

Chess Rating Estimation from Moves and Clock Times Using a CNN-LSTM

Michael Omori^[0009–0000–4632–9272], Prasad Tadepalli^[0000–0003–2736–3912]

Oregon State University, Corvallis OR, USA
 {omorim, prasad.tadepalli}@oregonstate.edu

Abstract. Current chess rating systems update ratings incrementally and may not always accurately reflect a player’s true strength at all times, especially for rapidly improving players or very rusty players. To overcome this, we explore a method to estimate player ratings directly from **game moves and clock times**. We compiled a benchmark dataset from Lichess with over one million games, encompassing various time controls and including move sequences and clock times. Our model **architecture comprises a CNN to learn positional features**, which are then integrated with clock-time data **into a Bidirectional LSTM**, predicting player ratings after each move. The model achieved an MAE of 182 rating points on the test data. Additionally, we applied our model to the 2024 IEEE Big Data Cup Chess Puzzle Difficulty Competition dataset, predicted puzzle ratings and achieved competitive results. This model is the first to use **no hand-crafted features to estimate chess ratings** and also the first to output a rating prediction after each move. Our method highlights the potential of using move-based rating estimation for enhancing rating systems and potentially other applications such as cheating detection.

Keywords: Chess · Rating Estimation · Cheating Detection.

1 Introduction

Traditional rating systems like Elo [6] and Glicko [7], though effective, often require many games to accurately reflect the true level of skill of a player, leading to potential mismatches and inaccuracies, especially for new accounts or rapidly improving players. This delay in accurate rating adjustment can skew competitive fairness and hinder effective matchmaking. For example, **lower-rated players were shown to have longer winning streaks than higher-rated players on average**, suggesting that their rating may be less accurate compared to higher-rated players [3]. This could occur if a **person creates a new account** and is either an experienced chess player who made a second private account or a new player who transferred skills from other board games. In both cases, their initial rating starts lower on sites such as lichess [16] and chess.com [2], but their performance could be much higher.

To overcome these limitations, we propose a novel method that estimates player ratings directly from **game moves and clock times**, aiming to provide accurate ratings after each game. Our dataset comes from Lichess, encompasses 1.2 million games from April 2021 to July 2024 (inclusive), covers various time controls and includes detailed move sequences and clock times. Chess streamers like **GothamChess** [23] have attempted this on their own (without computer aid) by using their chess knowledge to look at games and "**guess the elo**" and it is quite challenging to be consistently accurate.

Our method is a neural network architecture that combines a **four-layer Convolutional Neural Network (CNN)** and a **Bidirectional Long Short-Term Memory (LSTM)** network. The **CNN** extracts positional features from the chessboard, which are **then integrated with clock time data by the LSTM** to predict player ratings move-by-move. This approach allows the model to consider both spatial and temporal dimensions of the game, leading to more precise rating predictions. The model achieved a mean absolute error (MAE) of 183 rating points in the test data, demonstrating the effectiveness of the overall approach.

Ablation studies confirm that the **inclusion of clock time data improves the model performance**. Additionally, the same model is tested on the 2024 IEEE Big Data Cup Chess Puzzle Difficulty Competition dataset, where it predicts puzzle ratings competitively, indicating the model's versatility and robustness.

An additional application of the rating estimate for each move is to check for anomalous behavior. **For example, a 1300 rated player that is playing like a 2600 in half their games, but 1300 in the other half may be considered an anomaly.** A CNN was [20] trained on Lichess games but only achieved marginally better results than random guessing for **classifying cheaters**. **Lichess** itself has software called **Kaladin** for detecting cheaters [14]. It uses a CNN and also utilizes chess insights, which includes metrics such as average centipawn loss and number of moves. One downside to this approach is that it can only do as well as the people who labeled the data.

Our main contribution is creating the first neural network that predicts chess rating move by move without hand-crafted features. This is unlike the rating players normally get after playing an entire game, which is based off their current rating, the game result, and the opponent's rating. Second, we collated a benchmark from Lichess data for evaluation and finally demonstrated the utility of the amount of clock time remaining after each move in estimating player rating.

