Modelling Human Behaviour in Chess

Greg d'Eon (94920126) gregdeon@cs.ubc.ca
Shivam Thukral (94064177) tshivam2@cs.ubc.ca
Abner Turkieltaub (92286558) abner7@cs.ubc.ca

Abstract

While board games have long been a focus for AI research, the vast majority of this work has focused on creating superhuman AI. In this work, we instead develop a predictive model of human chess moves through supervised learning. The core idea behind this model is that it offloads the domain-specific knowledge about chess: it uses existing chess engines to quantify which moves are good or bad, and only needs to learn when humans will play a bad move. Our results show that our model is significantly better at fitting the distribution of human moves than existing chess engines, and we demonstrate qualitative reasons for these improvements.

1 Introduction

Board games provide environments with precise rules and well-defined state spaces where the optimal decisions are still not obvious. This structure makes board games an excellent testbed for psychologists and behavioural economists, who are interested in studying human decision making. Chase and Simon's experiments on grandmasters' perception of chess positions [1] are a classic example, using chess to gain insights about experts' decision-making processes. However, these types of studies have typically been limited to laboratory experiments, where small sample sizes limit their generalizability. This issue persists even with the advent of online chess servers: despite the huge number of games recorded online, psychology researchers have only studied this data in a limited capacity [2, 3].

Board games' clean structure and large datasets are also attractive to AI researchers, but state-of-the-art AI systems have rarely focused on predicting human play. Instead, the best AI in checkers, chess, Go, and poker are hailed as successes for being superhuman, beating top professionals in each of these games. When these systems are designed to predict human moves, they are typically trained on high-level play as a first step to bootstrap a longer deep learning pipeline. AI research is only now turning its focus to predict human play in games such as Hanabi [4] or Diplomacy [5], where elements of teamwork and cooperation make it necessary to understand other players' strategies.

In this project, we develop a predictive model that aims to play chess like a human. In other words, instead of treating chess as an optimization problem, we treat it as a supervised learning problem, using past data about people's actions to predict their future moves. We take a novel approach to this problem: our model relies heavily on existing chess engines to evaluate possible moves, offloading the domain knowledge about which moves are "good" or "bad". In a sense, our model does not need to learn how to play chess: it only needs to predict when humans will play these good or bad moves.

Contributions. We make two main contributions in this work. First, we introduce the core idea of developing predictive models by leveraging existing agents or bots. Second, we demonstrate this idea by creating a model that predicts human play with help from the Stockfish chess engine. In particular, our results show that transformer architectures, and their ability to handle variable-length inputs, are well-suited to this problem.

1.1 Related Work

Our work is situated between two broad research areas: machine learning techniques to play chess, and psychological research that uses chess to study human perception and decision-making processes.

AI research on board games, such as Go, chess, and Hanabi, has long focused on combining sophisticated search techniques, domain-specific adaptations, and hand-crafted features. However, this focus has shifted from using pre-existing knowledge of the game to developing deep learning techniques that need no such knowledge. For example, AlphaZero [6], DeepMind's first chess AI, was trained solely through self-play with no access to human moves, opening books, or endgame tables. With only 9 hours of training, AlphaZero far exceeded human performance, defeating Stockfish 8 in a 100-game tournament (28 wins, 0 losses and 72 draws). MuZero [7] is a model based RL approach that extends on AlphaZero: it uses the same search-based policy, but incorporates a learned model of game rules in the training phase. This approach does not require any knowledge of the game rules or environment dynamics, giving a way to apply such architectures to real-world problems. The key theme of this line of work is that it does not focus on predicting human play: instead, it attempts to achieve near-perfect play with as little domain knowledge as possible.

On the other hand, chess has been a canonical setting for studying human perception and decision making over the past 50 years. Chase and Simon's study [1], showing how chess experts perceive the board in "chunks", is a classic example, and more recent work has validated these ideas using eye trackers [8]. It is also well-known that expert players are more efficient at finding strong moves: De Groot [9] and Camptitelli and Gobet [10] showed that experts do not necessarily analyze a large number of variations, but focus more of their time on analyzing good moves. These differences are so consistent across players that it is possible to evaluate chess proficiency through standardized tests [11].

