Estimating chess puzzle difficulty without past game records using a human problem-solving inspired neural network architecture

Anan Schütt, Tobias Huber, Elisabeth André

Angaben zur Veröffentlichung / Publication details:

Schütt, Anan, Tobias Huber, and Elisabeth André. 2024. "Estimating chess puzzle difficulty without past game records using a human problem-solving inspired neural network architecture." In 2024 IEEE International Conference on Big Data (BigData), 15-18 December 2024, Washington, DC, USA, edited by Chang-Tien Lu Lu, Fusheng Wang, Bolong Zheng, and Yifeng Gao, 8396–8402. Piscataway, NJ: Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/bigdata62323.2024.10826087.

Estimating Chess Puzzle Difficulty Without Past Game Records Using a Human Problem-Solving Inspired Neural Network Architecture

1st Anan Schütt
University of Augsburg
Augsburg, Germany
anan.schuett@uni-a.de

2nd Tobias Huber
Technische Hochschule Ingolstadt
Ingolstadt, Germany
Tobias.Huber@thi.de

3rd Elisabeth André
University of Augsburg
Augsburg, Germany
elisabeth.andre@uni-a.de

***Abstract*—For chess players to sharpen their tactical skills effectively, they train on chess puzzles with a fitting difficulty level. This paper presents an approach to estimate the difficulty level of chess puzzles using a deep neural network. The proposed approach achieved second place in the *IEEE BigData Cup 2024 competition: Predicting chess puzzle difficulty*. For the design of our network architecture, we take inspiration from the human problem-solving process for chess puzzles. We train the model to predict the correct move as an auxiliary task to improve the training process. We also predict themes, which are patterns in chess puzzles as a second auxiliary task. Finally, we use the uncertainty in the position, i.e. how incorrect the model's move prediction is, as a further input to guide the estimation of the puzzle difficulty.**

***Index Terms*—chess puzzle, difficulty estimation, neural network**

I. Introduction

A major part of learning chess is going through chess puzzles. A chess puzzle is a position or a sequence of consecutive positions where only one correct series of moves leads to a decisive advantage, which means either ending the game in checkmate, or winning more material than the opponent. In chess platforms, solving a puzzle happens interactively, with the player playing the advantage-taking side against the computer, whose responses are fixed beforehand. Solving a puzzle often entails recognizing patterns (called themes) in the position [29]. With puzzles, players learn to recognize the themes that occur in critical positions and transfer this recognition skill into their own games, as commonly advocated by chess educators [6], [25].

In Figure 1 we show an example of a chess puzzle. One prominent theme in this puzzle is the fork, which is when a piece moves to attack two other pieces that are more valuable, guaranteeing that one of them can be taken. However, the fork is not apparent from the start. One needs to see one move further to find the move rook to the h7 square, giving check and forcing the king to move to h7 to take the rook. Then, the knight can move to g5, "forking" the king on h7 and the queen on d4, which forces the king to move again, after which the knight can take the queen. Over this forcing sequence, white trades his rook for the opponent's queen, which is an advantageous trade.

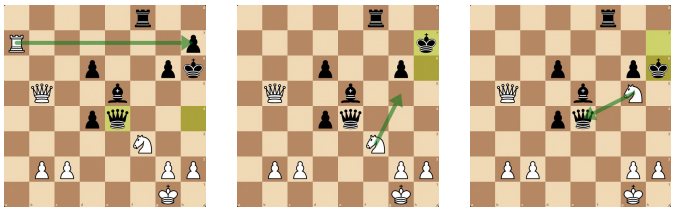

Fig. 1. Example puzzle with the fork as a theme. The figure shows subsequent positions where white makes a move. By sacrificing a rook, white can use its knight to attack the black king and queen at the same time.

As puzzles encompass a huge set of positions, their difficulty levels vary greatly, from simple two-move sequences to complex positions with multiple overlapping themes. Thus, it makes sense for players to only practice on puzzles that are on their level to keep the player from being under-challenged or over-challenged, which is necessary for achieving the state of flow [2], [5]. For this, it is important to know the difficulty level of chess puzzles. Investigating this problem for chess puzzles also provides insights into other similar learning tasks like practice exercises in mathematics or computer science [18], [22], [24].

Many platforms that offer chess puzzles already have a system to provide puzzles at the appropriate difficulty level to players [32], [33]. This is achieved using the Elo rating system [4], or a closely related variant like the Glicko system [8]. These methods use historical data of players attempting different puzzles to calculate players' ability ratings and puzzles' difficulty ratings. Hence, these methods do not work when we consider a newly generated position from a newly played game or an authored position, since there is no historical data to calculate with. This means new puzzles will be matched virtually randomly with players, potentially causing an annoying user experience.