In conclusion, this method presents an advancement in chess rating estimation by leveraging deep learning techniques to analyze in-game moves and clock times. The promising results and potential applications highlight the importance of this method, paving the way for more dynamic and fair rating systems in competitive chess.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

2 Background

There are various rating systems to estimate a player's skill, such as FIDE's Elo and Lichess's Glicko 2. Elo uses expected scores and a K-factor for rating updates, while Glicko incorporates rating deviation (RD) and volatility for more accurate adjustments. We will briefly explain Glicko2 since that is the rating system that we are trying to estimate through moves.

2.1 Glicko-2 Rating System

Glicko-2 incorporates volatility to indicate how consistent a player plays, along with a rating and a rating deviation for each player. A collection of games like in a tournament is used to update these values and is treated to have occurred simultaneously with the same ratings μ , volatility, and deviation ϕ for each game. The rating deviation is manually set for each player to a specified amount such as between 0.3 and 1.2.

First, the variance v is calculated based on the outcomes of the game for each opponent with the rating μ_j .

$$v = \left[\sum_{j=1}^m g(\phi_j)^2 E(\mu, \mu_j, \phi_j) \{1 - E(\mu, \mu_j, \phi_j)\} \right]^{-1} \quad (1)$$

where

$$g(\phi) = \frac{1}{\sqrt{1 + 3\phi^2/\pi^2}} \quad (2)$$

$$E(\mu, \mu_j, \phi_j) = \frac{1}{1 + \exp(-g(\phi_j)(\mu - \mu_j))} \quad (3)$$

The change in rating Δ is then calculated by comparing the actual outcome of the game (win 1, draw 0.5, loss 0) with the expected result based on the logistic formula in equation 3. Note that $g(\phi_j)$ and v are directly proportional to the step size of the rating change.

$$\Delta = v \sum_{j=1}^m g(\phi_j) (s_j - E(\mu, \mu_j, \phi_j)) \quad (4)$$

The new volatility is calculated using the Illinois algorithm [27], based on Δ , ϕ , and v . A more detailed explanation can be found in the paper by Mark Glickman [7]. The final step involves updating both the rating deviation and the player ratings to μ' and ϕ' .

$$\phi^* = \sqrt{\phi^2 + \sigma'^2} \quad (5)$$

$$\phi' = \frac{1}{\sqrt{\frac{1}{\phi^{*2}} + \frac{1}{v}}} \quad (6)$$

$$\mu' = \mu + \phi'^2 \sum_{j=1}^m g(\phi_j) (s_j - E(\mu, \mu_j, \phi_j)) \quad (7)$$

Overall, the Glicko-2 rating update is applied after each game, while our work tackles the challenge of rating updates after each move within a game.

3 Related Works

One of the first works on modeling chess player skill used Rybka 3 [31] evaluations to model player moves based off rating [22]. Later, 30 hand-crafted features were fed into random forests [1] and support vector machines [4] to predict ratings on players binned by rating into 10 different splits [28]. Rather than focusing the problem as a classification problem with limited bins, we choose to frame it as a regression problem, as this gives us more flexibility and precision in rating players. Also, the data from the previous rating estimation paper is not publicly available, so we compiled our own open source dataset from Lichess for this task.

The recent popularity of word prediction has permeated its way into the chess field as move prediction. Maia [17] adapted AlphaZero’s residual Convolutional Neural Network (CNN) [26], removed the search algorithm to predict moves based on rating bins, and improved the accuracy over a depth-limited Stockfish. Stockfish is currently the strongest chess engine [29] and the original Leela Zero was an open source clone of it, although it has since migrated to a transformer architecture [19]. The Maia model was then fine-tuned to predict specific player moves [18]. GPT-2 was fine-tuned on 2.8 million chess games, demonstrating strategic understanding without directly obtaining a rating. Another transformer model achieved a 2895 Lichess Blitz rating through supervised learning on Stockfish evaluations from 10 million games [24].

Further classification of chess games has been explored through a binary classifier for detecting cheaters, using the Euclidean distance to compare player features such as move accuracy and response time [12]. One paper showed that the winning likelihood strongly correlates with the remaining time in rapid chess games, essentially less time lowers chances of winning [25]. Multi-class classification of chess games, predicting outcomes such as win/draw/loss based on moves and ratings with an LSTM, has also been attempted [21]. Another study predicted the brilliance of moves (a binary classification) using features from chess engine analysis [33]. While these approaches are similar to ours, they focus on classification tasks, whereas our model predicts a continuous value.