However, little of this work has leveraged the wealth of data available from online chess servers. We are only aware of two papers that analyze individual online games: Sigman et al. [2] studied the amount of time that players spent on their moves, and Slezak and Sigman's analysis [3] suggests that people play slower and more accurately when they are outranked by their opponent. Thus, we believe that there is significant value in using supervised models to work with this relatively new source of data.

2 Chess Data and Engines

We begin by describing Lichess, the source for our dataset, and Stockfish, a strong chess engine that we used as our source of domain-specific knowledge.

2.1 The Lichess Database

Lichess is a free online chess server. It is continually growing in popularity, attracting hundreds of thousands of players, ranging from brand-new beginners to world champions. Within the server, players play a wide variety of chess games. At one extreme, long "classical" games give each side at least 30 minutes of thinking time; at the other, "bullet" games only give each player 1 minute of time, resulting in frantic time scrambles. Players' skill levels are tracked with an ELO rating that increases with victories and decreases with losses.

Of particular interest to our work, Lichess publishes monthly releases of their database of games. This database is enormous and still growing: as of April 2020, it contains 1.1 billion games. Each of these games are recorded in great detail, including information about the players' ELO ratings and the game clock after each move. In this work, we hoped to demonstrate our core idea on a small subset of these games; however, with such a large data source, it would be straightforward to collect more games for models that require a large amount of training data.

2.2 Stockfish

Stockfish is a top, free, open-source chess engine. It has two main components. First, it has a carefully-tuned valuation function that condenses expert domain knowledge to compute a score for every position. Second, it combines this valuation function with an efficient tree search algorithm, using positions' scores to help prune the search. Together, these features make it one of the strongest engines available.

While Stockfish's main goal is to play at a high level, it is also possible to configure the engine to play at a weaker skill level. To do this, its strength is reduced by adding a small probability to play weaker moves. However, even at weaker skill levels, Stockfish is far from a generative model of human play: in some cases, it makes obvious mistakes that a human would almost never play, while in others, it manages to play careful, precise moves in complex positions.

In this work, we leverage Stockfish in two ways. First, we use Stockfish to analyze positions in our dataset. These analysis results serve as the primary source of domain knowledge in our models. We also use Stockfish's predictions to create a simple baseline to compare our models against.

3 Method

Our dataset provides a unique challenge: each position has a different set of legal moves. Thus, predicting human moves is a multi-class classification problem, but with a different set of classes for each position. We dealt with this problem by creating separate features for each legal move and building models that accept a variable-length list of inputs. In this section, we describe our features and the models that work with them.

3.1 Features

For each legal move in a position, we compute three types of features: *move* features, which describe unique properties about each move, including Stockfish's analysis of each move; *board* features, which describe the current state of the game; and *evaluation* features, which contain Stockfish's static evaluation of the position. We give some examples of these features here; for completeness, the full list is included in Appendix A.

First, *move* features give details about one of the legal moves. Examples of move features include the type of piece being moved, how many squares it is moving, and whether it will capture an opposing piece.

The most important move feature is the result of Stockfish's analysis. For each legal move, we use Stockfish to estimate which player would have an advantage after playing this move. This advantage is measured in "centipawns" – the equivalent advantage in material – or the number of moves until a checkmate can occur, if Stockfish has found a way to reach checkmate. We gave Stockfish a relatively small reading depth of 10 ply to balance computation time and the accuracy of this analysis: though this is not as deep at the 18 ply depth used in Lichess's analysis tools, prior work suggests that Stockfish is still stronger than most amateurs with this depth [12].

Second, *board* features describe the current state of the game. Examples of board features include the time left on the player's clock and the number of pieces under attack. They also include high-level details about the game, such as players' ELO ratings. Board features are common across all legal moves.

Finally, *evaluation* features are the result of Stockfish's static evaluation of the position. These features give a breakdown of the advantages that each player has, such as differences in material, space, and king safety. We included these with the hope that they could help with identifying domain-specific strengths and weaknesses of players. For instance, a model could learn that players are more cautious than necessary when their king appears to be unsafe.

3.2 Models

As is typical of multi-class classification problems, the goal of our models is to output a probability distribution over the legal moves. However, this modelling was complicated by the variable number of legal moves. We designed 3 model architectures that deal with this issue: LINEAR and NEURAL NET models that consider each move independently, and a TRANSFORMER model that is able to make comparisons between each of the legal moves.

