weakness on c3 could provide White with counterplay and opportunities to regain the initiative.

Across these four positions, several similarities and differences emerge. A common theme is that White often possesses an early initiative and a lead in development, reflected in the positive Stockfish evaluations for the first two positions. The vulnerability of the Black king, often losing castling rights early in the game, is another recurring motif. Central control is consistently a crucial factor, with both players frequently contesting the central squares with pawns and minor pieces.

However, some differences are also clear. The material balance varies, with Black having a material advantage in the third position. The pawn structures are quite distinct in each scenario, influencing the strategic possibilities and potential weaknesses for both sides. The level of White's advantage fluctuates, ranging from a significant +2.12 to a more modest +1.05, demonstrating the dynamic nature of chess positions even within a few moves of each other.

2.2 Middlegame

The next figure (Figure 2) shows the position of a middlegame after 23 moves. We have also included the top three most similar positions, as calculated by our embedding model.

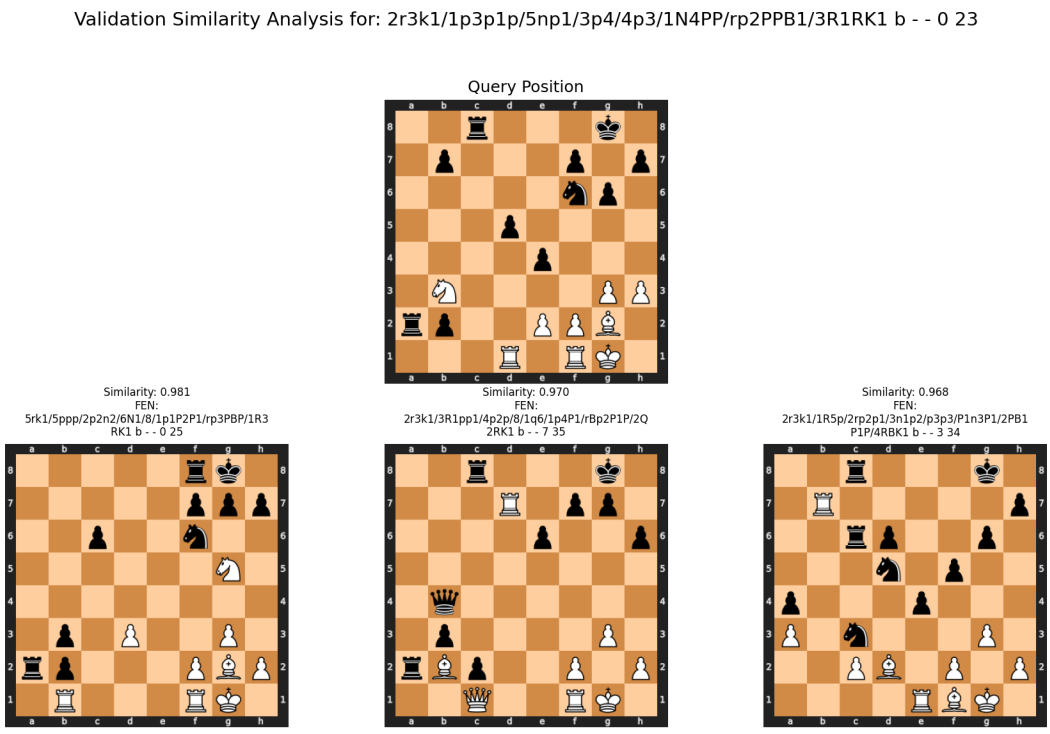


Figure 2: A middlegame position after 23 moves with similar positions calculated using the embedding model.

In the query position, Black has the advantage with an engine analysis of −1.91. Both sides have nearly equal material, although White is up a piece for three pawns. Both sides are castled kingside and have well-protected kings. Black is exerting significant pressure in the centre with their two centrally connected pawns and control of the open c file. Black is also close to promoting their pawn on b2. White has fianchettoed their light-squared bishop.

In the first position, we have a situation similar to the query position. However, in this case, White has the slight advantage with an engine score of 0.54. Black is down a minor

piece but is compensated with two extra pawns, giving White a slight material advantage. Black is also close to promoting their b2 and b3 pawns, with White defending with their rooks. Blacks rook is also similarly placed on a2 as in the query position. Both sides are similarly castled on the kingside, and white has also fianchettoed their light-squared bishop. The major difference seems to be Black's lack of central control without their central pawn structure.

Our second position also has similar themes. Black and White are also both castled kingside, with White having appeared to fianchettoed their light-squared bishop earlier in the game (although that piece has been removed from the board). Each side also has a similar amount of material on the board, with Black down a minor piece with advanced pawns on the lefthand side of the board. Black is applying pressure and threatening to promote their pawns, a similar theme to the query position. White is defending against this threat. Both kings are safe from immediate threats from the opposition. The main differences from the query position are that both players have a Queen, which is not the case in the query position. Black also enjoys a significant advantage according to the engine of -4.90. Moreover, there is limited activity in the middle of the board, with Black not maintaining their central advantage, as was the case in the query position. However, it could be argued that Black still has the central advantage as they have a central pawn on e6 (and White has no central pawns).

The last position seems somewhat different to the query position, however, there are still themes that were present in the query position. The material is relatively similar, with White having a single pawn advantage. However, in this position, each side has the same number of minor pieces. Black is also not threatening to promote their pawns, which was a key theme in earlier positions. Similarly, though, Black is maintaining central pressure with their pawn structure and central knights. Both kings are also relatively safe, having both castled kingside. Additionally, White has also fianchettoed their light-squared bishop earlier in the game, a theme across all positions. Continuing the variability in engine scores, Black has the advantage with a score of -1.76.

In the query position, Black holds a significant advantage with an engine evaluation of -1.91. Key features include Black's strong central control via connected pawns, pressure along the open c-file, and an advanced pawn threatening promotion on b2. Material is nearly equal, though White is a piece up for three pawns. Both kings are safely castled kingside, and White's light-squared bishop is fianchettoed. Comparing this to the first position, the engine evaluation flips to a slight White advantage of 0.54. While themes like fianchettoed bishops, kingside castling, and Black's advanced b-pawns defended by White rooks are present in both, the crucial difference lies in Black's lack of central control in the first position compared to the query. In the first position, Black is down a minor piece compensated by two pawns, giving White a small material edge, whereas material is more balanced in the query.

Moving to the second and third positions, we find further variations. The second position features a substantial Black advantage of -4.90. Unlike the query, Queens are present on the board. While both sides are castled kingside and Black has advanced pawns on the queenside similar to the promotion threat in the query, the central dynamics differ. The description notes limited central activity initially, contrasting with Black's strong central control in the query, although Black does retain a central pawn on e6. Material is described

as similar, with Black down a minor piece but with advanced pawns. The third position, rated −1.76 in Black's favor, shares the theme of Black maintaining central pressure, but here it's achieved through pawns and knights rather than primarily connected central pawns controlling an open file as in the query. Promotion threats are absent in the third position, unlike the query and second position. Material is similar in the third position to the query, with White having a single pawn advantage and an equal number of minor pieces, a difference from the material imbalance involving piece vs. pawns in the query and first two positions. All positions share the theme of kingside castling and White having a fianchettoed light-squared bishop.