Rating prediction has been conducted in other gaming domains, such as Massive Online Battle Arena (MOBA) games. In one case, a transformer was trained to predict player ratings for matchmaking using descriptive features like kills and deaths recorded in three-minute intervals [34]. The predicted rating showed a higher correlation with the win rate than the existing rating system, particularly for players with fewer than 15 games.

4 Dataset

1.2 million Lichess games from April 2021 to July 2024 are used in the dataset, with 30,000 games from each month.¹ Lichess is a chess website chosen because it is open source and contains the largest publicly available database of games for free use. The data is split randomly with 80% used for the training set and 20% used for the test set. Other available input features include the amount of clock time remaining after each move. The outputs to be estimated are the ratings of both players for a single game.

Moreover, we applied the same model architecture to the 2024 IEEE Big Data 2024 Cup Chess Puzzle Difficulty Competition [11]. This data contains a Forsyth-Edwards Notation (FEN) [5] string representation of the starting position, along with the moves in Universal Chess Interface (UCI) [10] notation for the solution.

4.1 Time Controls

Lichess time controls are less than 29 seconds for UltraBullet, 179 for Bullet, 479 for Blitz, 1499 for Rapid, else Classical. Games with increment add 40 times the increment to the total time [15]. A time control of "5+3", which is 5 minutes per side with an additional 3 seconds for each move, becomes $5 \times 60 + 40 \times 3 = 420$ seconds. 420 seconds is less than 479 and is considered blitz. 40 comes from an approximation of the average number of moves in a game.

4.2 Metrics

The Mean Absolute Error (MAE) is used to provide an intuitive metric to understand the performance of models. The MAE is given for the pre-standardized rating to allow for a more interpretable loss. The test data is split up by time control to see how model performance varies between time controls. The metric used for the Chess Puzzle Competition is the mean squared error (MSE). We also provide the MSE for the game ratings.

5 Method

5.1 Input Features

We use a similar input representation as AlphaZero, with one plane for each piece-color combination, totaling 12 planes [26]. Each square contains a 1 if the corresponding piece is present; otherwise, it includes a 0. For example, the plane for black pawns contains 1s in the seventh row and 0s everywhere else. Following [25], the remaining clock-time is included as a feature. That paper showed a strong correlation between winning likelihood, remaining time, and position

¹ Games are taken after March 2021 because of some incorrect results that month in the Lichess database due to a datacenter fire

evaluation in 3-minute chess games. The remaining clock-time feature is standardized by subtracting the mean and dividing by the standard deviation. The same standardization process is applied to the ratings for more stable learning. The mean rating is 1514, with a standard deviation of 366. The same input is used for the puzzle competition, except it does not have clock time features because they were unavailable. FEN string representations were converted to the array representation.

5.2 Model Architecture

The architecture, which we will refer to as RatingNet, is a CNN that takes in each board state and passes this into a Bidirectional Long Short Term Memory Neural Network (LSTM) to predict both player ratings for the game after each move. CNNs have demonstrated success in various computer vision tasks, having been initially applied to document recognition [13]. LSTMs have also been proven to be successful in modeling long-term dependencies [9]. A four-layer CNN is used with batch normalization and mean pooling after each layer. The CNN is kept shallow to mitigate overfitting without having to train on a lot of data. The output of the CNN is concatenated with the amount of clock time spent for that corresponding move. This is then fed through the LSTM, which has two fully connected layers on top to output the ratings for the players of that game. The LSTM takes in the hidden state representation at the previous move and clock usage as input. One dropout layer is used after the first fully connected linear layer. Leaky ReLU activations are employed [32]. A simple baseline for predicting the mean chess rating is also shown for comparison.

RatingNet does not use one player’s rating as input to predict the other player’s rating. The reason is to focus the network on understanding chess rather than just predicting a rating close to the other player’s rating. This is to mitigate potential problems when the rating difference is significant, one player’s rating is provisional, or when a player is playing on an alternate account.