The LINEAR model applies a single linear transformation to the inputs to produce a logit value for each move. The output probability distribution is produced by taking a softmax of these logits across the set of legal moves. Using the same linear layer for every move is a form of parameter tieing, allowing the model to generalize to any number of legal moves, but forcing it to consider each move in isolation. We trained a linear model on all of the move features. Note that board and evaluation features are shared across all legal moves, so a purely linear model would have no use for them.

The NEURAL NET model extends upon the LINEAR model by replacing the linear transformation with a feedforward neural network. This extension allows the model to learn more complicated outputs, but still no relationships between moves. In particular, we used neural networks with ReLU activations, 2 hidden layers consisting of 32 and 16 hidden nodes, and 10% dropout to avoid overfitting. We trained four neural network models: one using the move features excluding Stockfish analysis results ("No SF"), one using move features alone ("Moves"), one using move and board features ("Board"), and one using all features ("All").

Finally, the TRANSFORMER model uses a transformer encoder layer [13] to consider relationships between the legal moves. This type of layer is typically used in natural language processing models for its ability to make comparisons between all of the words in sentences of varying lengths. We adapt it for our dataset to handle a variable-length list of moves. Specifically, the model applies a transformer encoder layer with a single attention head to the move features, a fully-connected layer with ReLU activation to the remaining features, and a final linear layer to map each legal move to a logit value. As with the NEURAL NET models, we trained four TRANSFORMER models, using the same feature sets as the "No SF", "Moves", "Board", and "All" models.

We are not aware of any existing models for this task, so we also developed a baseline STOCKFISH model to compare against. This baseline is inspired by Stockfish's method for lowering its strength. It takes a softmax over Stockfish's analysis scores, corresponding to playing the best move most often, but adding a small probability of playing worse moves. A single parameter adjusts the scale of the scores before the softmax, interpolating between uniform random predictions and an ordinary "hard" max.

3.3 Experiments and Evaluation

To create a dataset for supervised learning, we took a small sample of 1000 games from the Lichess database. From these, we filtered out games with very fast time controls, keeping games with at least 3 minutes of thinking time. We also filtered out games with players below 1300 ELO or above 1800 ELO, keeping the middle 50% of ELO ratings. In each of the remaining games, we included every position in the dataset; the bottleneck of the data generation process was analyzing each of these positions with Stockfish. This process produced 24975 positions, which we randomly split into a training set (60%), validation set (20%), and test set (20%).

We trained each model using the AdamW optimizer [14] to minimize the cross entropy loss, keeping the model with the lowest validation set loss. Then, we compared the models' performance on the test set with four metrics: the negative log-likelihood (NLL), and the top-1, top-3, and top-5 accuracy. We used all three accuracy metrics due to the high amount of variation in players' moves, which we discuss in the following section.

4 Results

Next, we show our experimental results, show some of the qualitative strengths and weaknesses of our models, and describe the extent to which future models can improve on our results.

4.1 Test Performance

Our main results are shown in Figure 1. Compared to the STOCKFISH baseline, we found that the two models without Stockfish's analysis features achieved better NLL, but had significant losses in accuracy. The three NEURAL NET models all achieved far lower NLL with comparable accuracy to STOCKFISH, but with only minor differences between the three feature sets. Finally, the last three TRANSFORMER models had the lowest NLLs and highest accuracies, but with virtually no differences between the three models. Thus, while even simple models can fit the distribution of human play better than our baselines, it is clear that the transformer models' ability to make comparisons between moves is a great advantage.

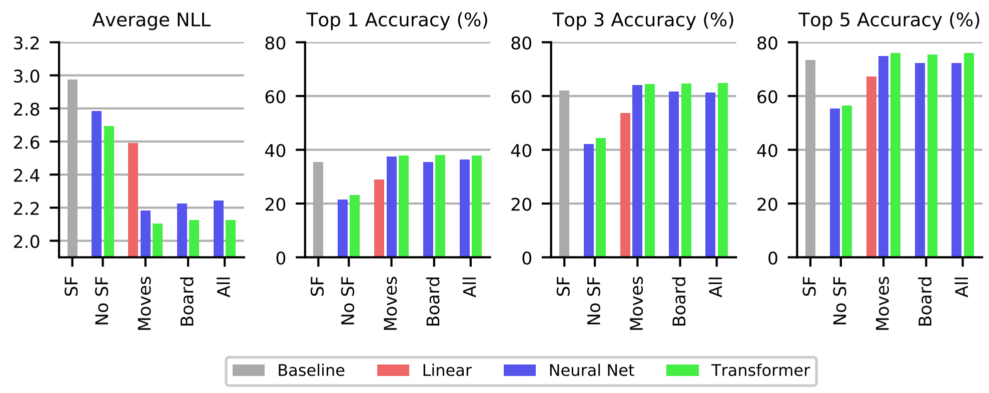

Figure 1: Results of evaluation on the test set.