Across all four positions, several key themes are consistently present. In each case, both White and Black have castled kingside, suggesting a similar early-game development phase focused on king safety. A recurring feature is White's fianchetto of the light-squared bishop. Material balance is relatively similar across the positions, generally featuring near equality or minor imbalances, often involving a minor piece deficit for Black compensated by pawns. Black often maintains some form of central presence or potential, either through strong central pawns (query, last position) or central activity (last position). Furthermore, the kings in all positions is relatively safe from immediate threats. The first and second positions also share the theme of Black having advanced pawns, particularly on the queenside, threatening promotion, similar to the threat in the query position.

Despite the similarities, significant differences exist between the query position and the other three. The most obvious difference is the engine evaluation, which varies considerably: −1.91 for Black in the query, 0.54 for White in the first, −4.90 for Black in the second, and −1.76 for Black in the last. This indicates vastly different strategic and tactical situations. While the query position features strong central control by Black via connected pawns and control of the c-file, in the first position Black's lacks central control due to the absence of a central pawn structure. The second position also highlights limited central activity compared to the query, although Black has a central pawn on e6. Material imbalances differ; while the query has White up a piece for three pawns, the first and second positions have Black down a minor piece for fewer pawns. A major difference in the second position is the presence of Queens for both players, which are absent in the query position. Finally, the threat of pawn promotion, a key feature of the query and the first two positions, is notably absent in the last position.

2.3 Endgame

The endgame position in figure (Figure 3) shows the position of a game after 45 moves. We have also included the top three most similar positions, as calculated by our embedding model.

The query position shows an endgame position where White has the clear advantage, with two pawns of extra material. White has three passed pawns that will threaten promotion – and notably two on the Queenside that the Black king must attempt to defend against. Black only has three pawns on the Kingside, defended by two White pawns, that could threaten promotion if White does not play optimally. Indeed, White has a +5.74 advantage according to the engine.

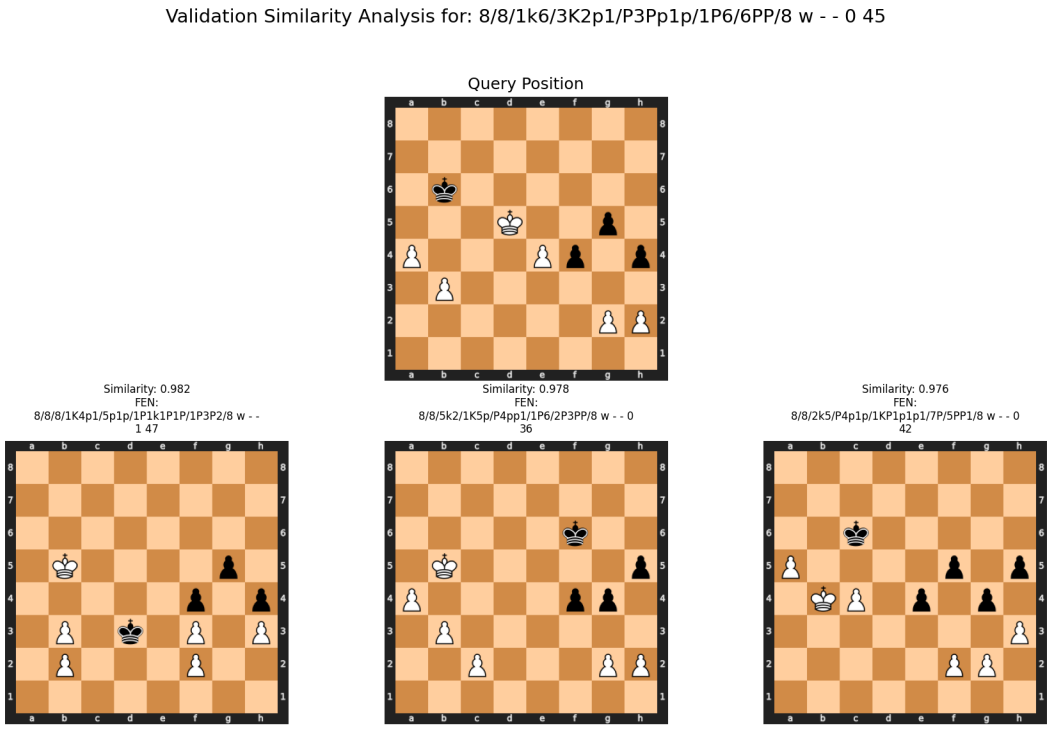


Figure 3: A endgame position after 23 moves with similar positions calculated using the embedding model.

In the first position, we have a similar pawn structure, with Black having three pawns on the Queenside in the same position as the Query position. White pawn structure differs slightly with three pawns on the Queenside in front of Black's pawns. White also has passed pawns on the Kingside, although they are doubled. The material difference is also identical to the Query position. The main difference in this position is the location of the Kings – Black's king is closer to the action, threatening White's pawns. White's queen is further away from the action but is supporting their past pawns. This difference makes the position drawn, with an engine rating of 0.00. The position is drawn after both sides promote and trade off Queens.

The second position is also rather similar to the Query position, with White having a clear Kingside pawn majority threatening promotion. Black also has a similar Queensuide pawn structure, with White's Queenside pawn structure being identical. The clear difference here is that Black is threatening to promote before White can, and White's king will have to backtrack to stop the promotion. However, White has the clear advantage due to their Queenside pawns, and this is reflected in the +5.73 evaluation.

The final position is similar to the Query position, with White having two passed pawns on the Kingside with three pawns vs. Black's four on the Kingside. The Kings also have a similar position on the board. Unlike the Query position, Black is threatening to promote

with their pawn majority on the Queenside. White will have to backtrack their King to protect against this. The position is relatively even, with White having the slight advantage at a +0.30 evaluation.

The query position and the three positions share several key characteristics, primarily revolving around a material imbalance favouring White. In all positions, pawn structures play a crucial role, with both sides possessing pawns on opposite flanks that could potentially become passed and threaten promotion. Specifically, the threat of passed pawns, particularly for White, is a recurring theme, often shaping the flow of the game. The material advantage of two pawns for White is consistent between the query position and the other positions.

Despite these similarities, significant differences exist that lead to different evaluations and outcomes compared to the query position's clear White advantage (+5.74). In the first similar position, the main difference lies in the kings' activity; Black's king being closer to White's attacking pawns and White's queen being less optimally placed neutralises White's material and structural advantages, resulting in a 0.00 evaluation and a drawn outcome after queens are traded. The second position maintains a high evaluation for White (+5.73) due to their strong Kingside pawn majority and favourable Queenside structure, despite Black's temporary threat of promotion, which White can counter. The final position presents a more balanced scenario (+0.30) where Black's Queenside pawn majority poses a significant threat, forcing White's king to defend and reducing White's overall advantage compared to the dominant position in the query, even though White still has passed pawns on the Kingside. These variations in king placement, the specific configuration of passed pawns, and each side's pawn structure give rise to positions with different balances.