In this paper, we propose a method to estimate the difficulty level of chess puzzles without using historical data. We use a deep convolutional neural network and weighted averaging

over the puzzle's sequence of positions to predict the difficulty level. Importantly, inspired by human puzzle-solving, we include puzzle theme and solution move prediction as auxiliary tasks to help the network learn. We also use the cross-entropy between the predicted and the true solution move as input to the network to reflect the human decision process in chess, which improves prediction performance. Our model achieved second place in the *IEEE BigData Cup 2024 competition* [31].

II. Related Work

At present, the difficulty ratings of chess puzzles are mostly determined using systems such as the Elo system [4] or Glicko system [8]. These systems work very similarly in principle. The process starts by giving a puzzle without a known difficulty rating a preliminary difficulty rating, usually a value in the middle of the rating range. Then, players of different ratings attempt the puzzle, to which the rating is updated, depending on the outcome and the player's rating. The process repeats, making the difficulty rating landing close to the true value. Systems such as these require many players to interact with the puzzle to know the difficulty value, which might cause mismatches between the player and the puzzle, deteriorating the user experience. We address this by estimating the puzzle difficulty without requiring any previous player attempts.

Previous works on estimating chess puzzle difficulty without past game records have used the search tree of chess engines to estimate a chess puzzle's difficulty level [9], [27]. These works focused on features such as search tree depth, spanning factor of different nodes, and the engine's evaluation score. While some of these approaches already attempted to model the human problem-solving process [27], their approach does not fully capture how humans think, as human cognition differs from that of a chess engine. Humans tend to make predictable mistakes, since some chess moves seem counterintuitive to the human eye, making players overlook them, such as moving pieces backward [19]. Hristova et al. [11] studied different factors that influence how difficult a chess puzzle is for humans, and identified two main factors: how difficult it is to spot the theme(s), and how many reasonable-looking variations there are in the position. In this work, we aim to utilize these human-centered factors but instead of using the search tree of chess engines, we utilize a neural network.

Neural networks have shown similarities to human problem-solving when applied to chess and other puzzle games. McIlroy-Young et al. [16], [17] trained and studied neural networks that simulate how people of different ability levels play chess, indicating that neural networks can imitate the imperfect human decision process. Furthermore, Jenner et al. [13] demonstrate that chess-playing neural networks show signs of planning ahead, which is also part of the human problem-solving process when playing chess. Neural networks have also been applied to predict chess player ratings from looking through a game [28] and to detect cheating [21]. Additionally, neural networks were used to assess the difficulty rating of other puzzle games [3], [30]. To the best of our knowledge, neural networks have not been used to estimate the difficulty of chess puzzles. Compared to a normal game of chess, chess puzzles come with unique challenges like spotting the theme of the puzzle [11]. As pointed out in the example puzzle in Figure 1, the themes of a puzzle are not always apparent from the starting position, increasing its difficulty level for humans.

Continuing on the findings that neural networks can model human behavior in chess, we apply a neural network to predict chess puzzle difficulty. We advance previous neural network approaches in chess by developing a novel architecture specifically designed to target chess puzzles. This architecture is inspired by the human problem-solving strategies observed during chess puzzle-solving, as identified in the work of Hristova et al. [11].

III. Methods

A. Dataset & Data Preparation

We use a dataset of 3.6 million chess puzzles provided by the Lichess database [34], containing the starting positions and the difficulty levels calculated using the Glicko system. In addition to that, the dataset also provides the sequence of correct moves, the list of themes associated with the puzzle, and the number of players who have attempted the puzzle.

There are three parts to representing the chess puzzle in the prediction model. The first part is the position, which we encode as a binary tensor of size 18x8x8, with each of the 18 channels representing different piece types or board status, similar to [26]. There are six types of chess pieces: pawn, rook, knight, bishop, queen, and king. We use one channel for each piece type and for each player to encode which square contains that piece type of that player, making up 12 channels. We also need to encode whether the two special moves in chess: castling and en passant, are possible[1]. For this, we use four channels to encode king- and queenside castling rights for each player, where the corresponding channel is filled with ones if castling on the respective side is still valid. Lastly, there are two channels to represent en passant for the person at play, one channel indicates whether en passant is possible at all, and another one indicates at which square it is possible. We normalize the positions so that the encoding uses certain channels for the player at play, and other ones for the opponent, instead of encoding by black/white pieces.

The second part of the puzzle representation is the move number encoding. Here, we encode how many moves there are since the start of the puzzle to the current position, and how many moves there are left until the end. To this end, we use two sets of one-hot encodings of length 15 (the maximum length of a puzzle) where the 1 entry denotes the current move and the number of remaining moves, respectively. By encoding how many moves it has been since the start and how many moves there still are until the end, the length of the puzzle is implicitly encoded.

[1]Castling is a move where the king and a rook on one of the sides move past each other, and en passant is a special type of pawn capture. These moves have their specific conditions when they can be played.