6 Results

Table 1. MAE of Different Methods Across Various Chess Time Controls

Time Control	RatingNet	RatingNetNoClock	Mean
Average Test Loss	182	239	346
Ultrabullet Loss	186	269	250
Bullet Loss	182	274	340
Blitz Loss	183	244	378
Rapid Loss	182	207	305
Classical Loss	151	185	246

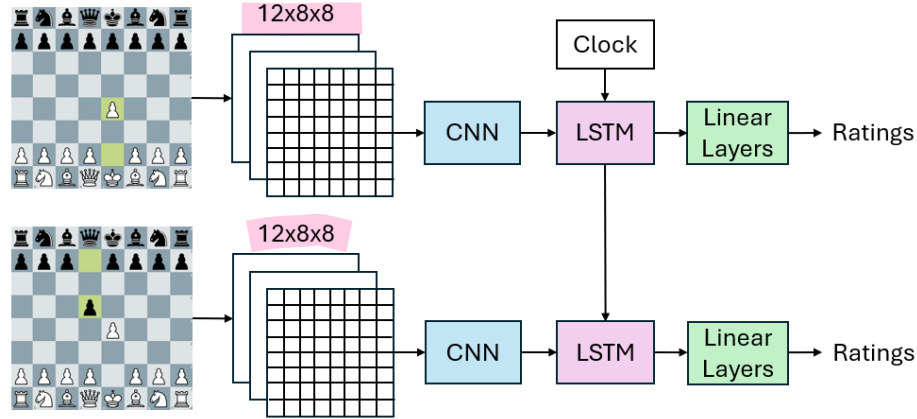


Fig. 1. The model architecture used to predict chess ratings after each move using a CNN and LSTM taking in the remaining clock time feature. This shows the first two time steps of a sample game.

Table 2. MSE of Different Methods Across Various Chess Time Controls

Time Control	RatingNet	RatingNetNoClock	Mean
Average Test Loss	56,618	91,619	183,538
Ultrabullet Loss	56,927	115,833	95,459
Bullet Loss	57,569	117,776	176,296
Blitz Loss	57,330	94,048	216,803
Rapid Loss	56,433	71,215	141,016
Classical Loss	38,491	56,032	94,304

The average MAE across the time controls is 182 for RatingNet. This is reasonable given that some chess categories, such as the United States Chess Federation (USCF), are 200 points per class [30]. The loss for longer-time controls is lower. The most likely reason for this is that the range of ratings decreases as the time control increases, partially because faster time controls are more popular. In addition, faster time controls may have a bit more randomness because players have less time to think, making it harder to estimate the skill levels.

The results may need to be taken with a grain of salt however because the percentage of times that the white player won with a higher rating and when the black player won with a higher rating was calculated only to be 53% in the training data. Such rating problems also appear in over-the-board chess games and possible factors for this center on the fact that rating systems are often overoptimistic for players with higher ratings due to the simplicity of their model [8]. Another consideration is that there is no rating label for each move, but only for the player after having played a certain number of games. The rating outputs after each move is modeling the rating estimate of the player after having seen that move and all the previous moves made by the players in that game.

We additionally report the MSE on the test set of the IEEE Big Data 2024 Cup Chess Puzzle Difficulty Competition. On the public leaderboard, the MSE was 82,049. The score is comparable to the MSE for RatingNetNoClock on estimating rating in games.

7 Clock Ablation

Because faster time controls have less time for thinking and are closer to random play than longer time controls, our hypothesis was that the correlation between move quality and chess rating may weaken, thus having additional information such as the amount of time spent on each move could be useful. To test this, an ablation experiment was performed to determine the usefulness of the clock time feature. Without the clock time feature (setting the clock input to a constant stream of 0s), the model does slightly worse; specifically, the addition of clock information reduces the MAE by 57 points, a 24% improvement. Notably, the clock feature helps significantly at faster speeds, such as bullets, with a 92-point reduction in error, or a 34% improvement. When the time control is longer, for example, in classical, the error is the lowest, with a 151 MAE for RatingNet. Overall, the network is able to utilize the time spent per move as an effective feature for skill estimation.