Our model demonstrates a notable strength in identifying positions that share overarching structural and thematic similarities, even across different phases of the game (opening, middlegame, endgame). The analysis consistently highlights shared features such as material imbalances, recurring pawn structures on opposite flanks, the presence or threat of passed pawns, patterns of king safety (like Kingside castling), and the fianchetto of specific pieces (like White's light-squared bishop in the middlegame examples). The model appears capable of capturing these broader positional motifs – features that contribute to the general 'look and feel' of a position -– suggesting it is learning to recognise common strategic elements and pawn formations that appear frequently in chess games. This ability to group positions based on these high-level similarities could be valuable for tasks requiring the retrieval of similar chess positions.

However, the analysis also reveals significant shortcomings in the model's ability to determine nuances that fundamentally alter the evaluation and outcome of a position. While the model identifies positions as similar based on shared themes, the engine evaluations provided show vast discrepancies between the query positions and the 'similar' ones. The text points out that subtle differences, such as the precise location and activity of the kings, the exact configuration and threat level of passed pawns, specific material imbalances (piece vs. multiple pawns), or the presence/absence of key pieces like the Queen, have a

dramatic impact on the position's assessment. The model's similarity scores do not seem to adequately penalise these crucial differences, indicating that it may be overly focused on structural resemblances rather than features that drive the actual evaluation of a chess position. This is to be expected as our model was not trained for position evaluation, but rather to learn general themes and 'feel' of chess positions. Nevertheless, this model is unlikely to be successful as a base model for chess position evaluation.

3 Application: Puzzle Suggestions

Now that we have an embedding model that provides good recommendations of similar positions (given a database of positions), we can look at some applications. One of these applications is suggesting new chess puzzles to a user. The user provides a position: either a puzzle or a position they've seen in a game and the application suggests a puzzle for them based on the cosine similarity of a database of puzzles.

This section will look at the implementation of such an approach.

3.1 Data Preparation

Before the application can run, a preparatory step generates the necessary data:

- A large dataset of chess puzzles (from Lichess [14]) is processed.
- Our chess encoder model is used to generate the embeddings for the starting position (FEN) of each puzzle.
- These embeddings, along with puzzle metadata (FEN, solution moves, themes), are stored.

This will serve as a database of positions that we can use to make recommendations of puzzles to the user.

3.2 Application Workflow

We'll build a simple app in Flask to allow users to receive puzzle recommendations based on a chess position. The application follows these steps:

The user submits the FEN of the position they want to find similar puzzles for:

- **Load Data:** The Flask backend server loads the pre-computed puzzle embeddings and metadata from the database into memory.
- **Load Model:** The encoder model used for data preparation is loaded.
- **Generate Input Embedding:** The backend generates a vector embedding for the user-provided FEN using the encoder model.
- **Calculate Similarity:** Cosine similarity is calculated between the input FEN's embedding and all the pre-computed puzzle embeddings stored in the database.
- **Identify Top Matches:** The puzzles are ranked by similarity score, and the top N (e.g., 10) most similar puzzles are identified.
- **Random Selection:** One puzzle is randomly selected from the top N set (although other selection choices are available).
- **Prepare Response:** The selected puzzle's starting FEN, its associated themes, and the complete sequence of solution moves (in UCI format) are packaged into a JSON response.

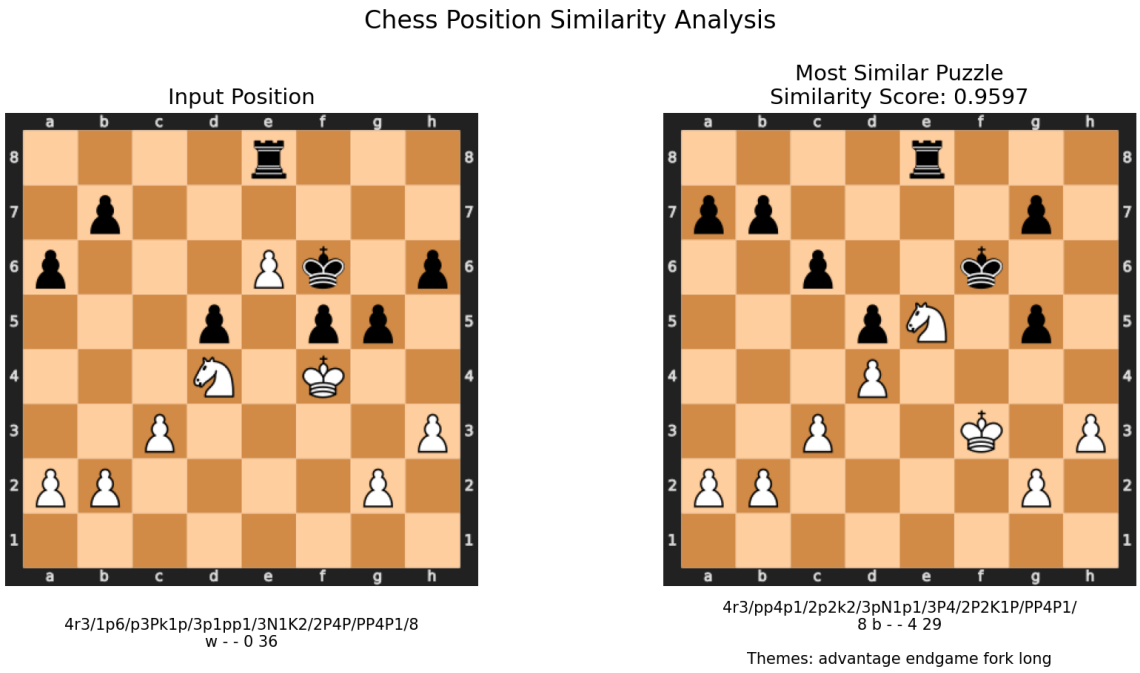


Figure 4: A position reached in a game and the most similar chess puzzle found in our database.

- **Send Response:** The JSON response containing the puzzle details is sent back to the frontend.

The user's browser receives the puzzle data and handles the interactive solving process:

- **Initialise Board:** A visual chessboard is displayed, configured with the starting FEN of the received puzzle.
- **Set Orientation:** The board is oriented based on the player's colour. Since Lichess puzzles typically start with the computer's move, the player's colour is determined as the opposite of the side to move in the initial FEN. The board is flipped accordingly.
- **Computer's First Move:** The application automatically makes the first move from the received solution sequence on the board, simulating the start of a Lichess-style puzzle.
- **User Interaction:** The user can now interact with the board by dragging and dropping pieces to make their move.

3.3 Results

Initial usage indicates that the puzzles suggested by the application are reasonable and intuitively feel like the reference position. As was done in section 1.5, we'll briefly analyse a suggestion returned from a given position.

Figure 4 shows a position reached in a game with a similar puzzle position found in our database.

Both positions show remarkable similarities, with Black having a two-point advantage with identical material on the board for each side. Black has good central control in each position, with White contesting control with their central knight. The pawn structure is also similar between each of the positions, albeit not identical. In each position, Black has the advantage, with the query position having an engine score of -3.75 and the selected puzzle -3.56.