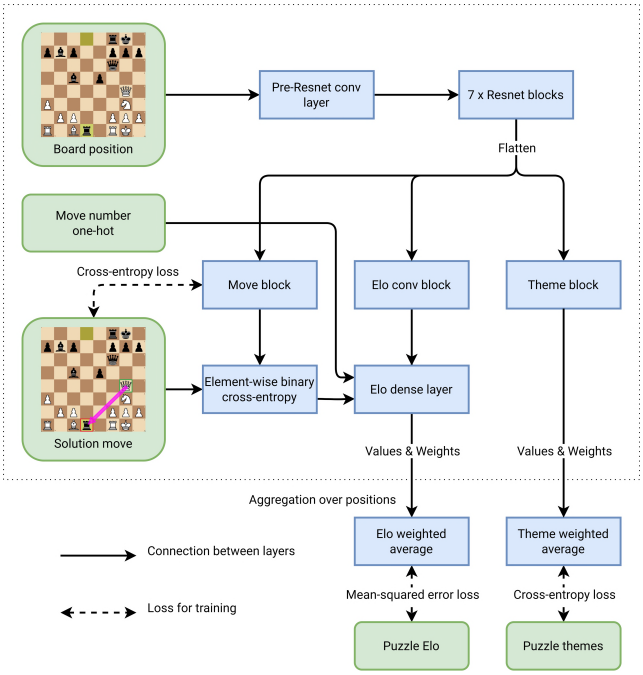

Fig. 2. Diagram of our proposed network architecture.

The third part of the puzzle representation is the correct move of each position. We encode the correct move at the current position with two one-hot vectors of length 64, one for the move's starting square and one for the target square.

The labels of the prediction task also consist of three parts: the Elo rating, the associated themes, and the correct next move. We normalize the Elo ratings to have a mean of zero and a standard deviation of one. We encode the associated themes as binary vectors of length 61, and the correct next move using the same method as in the input. Note that the Elo rating and the themes are puzzle-level labels, meaning the label is shared between the multiple positions inside the puzzle. In contrast, the correct next move is a position-level label.

B. Network Structure

The structure of the prediction network mainly takes inspiration from the chess engine Leela [36]. The network has a convolutional (abbreviated as conv in Diagram 2 and Table I) layer, followed by a series of Resnet blocks [10] with squeeze-and-excitation layers [12] as the main trunk. We refer to this main trunk as ϕ_{CNN}. Following this main trunk, the network has separate heads for different tasks, as depicted in Figure 2. We provide the composition and the layer sizes of the different blocks in Table I.

The core of the network processes chess positions, while the Elo rating is a puzzle-level label, so there needs to be an aggregation mechanism. For this, we use the weighted average. Each position outputs one Elo rating estimate and one weight value. The weight values of all the positions in a puzzle are passed through the softmax function together to get the actual weight for averaging. Formally, the last layer of the

TABLE I
COMPOSITION AND LAYER SIZES OF THE BLOCKS INSIDE THE NEURAL NETWORK

Block	Layer list	Layer sizes
Pre-Resnet conv layer	Conv	18x128x3x3
	Batch Norm + ReLU	
Resnet block	Conv	128x128x3x3
	Batch Norm + ReLU	
	Conv	128x128x3x3
	Batch Norm + ReLU	
Squeeze-and-excitation (part of Resnet)	Avg. pool	
	Dense	128x32
	ReLU	
	Dense	32x256
	Sigmoid & Identity	
Theme & Move block	Conv	128x64x1x1
	Dense	4096x256
	ReLU	
	Dense	256x61 / 256x128
	Sigmoid	
Elo conv block	Conv	128x64x1x1
	Dense	4096x256
	ReLU	
Elo dense layer	Dense	414x1

Elo estimation head ϕ_{elo_dense} outputs two numbers for each input position $\mathbf{x}_i$ in the puzzle: v_i^{elo} and w_i^{elo}, which are the prediction values and the prediction weights, respectively. To aggregate the outputs, we apply the following operation:

$$\hat{y}^{elo} = \sum_i \frac{e^{w_i^{elo}}}{\sum_j e^{w_j^{elo}}} v_i^{elo} \tag{1}$$

where the summations go over the positions in the puzzle.

Inspired by the human problem-solving process during chess puzzles, we use two auxiliary tasks to help the network develop a better understanding of the positions within the puzzle: correct move prediction and theme prediction. For the first auxiliary task, the network has a dense block ϕ_{move} that predicts which move the chess engine deems to be the best move for position $\mathbf{x}_i^{pos}$ within the puzzle. Note, that because of the move encoding scheme we use, it is possible for the network to output an illegal move. However, this possibility doesn't contradict our intention with the network. The second auxiliary task is to predict the theme associated with the puzzle using the theme block ϕ_{theme}. This auxiliary task uses the same weight average method as described in Equation 1. For theme prediction, the weights and values for aggregation are vector outputs from the theme block.