8 Game Analysis

A couple of sample games are investigated to determine how different types of moves affect the rating estimation of the model. We sample a game shown in Fig. 2 in which the model performs well. It correctly lowered the rating estimate of the player of the white pieces after the move Nf5, which allows the knight to be taken by the opponent’s bishop. The full game is shown in the appendix as game 1. We also analyze a game with high loss, shown in the appendix as game 2. This bullet game was played between a white rating of 2922 and a black rating of 3163. The final predicted rating for white is 1459 and for black it is 1460. Because the average rating is around 1500, we do expect higher ratings to present more difficulties because there are fewer examples of such high rated games. We expect that in some games however, RatingNet would provide more accurate rating estimations than the player’s actual rating as players do have off days or games where they play really well. With an accurate enough rating estimation system, one could consider further extending this work to potentially detect engine use or external assistance. For example, if a specific move could be marked as having a much higher rating than what the player’s true strength is, further investigation could occur.

Un movimiento que por ejemplo, en base a otras partidas con esa posición, nunca se hizo y de repente si

9 Conclusion

Our proposed method for online chess rating estimation leverages game moves and clock times without relying on hand-crafted features for a variety of time con-



Fig. 2. White plays Nf5, a blunder because black can take it with the bishop on d3. White’s estimated rating goes from 1255 to 1241. Their actual Lichess rating in this bullet game is 1224.

trols. By integrating a CNN-LSTM architecture, we demonstrated the potential to accurately predict player rating on a move-by-move basis. The performance of the model, with an MAE of 182, shows promise in improving rating systems, providing real-time and dynamic assessments of the player’s skills. This novel approach could also serve as a foundation for applications beyond rating estimation, such as anomaly detection in games and paves the way for more granular rating systems in chess

10 Appendix

10.1 Hyperparameters and code

training batch size: 32, learning rate: 1e-4, weight decay: 1e-5, epochs: 50, learning rate patience: 10, learning rate factor: 0.5, dropout: 0.5.

The model was trained on one A40 gpu and took 12 hours to finish training. The code: <https://github.com/AstroBoy1/RatingNet>

10.2 Game 1

1. e4 e5 2. Nf3 Nc6 3. d4 exd4 4. Nxd4 Nf6 5. Nxc6 dxc6 6. Nc3 Qxd1 7. Nxd1 Bc5 8. Bd3 O-O 9. O-O Re8 10. Be3 Bxe3 11. Nxe3 Nxe4 12. Bxe4 Rxe4 13. Nd1 Bf5 14. f3 Re2 15. Rf2 Rxc2 16. Rxc2 Bxc2 17. Ne3 Bd3 18. Nf5 Bxf5 19. g4 Bg6 20. Kf2 Rd8 21. Kg3 Rd2 22. Rb1 Bxb1 23. h4 Rxb2 24. h5 Rxa2 25. g5 Bf5 26. f4 h6 27. Kh4 hxg5 28. Kxg5 Bd3 29. h6 gxh6 30. Kxh6 Rg2 31. f5 f6 32. Kh5 Be2 33. Kh6 Rg5 1/2-1/2

10.3 Game 2

1. e4 g6 2. d4 Bg7 3. Nc3 d6 4. h4 Nf6 5. f3 Nc6 6. Be3 h5 7. Nge2 e5 8. d5 Ne7
 9. Qd2 O-O 10. O-O-O a6 11. Bh6 b5 12. Bxg7 Kxg7 13. g4 hxg4 14. fxg4 Bxg4
 15. a3 Rb8 16. Bg2 b4 17. axb4 Rxb4 18. Qd3 Qb8 19. b3 a5 20. Kd2 Qb6 21.
 Qe3 Bxe2 22. Qxb6 cxb6 23. Kxe2 Rc8 24. Kd3 Rd4 25. Ke2 Rxc3 0-1