Running and using the puzzle recommendation application gives very reasonable puzzle suggestions based on the position provided by the user, particularly in opening and endgame positions.

4 Conclusions

This project successfully developed `ChessLM`, a Transformer-based model designed to learn contextual vector representations (embeddings) for chess positions, moving beyond traditional evaluation-focused approaches. Inspired by the advancements in Natural Language Processing with models like BERT, our approach leverages self-supervised learning on large datasets of chess games to capture the intricate strategic and tactical themes inherent in different board states. By adapting the Vision Transformer architecture and employing two novel self-supervised tasks — Masked Piece Prediction and Moves Difference Prediction — the model learned to understand the spatial relationships between pieces, typical configurations, and the dynamic evolution of positions.

Qualitative analysis of the generated embeddings demonstrated the model's strength in identifying positions that share general structural and thematic similarities. The analysis across opening, middlegame, and endgame examples consistently showed that the model groups positions based on shared features such as material imbalances, recurring pawn structures, the presence or threat of passed pawns, king safety patterns, and the fianchetto of bishops. This ability to capture high-level positional motifs suggests that the model has learned a meaningful representation of the general 'look and feel' of a chess position, which is valuable for tasks requiring the retrieval of related positions.

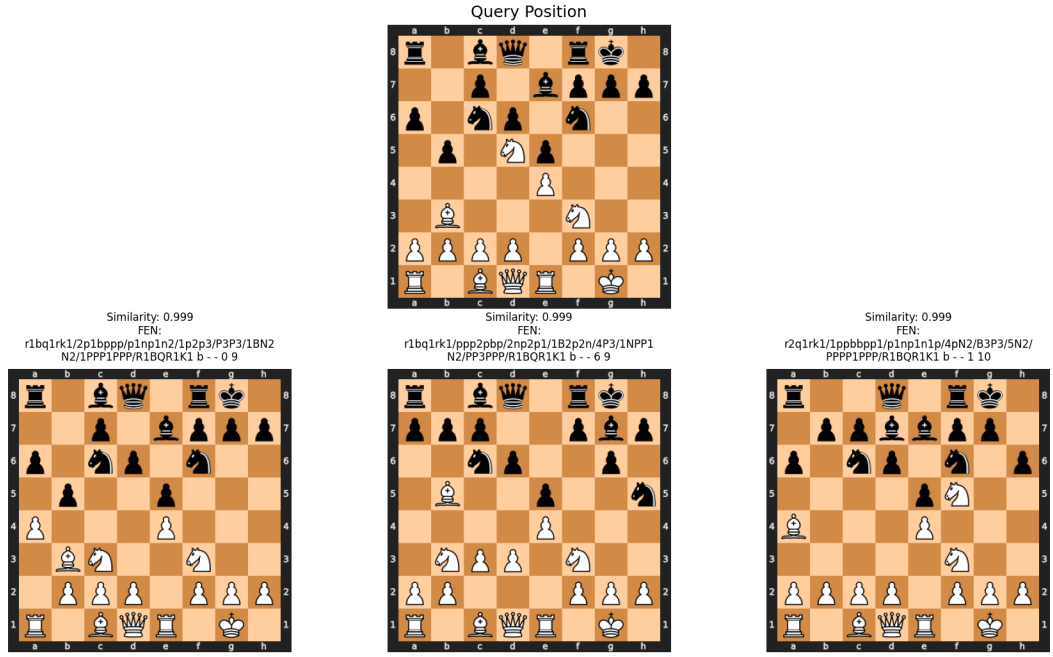
However, the analysis also exposed a key weakness: while the model identifies positions with shared themes, its similarity scores do not fully capture the subtle nuances that dramatically alter the evaluation and outcome of a position. Discrepancies in engine evaluations between seemingly similar positions highlight that factors like the precise activity and placement of kings, the exact threat level of passed pawns, and specific material compositions (e.g., a piece vs. multiple pawns) are not adequately weighted by the model's current learned representation.

The learned chess position embeddings open several avenues for potential applications. As demonstrated, a primary application is the retrieval of similar positions, which can be leveraged to suggest chess puzzles based on a user's provided position. Beyond this, these embeddings could enhance chess engines or analysis tools through Retrieval-Augmented Generation (RAG), providing context from historical games with similar positions. They could also facilitate downstream tasks such as classifying tactical motifs, or even as a rich input representation for models attempting to predict game outcomes or evaluate move quality.

Future research could focus on refining the self-supervised tasks or incorporating additional objectives to encourage the model to learn representations that are more sensitive to the nuances of positional evaluation. Exploring alternative model architectures or incorporating explicit knowledge about chess rules and piece movements could also lead to embeddings that better capture the strategic depth of the game. Furthermore, quantitative evaluation benchmarks for chess position understanding, beyond simple similarity retrieval or move prediction, would be crucial for rigorously assessing and comparing different representation learning approaches in chess.

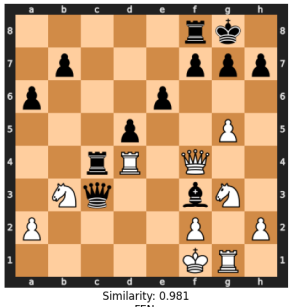
Appendix A. Similar Positions

This Appendix lists other similar positions identified by our model that we analysed during this research.

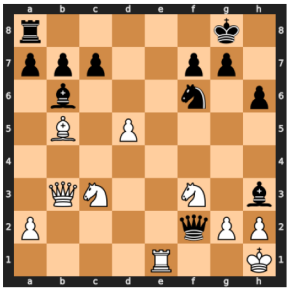


Validation Similarity Analysis for: 5rk1/1p3ppp/p3p3/3p2P1/2rR1Q2/1Nq2bN1/P4P1P/5KR1 w - - 6 27

Query Position

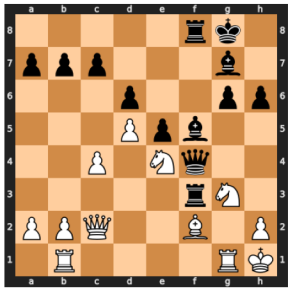


Similarity: 0.983
FEN:
r5k1/ppp2pp1/1b3n1p/1B1P4/8/1QN2N1b/P4qP
P/4R2K w - - 2 22



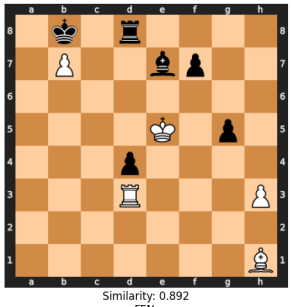
Similarity: 0.981
FEN:
r5k1/1b2rppp/p7/3pN3/Q1p2B2/1P4Pn/P5BP/2
R3qK w - - 4 23

Similarity: 0.980
FEN:
5rk1/ppp3b1/3p2pp/3Ppb2/2P1Nq2/5rN1/PPQ2
B1P/1R4RK w - - 4 25

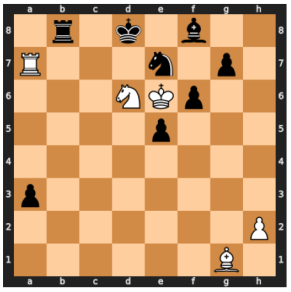


Validation Similarity Analysis for: 1k1r4/1P2bp2/8/4K1p1/3p4/3R3P/8/7B b - - 26 53