There are two surprising findings in these results. One is that the STOCKFISH baseline has quite high accuracy, but poor NLL. This combination indicates that humans often play good moves, but sometimes make mistakes that Stockfish would almost never play. In other words, Stockfish does a reasonable job of fitting the high-probability events, but grossly underestimates the low-probability events.

The second surprising finding is the consistency of the NEURAL NET and TRANSFORMER models: the models with access to the *board* and *evaluation* features do no better than the models with only *move* features. We expected that giving the models information about players' skill levels and time pressure would help to make more accurate predictions. To reach even better performance, we suspect that we would need larger models with more expressiveness, more data, and more training time.

4.2 Qualitative Analysis

While it is clear that our models outperformed the baseline on the test set, we were curious if there are specific types of positions where our models excel. It is difficult to quantify patterns in these positions, but we took a first step by considering our best model – the TRANSFORMER using move features – and searching for positions where this model makes particularly good or bad predictions.

Specifically, we first found positions where our model does well: it correctly predicts the human move, while Stockfish's prediction puts this move outside of the top 5 choices. One example is shown in Figure 2. In these positions, we found that Stockfish tended to prefer slow, developing moves, while human players prefer to capture a piece or initiate an exchange. Our model recognizes this human quality, predicting that these captures are more likely when there is tension on the board.

We also found positions where our model made poor predictions, assigning a very low probability (below 10^{-3}) to the human move. These moves tended to be serious mistakes: for example, in the position in Figure 3, the human move immediately loses the player's queen. Thus, there are still positions where human player make blunders so serious that our model had ruled the move out. We suppose that mistakes of this magnitude are relatively rare, and significantly more training data would be needed to give more examples of the types of oversights that humans tend to make.

4.3 Theoretical Performance Limits

Something worth noticing is that this problem is not a classical prediction problem where there is just one correct answer. When faced with the same position multiple times, players might – and often do – play a variety of different moves. This variation puts an upper bound on our models' performance: the selected move is a random variable drawn from an unknown probability distribution, so we cannot hope to reach 100% accuracy. How well could our models possibly perform?

It is infeasible to find the true distribution of selected moves for every position. To get a sense of the performance bound, we investigated the initial position, where Lichess's database summarizes 140 million moves from players with at least 1600 ELO. In this position, there are several popular opening moves. The most common moves are e4 and d4, with 74 million and 39 million plays, respectively;

Nf6 and c4 are less common, and there is a long tail of rarer moves. Even if a model had access to this exact distribution of moves, it could only achieve top-1, top-3, and top-5 accuracies of 53.0%, 86.9%, and 93.4%, respectively. Further, the optimal average NLL would be achieved by reporting this distribution, giving an NLL of 1.37. We believe that many other positions later in the game likely have a wider distribution of plausible moves, making this performance bound even worse.

We conclude that predictive models for this problem cannot and should not reach perfect accuracy. Instead, their goal is to match the underlying distribution of human moves as closely as possible. In future work, it may be helpful to quantify the spread of this distribution in a wider range of positions to help understand how close a model is to achieve the optimal performance.

5 Discussion

Our experimental results showed that our models are significantly better at predicting human chess moves than existing baselines. We found that a feedforward neural network architecture, considering each legal move independently, achieves a similar accuracy to Stockfish's predictions with significantly better NLL. Further, we also found that using a transformer-based architecture with the ability to make comparisons between each of the legal moves led to significantly better performance in both metrics. In both cases, this performance was largely driven by having access to Stockfish's analysis outputs, showing the importance of providing domain-specific knowledge to these models.

However, it is also clear from the results that there is room for improvement. Despite training a variety of models with access to additional features about the positions, we found that the best model only used features pertaining to the legal moves. This finding surprised us: we expected that information about the player's skill level would help to make more accurate predictions. Thus, we are suspicious that our models had too little capacity, and we believe that more complex models with a larger dataset could perform significantly better.