References

1. Breiman, L.: Random forests. *Machine learning* **45**, 5–32 (2001)
2. Chess.com: Chess.com play chess online, <https://www.chess.com>
3. Chowdhary, S., Iacopini, I., Battiston, F.: Quantifying human performance in chess. *Scientific Reports* **13**(1), 2113 (2023)
4. Cortes, C.: Support-vector networks. *Machine Learning* (1995)
5. Edwards, S.: Forsyth-edwards notation, https://www.chessprogramming.org/Forsyth_Edwards_Notation
6. Elo, A.E., Sloan, S.: The rating of chessplayers: Past and present. (No Title) (1978)
7. Glickman, M.E.: Example of the glicko-2 system. Boston University **28** (2012)
8. Glickman, M.E., Jones, A.C.: Rating the chess rating system. *CHANCE-BERLIN THEN NEW YORK-* **12**, 21–28 (1999)
9. Hochreiter, S.: Long short-term memory. *Neural Computation* MIT-Press (1997)
10. Huber, R., Meyer-Kahlen, S.: Universal chess interface, <https://www.chessprogramming.org/UCI>
11. IEEE: Bigdata 2024 cup: Predicting chess puzzle difficulty. <https://knowledgepit.ml/predicting-chess-puzzle-difficulty/>
12. Laarhoven, T., Ponukumati, A.: Towards transparent cheat detection in online chess: An application of human and computer decision-making preferences. In: *International Conference on Computers and Games*. pp. 163–180. Springer (2022)
13. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86**(11), 2278–2324 (1998)
14. Lichess: Kaladin: Machine learning tool aimed at automating cheat detection using insights data., <https://github.com/lichess-org/kaladin>
15. Lichess: Lichess faq, <https://lichess.org/faq#time-controls>
16. Lichess: Lichess: free online chess server. <https://lichess.org>
17. McIlroy-Young, R., Sen, S., Kleinberg, J., Anderson, A.: Aligning superhuman ai with human behavior: Chess as a model system. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. pp. 1677–1687 (2020)
18. McIlroy-Young, R., Wang, R., Sen, S., Kleinberg, J., Anderson, A.: Learning models of individual behavior in chess. In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. pp. 1253–1263 (2022)
19. Monroe, D., Team, T.L.C.Z.: Mastering chess with a transformer model. *arXiv preprint arXiv:2409.12272* (2024)
20. Patria, R., Favian, S., Caturdewa, A., Suhartono, D.: Cheat detection on online chess games using convolutional and dense neural network. In: *2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*. pp. 389–395. IEEE (2021)

21. Reddy, K.V., Kumar, B., Kumar, N.P., Parasuraman, T., Balasubramanian, C., Ramakrishnan, R.: Chess match outcome prediction via sequential data analysis with deep learning. In: 2023 International Conference on Sustainable Communication Networks and Application (ICSCNA). pp. 1442–1447. IEEE (2023)
22. Regan, K., Haworth, G.: Intrinsic chess ratings. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 25, pp. 834–839 (2011)
23. Rozman, L.: Gothamchess: Chess youtube channel, <https://www.youtube.com/@GothamChess>
24. Ruoss, A., Delétang, G., Medapati, S., Grau-Moya, J., Wenliang, L.K., Catt, E., Reid, J., Genewein, T.: Grandmaster-level chess without search. arXiv preprint arXiv:2402.04494 (2024)
25. Sigman, M., Etchemendy, P., Slezak, D.F., Cecchi, G.A.: Response time distributions in rapid chess: a large-scale decision making experiment. *Frontiers in Decision Neuroscience* **4**, 60 (2010)
26. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* **362**(6419), 1140–1144 (2018)
27. Snyder, J.: Inverse interpolation, a real root of $f(x)=0$. University of Illinois Digital Computer Laboratory, ILLIAC I Library Routine H1-71 **4** (1953)
28. Tijhuis, T., Blom, P.M., Spronck, P.: Predicting chess player rating based on a single game. In: 2023 IEEE Conference on Games (CoG). pp. 1–8. IEEE (2023)
29. Tord Romstad, Marco Costalba, Joona Kiiski, Gary Linscott, open source community: Stockfish Chess Engine, <https://stockfishchess.org/>
30. United States Chess Federation: Uscf, <https://new.uschess.org/>
31. Vasik Rajlich: Rybka, <http://www.rybkachess.com/>
32. Xu, J., Li, Z., Du, B., Zhang, M., Liu, J.: Reluplex made more practical: Leaky relu. In: 2020 IEEE Symposium on Computers and communications (ISCC). pp. 1–7. IEEE (2020)
33. Zaidi, K., Guerzhoy, M.: Predicting user perception of move brilliance in chess. arXiv preprint arXiv:2406.11895 (2024)
34. Zhang, C., Wang, K., Chen, H., Fan, G., Li, Y., Wu, L., Zheng, B.: Quickskill: Novice skill estimation in online multiplayer games. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. pp. 3644–3653 (2022)