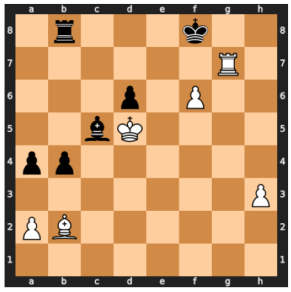
Query Position



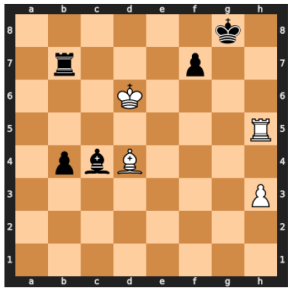
Similarity: 0.906
FEN:
1r1k1b2/R3n1p1/3NKp2/4p3/8/p7/7P/6B1 b -
- 8 48



Similarity: 0.892
FEN:
1r3k2/6R1/3p1P2/2bK4/pp6/7P/PB6/8 b - -
0 45

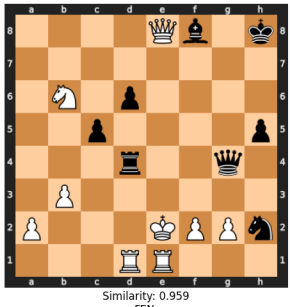


Similarity: 0.888
FEN:
6k1/1r3p2/3K4/7R/1pbB4/7P/8/8 b - - 1 47

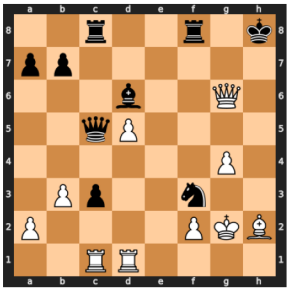


Validation Similarity Analysis for: 4Qb1k/8/1N1p4/2p4p/3r2q1/1P6/P3KPPn/3RR3 w - - 6 41

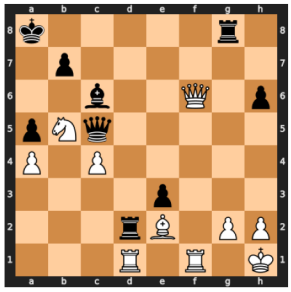
Query Position



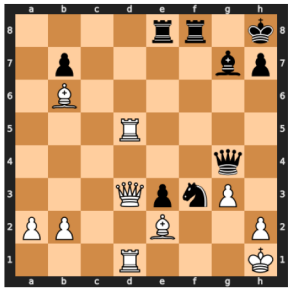
Similarity: 0.967
FEN:
2r2r1k/pp6/3b2Q1/2qP4/6P1/1Pp2n2/P4PKB/2RR4 w - - 2 31



Similarity: 0.959
FEN:
k5r1/1p6/2b2Q1p/pNq5/P1P5/4p3/3rB1PP/3R1R1K w - - 2 29

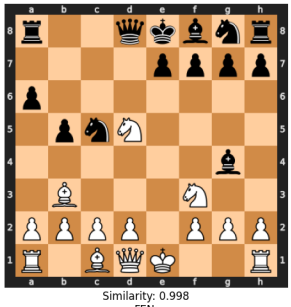


Similarity: 0.957
FEN:
4rr1k/1p4bp/1B6/3R4/6q1/3QpnP1/PP2B2P/3R3K w - - 4 33



Validation Similarity Analysis for: r2qkbnr/4pppp/p7/1pnN4/6b1/1B3N2/PPPP1PPP/R1BQK2R w KQkq - 2 9

Query Position



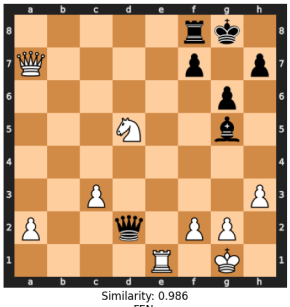
Similarity: 0.998
FEN:
r2qkb1r/p4ppp/2p2n2/nB2p1N1/6b1/3P4/PPP2PPP/RNBQK2R w KQkq - 0 9

Similarity: 0.998
FEN:
r2qkb1r/pb3ppp/2p2n2/n3N3/B2Pp3/8/PPP2PPP/RNBQK2R w KQkq - 2 10

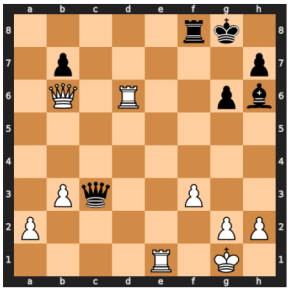
Similarity: 0.998
FEN:
m1qkb1r/pp3ppp/2p5/3p1b2/3Pn3/2NB1N2/PP2PPP/R1BQK2R w KQkq - 2 8

Validation Similarity Analysis for: 5rk1/Q4p1p/6p1/3N2b1/8/2P4P/P2q1PP1/4R1K1 w - - 1 31

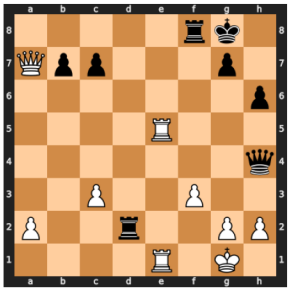
Query Position



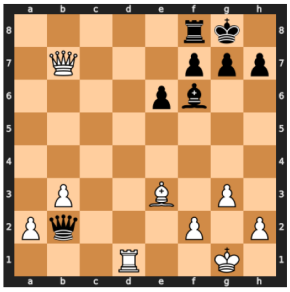
Similarity: 0.987
FEN:
5rk1/1p5p/1Q1R2pb/8/8/1Pq2P2/P5PP/4R1K1
w - - 2 35



Similarity: 0.986
FEN:
5rk1/Qpp3p1/7p/4R3/7q/2P2P2/P2r2PP/4R1K1
w - - 1 22



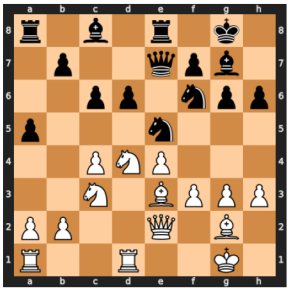
Similarity: 0.983
FEN:
5rk1/1Q3ppp/4pb2/8/8/1P2B1P1/Pq3P1P/3R2K
1 w - - 1 29



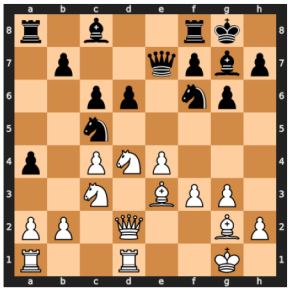
Validation Similarity Analysis for: r4rk1/1p1nqbbp/2pp1n2/p4pp1/P1PP4/1PN1PRP1/1B1QN1BP/R5K1 b - - 0 17