In sum, our core technique of leveraging domain knowledge from existing AI systems made it possible for relatively simple models to make reasonably accurate predictions of human behaviour. While our findings are specific to chess, we believe that this idea could be applied to great effect on other problems in psychology and behavioural economics.

References

[1] William G. Chase and Herbert A. Simon. Perception in chess. *Cognitive psychology*, 4(1):55–81, 1973.

[2] Mariano Sigman, Pablo Etchemendy, Diego Fernandez Slezak, and Guillermo A Cecchi. Response time distributions in rapid chess: a large-scale decision making experiment. *Frontiers in neuroscience*, 4:60, 2010.

[3] Diego Fernandez Slezak and Mariano Sigman. Do not fear your opponent: Suboptimal changes of a prevention strategy when facing stronger opponents. *Journal of Experimental Psychology: General*, 141(3):527, 2012.

[4] Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280:103216, 2020.

[5] Philip Paquette, Yuchen Lu, Steon Steven Bocco, Max Smith, O-G Satya, Jonathan K Kummerfeld, Joelle Pineau, Satinder Singh, and Aaron C Courville. No-press diplomacy: Modeling multi-agent gameplay. In *Advances in Neural Information Processing Systems*, pages 4476–4487, 2019.

[6] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[7] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering

atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019.

[8] Neil Charness, Eyal M. Reingold, Marc Pomplun, and Dave M. Stampe. The perceptual aspect of skilled performance in chess: Evidence from eye movements. *Memory & cognition*, 29(8):1146–1152, 2001.

[9] Adriaan D. De Groot. *Thought and choice in chess*. The Hague: Mouton, 1946/1978.

[10] Guillermo Campitelli and Fernand Gobet. Adaptive expert decision making: Skilled chess players search more and deeper. *ICGA Journal*, 27(4):209–216, 2004.

[11] Han L. J. Van Der Maas and Eric-Jan Wagenmakers. A psychometric analysis of chess expertise. *The American journal of psychology*, pages 29–60, 2005.

[12] Diogo R Ferreira. The impact of the search depth on chess playing strength. *ICGA Journal*, 36(2):67–80, 2013.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[14] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Appendix A List of features for training

The feature list is divided into 3 categories: move features, board features, and evaluation features. Each category is summarized in the following section.

A.1 Move Features

- move-uci : Notation of move played
- move-from-x : Initial file
- move-from-y : Initial rank
- move-to-x : Ending file
- move-to-y : Ending rank
- move-dx : Signed move distance (horizontal)
- move-dy : Signed move distance (vertical)
- move-piece : One of P, N, B, R, Q, K, or lowercase for black; one-hot encoded into
  - move-piece-pawn
  - move-piece-knight
  - move-piece-bishop
  - move-piece-rook
  - move-piece-queen
  - move-piece-king
- move-is-capture : True if this move would capture a piece
- move-is-threatened : True if this piece is under attack
- move-is-defended : True if this piece has a defender of any piece type
- move-stockfish-eval : Stockfish evaluation of this move, in centipawns

A.2 Board Features

- game-time-control-initial : Initial seconds in game
- game-time-control-increment : Increment per move in game
- game-ELO self : ELO of player to move
- game-ELO opponent : ELO of opponent
- board-full-moves : Move number (where 1 move consists of a white move, then a black move)
- board-white-to-move : True if it's white's turn
- board-clock-self : Seconds left at the start of this move
- board-clock-opponent : Opponent's seconds left at the start of this move
- board-clock-used : Seconds that the player spent on this move
- board-check : True if the king is in check
- board-can-kingside-castle : True if the player has kingside castling rights
- board-can-queenside-castle : True if the player has queenside castling rights
- board-number-of-legal-moves : Number of possible moves that can be played
- board-attacked-pieces : Number of player's pieces that are under attack
- board-attacking-pieces : Number of player's pieces that are attacking an opposing piece
- board-undefended-pieces : Number of player's pieces that are hanging (no defender)
- board-material : Material balance score, normalized to [-1, 1]
- prev-move-is-capture : True if the previous move captured a piece
- prev-move-clock-used : Number of seconds the opponent used on the previous move
- prev-move-threat-on-undefended-piece : True if the previous move threatened a hanging piece

A.3 Evaluation Features

Stockfish's static evaluation breaks the score into a number of terms pertaining to different features of the board:

- Material: number of pieces on the board, weighted by their strength, with bonuses for pieces on useful squares
- Imbalance: material advantages that are not captured in raw material count
- Pawns: strength of pawn structure, with higher values for well-supported pawn chains
- Knights: bonuses for knights on well-defended "outpost" squares
- Bishops: bonuses for bishops on outposts or with long line of sight
- Rooks: bonuses for rooks coordinating with other pieces or with long line of sight
- Queens: penalties for being in vulnerable positions
- Mobility: bonuses for pieces with many available squares
- King safety: bonuses for king that is well defended by pieces or pawn structure
- Threats: bonuses for threatening or restricting valuable pieces
- Passed pawns: bonuses for pawns that are close to promoting
- Space: bonuses for having protected, open space to manoeuvre pieces
- Intiative: corrections for positions that are highly dynamic

Most of these terms are split into three parts: a "white" score, a "black" score, and a "total", which is the difference of the scores for the two colors. (Material, Imbalance, and Pawns are exceptions, only providing a "total" score.) Additionally, each of these terms is split into a "midgame" and an "endgame" score.

Appendix B Sample Model Outputs

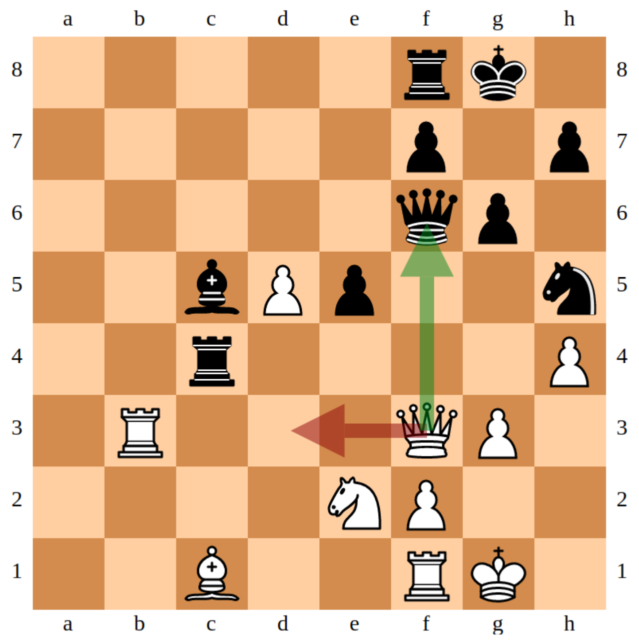

Figure 2: An example of an accurate prediction by our model. Our model predicts the human move Qxf6 (green arrow), which trades queens with the opponent. Stockfish predicts the alternative Qd3 (red arrow), which is a slower threat on black's pieces. The human move is not in Stockfish's top 5 suggestions.

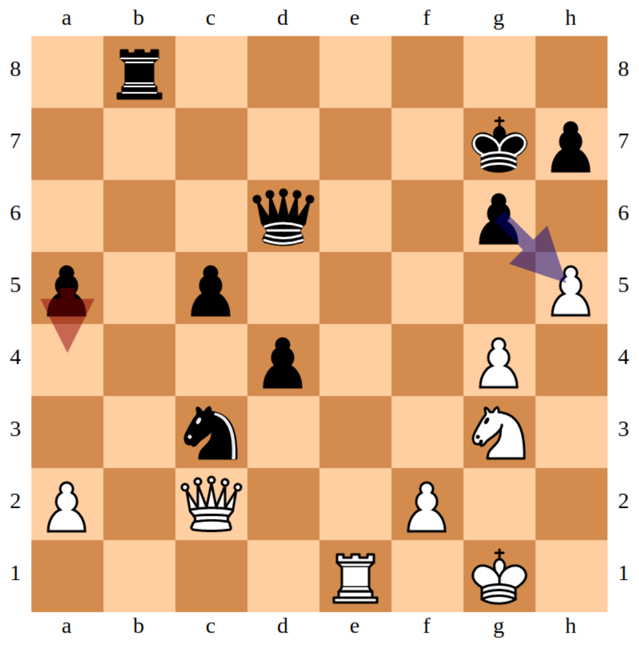

Figure 3: An example of an inaccurate prediction by our model. Our model incorrectly predicts the move a4 (red arrow), and suggests that the human move gxh5 (blue arrow) has probability less than 2×10^{-4}. The actual move is a serious blunder, losing black's queen in 2 moves.