Query Position

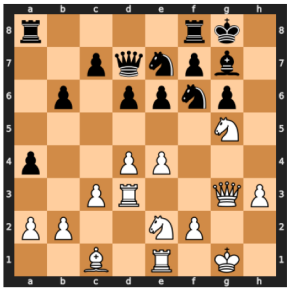
Similarity: 0.995
FEN:
r1b1r1k1/1p2qpb1/2pp1npp/p3n3/2PNP3/2N1B
PPP/PP2Q1B1/R2R2K1 b - - 0 15



Similarity: 0.992
FEN:
r1b2rk1/1p2qpbp/2pp1np1/2n5/p1PNP3/2N1BP
P1/PP1Q2BP/R2R2K1 b - - 2 14



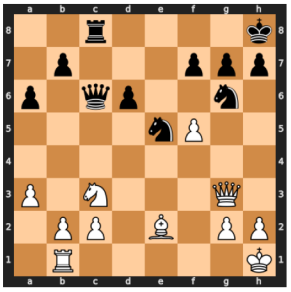
Similarity: 0.991
FEN:
r4rk1/2pqnpb1/1p1ppnp1/6N1/p2PP3/2PR2QP/
PP2NP2/2B1R1K1 b - - 1 19



Validation Similarity Analysis for: 3r3k/ppq3pp/2p2p2/8/5nN1/2P1Q2P/PP3PP1/4R2K b - - 2 26

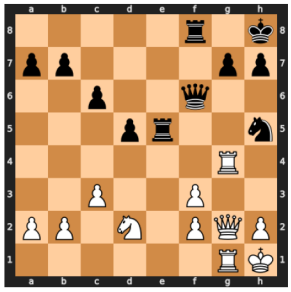
Query Position

Similarity: 0.986
FEN:
2r4k/1p3ppp/p1qp2n1/4nP2/8/P1N3Q1/1PP1B1PP/1R5K b - - 0 30



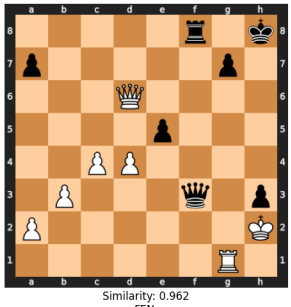
Similarity: 0.982
FEN:
7k/pp1q2pp/2p1rb2/2np4/6N1/2N2Q1P/PPP2PP1/3R1K2 b - - 1 24

Similarity: 0.978
FEN:
5r1k/pp4pp/2p2q2/3pr2n/6R1/2P2P2/PP1N1PQP/6RK b - - 9 28

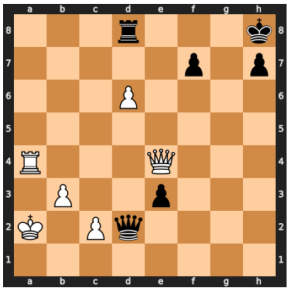


Validation Similarity Analysis for: 5r1k/p5p1/3Q4/4p3/2PP4/1P3q1p/P6K/6R1 b - - 1 40

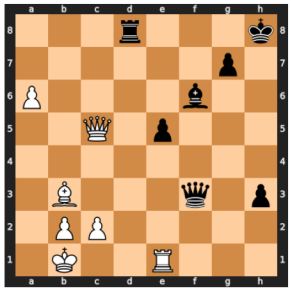
Query Position



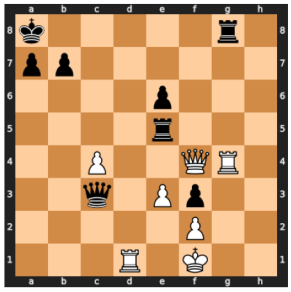
Similarity: 0.964
FEN:
3r3k/5p1p/3P4/8/R3Q3/1P2p3/K1Pq4/8 b - - 8 37



Similarity: 0.962
FEN:
3r3k/6p1/P4b2/2Q1p3/8/1B3q1p/1PP5/1K2R3 b - - 1 28

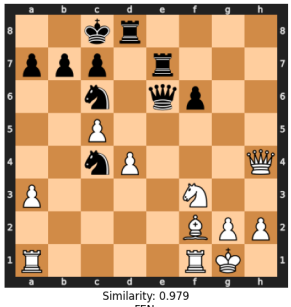


Similarity: 0.960
FEN:
k5r1/pp6/4p3/4r3/2P2QR1/2q1Pp2/5P2/3R1K2 b - - 0 37

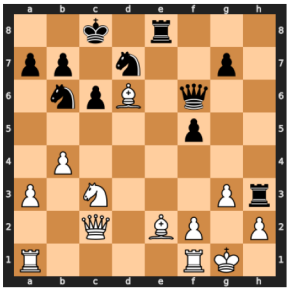


Validation Similarity Analysis for: 2kr4/ppp1r3/2n1qp2/2P5/2nP3Q/P4N2/5BPP/R4RK1 b - - 2 22

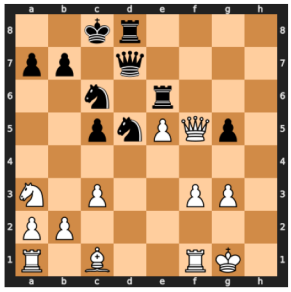
Query Position



Similarity: 0.979
FEN:
2k1r3/pp1n2p1/1npB1q2/5p2/1P6/P1N3Pr/2Q1BP1P/R4RK1 b - - 0 24



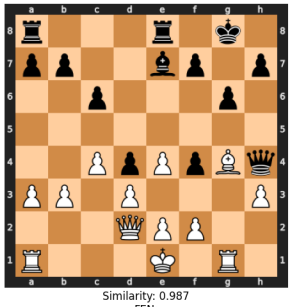
Similarity: 0.979
FEN:
2kr4/pp1q4/2n1r3/2pnPQp1/8/N1P2PP1/PP6/R1B2RK1 b - - 0 20



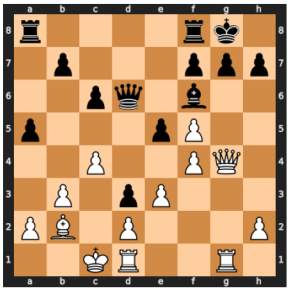
Similarity: 0.977
FEN:
1r2r1k1/pR1q2pp/2n1p3/3nP3/2pP4/B1P2QP1/P5BP/5RK1 b - - 0 20

Validation Similarity Analysis for: r3r1k1/pp2bp1p/2p3p1/8/2PpPpBq/PP1P3P/3QPP2/R3K1R1 w Q - 2 19

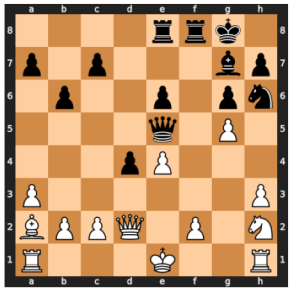
Query Position



Similarity: 0.987
FEN:
r4rk1/1p3ppp/2pq1b2/p3pP2/2P2PQ1/1P1pP3/PB1P3P/2KR2R1 b - - 0 18



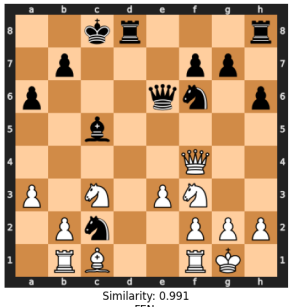
Similarity: 0.987
FEN:
4rrk1/p1p3bp/1p2p1pn/4q1P1/3pP3/P6P/BPPQ1P1N/R3K2R w KQ - 1 22



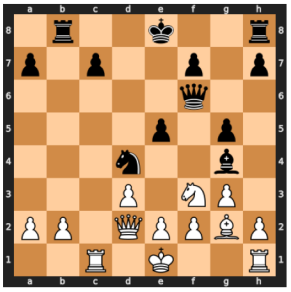
Similarity: 0.986
FEN:
3rk2r/pp1n1p1p/1qp5/6p1/2pPp3/P3P1P1/1P1QBPP1/1R2K2R w Kk - 0 17

Validation Similarity Analysis for: 2kr3r/1p3pp1/p3qn1p/2b5/5Q2/P1N1PN2/1Pn2PPP/1RB2RK1 w - - 0 16

Query Position

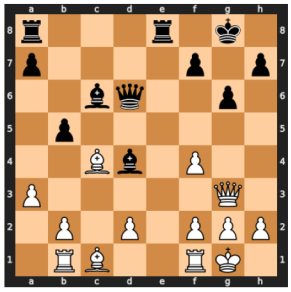


Similarity: 0.991
FEN:
1r2k2r/p1p2p1p/5q2/4p1p1/3n2b1/3P1NP1/PP1QPPBP/2R1K2R w Kk - 8 18



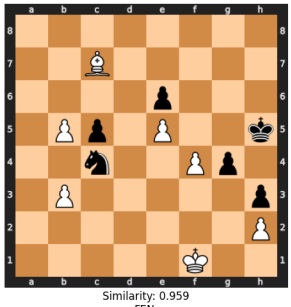
Similarity: 0.991
FEN:
2r1k2r/p1qn1ppp/4p3/6B1/3P2n1/2PQPN1P/5PPb/RNR3K1 w k - 1 17

Similarity: 0.991
FEN:
r3r1k1/p4p1p/2bq2p1/1p6/2Bb1P2/P5Q1/1P1P1PPP/1RB2RK1 w - - 0 20

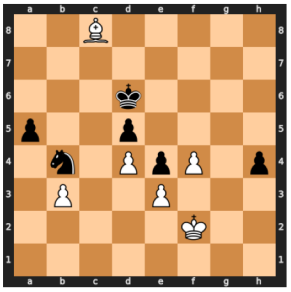


Validation Similarity Analysis for: 8/2B5/4p3/1Pp1P2k/2n2Pp1/1P5p/7P/5K2 b - - 0 53

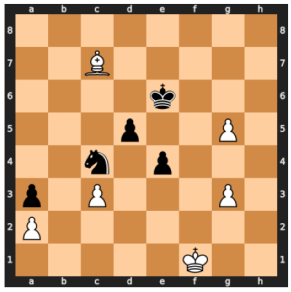
Query Position



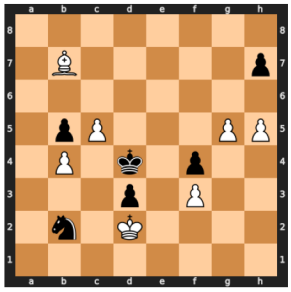
Similarity: 0.960
FEN:
2B5/8/3k4/p2p4/1n1PpP1p/1P2P3/5K2/8 w - - 6 48



Similarity: 0.959
FEN:
8/2B5/4k3/3p2P1/2n1p3/p1P3P1/P7/5K2 w - - 11 71

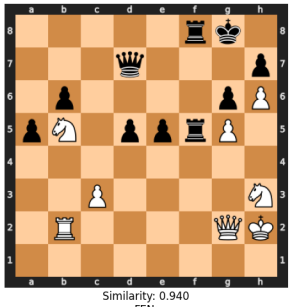


Similarity: 0.958
FEN:
8/1B5p/8/1pP3PP/1P1k1p2/3p1P2/1n1K4/8 w - - 2 46

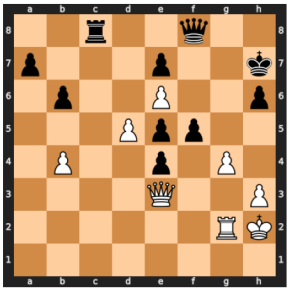


Validation Similarity Analysis for: 5rk1/3q3p/1p4pP/pN1pprP1/8/2P4N/1R4QK/8 w - - 6 49

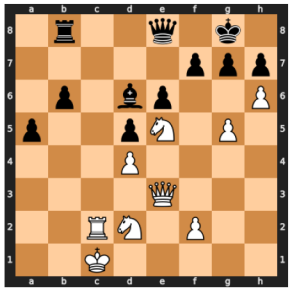
Query Position



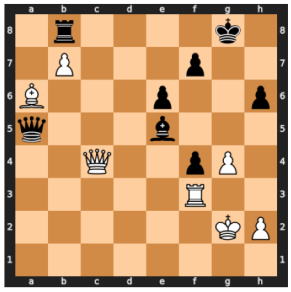
Similarity: 0.950
FEN:
2r2q2/p3p2k/1p2P2p/3Ppp2/1P2p1P1/4Q2P/6R
K/8 w - - 0 56



Similarity: 0.940
FEN:
1r2q1k1/5ppp/1p1bp2P/p2pN1P1/3P4/4Q3/2RN
1P2/2K5 w - - 1 30

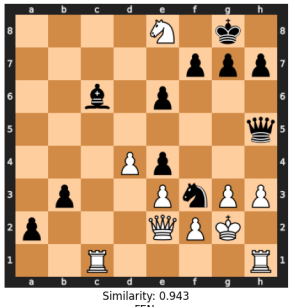


Similarity: 0.940
FEN:
1r4k1/1P3p2/B3p2p/q3b3/2Q2pP1/5R2/6KP/8
w - - 0 52

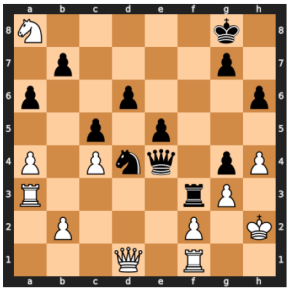


Validation Similarity Analysis for: 4N1k1/5ppp/2b1p3/7q/3Pp3/1p2PnPP/p3QPK1/2R4R w - - 0 31

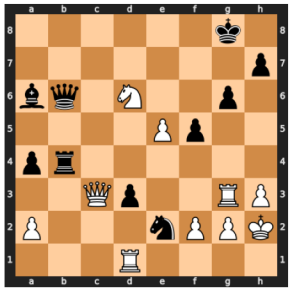
Query Position



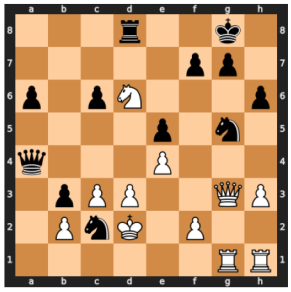
Similarity: 0.946
FEN:
N5k1/1p4p1/p2p3p/2p1p3/P1Pnq1pP/R4rP1/1P
3P1K/3Q1R2 w - - 4 23



Similarity: 0.943
FEN:
6k1/7p/bq1N2p1/4Pp2/pr6/2Qp2RP/P3nPPK/3R
4 w - - 1 37



Similarity: 0.943
FEN:
3r2k1/5pp1/p1pN3p/4p1n1/q3P3/1pPP2QP/1Pn
K1P2/6RR w - - 1 28

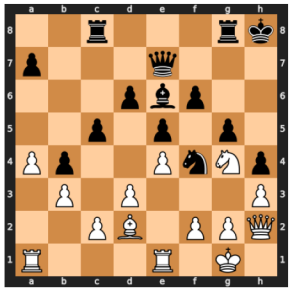


Validation Similarity Analysis for: r3r2k/4q3/bp1p1ppn/p1pPp2p/P1N1Pn2/2P3QP/1PBN1PP1/R3R1K1 b - - 3 25

Query Position

Similarity: 0.988
FEN:
r4r1k/3qbnpp/3p4/p1pPp2b/PpP2p2/1P1P1P2/
1BBQRNPP/R5K1 b - - 7 27

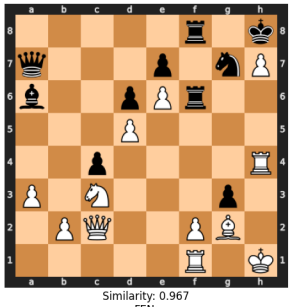
Similarity: 0.984
FEN:
2r3rk/p3q3/3pbp2/2p1p1p1/Pp2PnNp/1P1P3P/
2PB1PPQ/R3R1K1 b - - 1 28



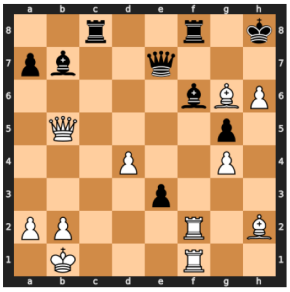
Similarity: 0.984
FEN:
r3q1rk/2n1p3/1pp1n2b/p2pP1pp/P2P1p2/BPP2
P1P/2N1Q1P1/1R1RN1K1 b - - 8 31

Validation Similarity Analysis for: 5r1k/q3p1nP/b2pPr2/3P4/2p4R/P1N3p1/1PQ2PB1/5R1K w - - 9 35

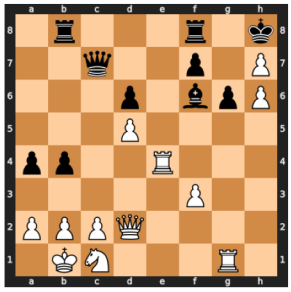
Query Position



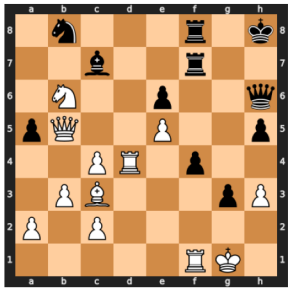
Similarity: 0.968
FEN:
2r2r1k/pb2q3/5bBP/1Q4p1/3P2P1/4p3/PP3R1B
/1K3R2 w - - 0 35



Similarity: 0.967
FEN:
1r3r1k/2q2p1P/3p1bpP/3P4/pp2R3/5P2/PPPQ4
/1KN3R1 w - - 1 30



Similarity: 0.966
FEN:
1n3r1k/2b2r2/1N2p2q/pQ2P2p/2PR1p2/1PB3pP
/P1P5/5RK1 w - - 0 36



References

[1] B. Kapicioglu, R. Iqbal, T. Koc, L. N. Andre, and K. S. Volz, "Chess2vec: Learning Vector Representations for Chess," 2020. [Online]. Available: https://arxiv.org/abs/2011.01014

[2] C. Pascutto, I. Bravi, A. Pollastro, L. Minto, and M. Wiering, "Learning to Evaluate Chess Positions with Deep Neural Networks and Limited Lookahead," in *Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2018)*, INSTICC. SciTePress, 2018, pp. 596–603. [Online]. Available: https://www.ai.rug.nl/~mwiering/GROUP/ARTICLES/ICPRAM_CHESS_DNN_2018.pdf

[3] Reddit Community, "How to input chess positions into a neural network," https://www.reddit.com/r/learnmachinelearning/comments/o8ti5f/how_to_input_chess_positions_into_a_neural_network/, June 2021, accessed: 2025-04-25.

[4] lifeofsquinting, "I trained an embeddings model for chess positions," https://www.reddit.com/r/LocalLLaMA/comments/1dc4udw/i_trained_an_embeddings_model_for_chess_positions/, May 2024, accessed: 2025-04-25.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[6] N. Carlini, N. Dacosta, E. Hou, A. L. Jones, M. Jagielski, M. Nasr, I. Paleka, A. Petrov, A. Schwarzschild, and S. Song, "Mastering Chess with a Transformer Model," 2024. [Online]. Available: https://arxiv.org/abs/2409.12272

[7] C. Nash, E. Hou, N. Carlini, A. L. Jones, N. Dacosta, S. Song, M. Jagielski, A. Petrov, I. Paleka, A. Schwarzschild, M. Nasr, J. Susskind, C. Paduraru, J. Kramár, Y. Chen, A. Ulrich, M. Reynolds, T. Pearce, I. Antonoglou, J. Schrittwieser, and D. Hassabis, "Amortized Planning with Large-Scale Transformers: A Case Study on Chess," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 37. Curran Associates, Inc., 2024, pp. 43 643–43 662. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/file/78f0db30c39c850de728c769f42fc903-Paper-Conference.pdf

[8] E. Boström and E. Nilsson, "Transforming Chess : Investigating Decoder-Only Architecture for Generating Realistic Game-Like Positions," Master's thesis, Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden, 2023, master's Thesis. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1828575/FULLTEXT01.pdf

[9] H. Jhamtani, V. Gangal, E. Hovy, G. Neubig, and T. Berg-Kirkpatrick, "Learning to Generate Move-by-Move Commentary for Chess Games from Large-Scale Social Forum Data," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018, pp. 1236–1246. [Online]. Available: https://aclanthology.org/P18-1115

[10] A. McInerney, M. Reid, D. Sejdinovic, and C. Clopath, "Detecting Individual Decision-Making Style: Exploring Behavioral Stylometry in Chess," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 17 475–17 486. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/95151403b0db4f75bfd8da5b61509558-Paper.pdf

[11] G. Sukthankar and R. Sukthankar, "Decoding Chess Mastery: A Mechanistic Analysis of a Chess Language Transformer Model," in *Artificial General Intelligence. AGI 2024. Lecture Notes in Computer Science*, vol. 14760. Springer, Cham, 2024. [Online]. Available: https://ial.eecs.ucf.edu/pdf/Sukthankar-AGI2024.pdf

[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, S. Tuckerman, O. Kaiser, A. Karpathy, B. Leibe *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[14] Lichess, "Lichess Database," https://database.lichess.org/, 2025.

[15] CCRL, "Computer Chess Ratings List (CCRL)," https://www.computerchess.org.uk/ccrl/, 2025.

[16] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.