Finally, we include uncertainty in the current position as input to the Elo dense layer. This acts as a proxy for the number of reasonable-looking move variations in the current position, which was shown to influence the human puzzle-solving process [11]. We provide uncertainty by using the cross-entropy between the network's prediction and the correct move label. Let

$$\hat{\mathbf{y}}_i^{move} = \phi_{move}(\phi_{CNN}(\mathbf{x}_i^{pos})) \tag{2}$$

be the network's move prediction for the current position $\mathbf{x}_i^{pos}$. Then the uncertainty is given by the cross-entropy

$$\mathbf{CE}_i^{move} = -\mathbf{y}_i^{move} \odot log(\hat{\mathbf{y}}_i^{move}), \tag{3}$$

where $\mathbf{y}_i^{move}$ is the correct move in position $\mathbf{x}_i^{pos}$ as provided by the puzzle representation.

The main task of the network is to output the Elo rating of the puzzle. For this, the puzzle's position $\mathbf{x}_i^{pos}$ is passed through the main trunk ϕ_{CNN} and an Elo-specific CNN block ϕ_{elo_conv}. The result of ϕ_{elo_conv} is concatenated with the uncertainty $\mathbf{CE}_i^{move}$ and the move number one-hot encoding $\mathbf{x}_i^{num}$. In total, we get the Elo prediction for the input position $\mathbf{x}_i^{pos}$ with corresponding move number one-hot encoding $\mathbf{x}_i^{num}$ through

$$(v_i^{elo}, w_i^{elo}) = \tag{4}$$
$$\phi_{elo\_dense}(\phi_{elo\_conv}(\phi_{CNN}(\mathbf{x}_i^{pos})) \oplus \mathbf{x}_i^{num} \oplus \mathbf{CE}_i^{move}).$$

C. Training Setup

The loss function we use for training consists of three parts, as depicted by dashed lines in Figure 2. First is the mean squared error for the difficulty rating prediction, representing the main task. The two other parts are cross-entropy loss for puzzle themes and correct next move, which can be seen as auxiliary tasks.

We split the dataset into the training and validation sets, with an 80:20 ratio. We do not dedicate a split as the test set, since our target test set is the challenge's dataset (see Section IV-A).

We implement the model using PyTorch [20], and use the Adam optimizer [14] for training. We train the model and monitor the main validation loss, i.e. the mean squared error of the difficulty rating. After each training epoch, we save the model with the lowest main validation loss, and stop training when there is no improvement after three epochs.

IV. Evaluation

Our evaluation methods can be mainly divided into two parts. The first is the main testing on the *IEEE BigData Cup* test set, which counts toward the challenge. We perform further evaluations on the validation set we have locally split from the training set to gain more insights about our proposed architecture.

A. IEEE BigData Cup

The goal of the challenge [31] is to predict the difficulty rating of chess puzzles as closely as possible. The challenge uses the mean squared error of the unnormalized Elo rating (ranging from 399 - 3331) as the evaluation metric. As the provided chess puzzle dataset is publicly available, the challenge organizers prepared a separate test dataset.

The test dataset contains 2,282 puzzles that are derived from real games using the same method as the public Lichess dataset [34]. The labels in the test dataset are calculated using the Glicko system [8] with solving attempts from approximately 20 - 50 attempts per puzzle. The test dataset is then again divided into two parts: the holdout test dataset and the final test dataset. During the challenge, only the evaluation score on the holdout test is known. The evaluation score on the final test dataset is only reported after the challenge was over, and is used to rank the solutions.

As the test dataset has relatively few attempts for each calculation, the ratings might not be accurately computed for every puzzle and might have a different distribution than the public training dataset. Furthermore, by simulating the Glicko system on a similar scale of data, we found that the ratings of extremely easy and extremely hard puzzles regress toward the population mean. To specifically correct our predictions for this challenge's distribution shift, we tested different linear transformations to make rating predictions closer to the mean. We then choose the linear transformation that is not too drastic while still giving good predictions on the holdout test dataset. To be precise, we apply the following transformation:

$$\hat{y}^{trans} = c + g(\hat{y} - c) + s \tag{5}$$

where g, c, and s are the gradient, center point, and shift of the transformation, respectively.

We only apply this transformation for the evaluation on the *IEEE BigData Cup* test dataset, and not on any of the following evaluations.

B. Ablation Study on Validation Set

We use our own validation set to conduct an ablation study and investigate the utility of different components in our approach. As a first step, we evaluate our final model in predicting the puzzle difficulty rating on the validation set. We use the mean squared error of the unnormalized Elo rating for evaluation, aligning with the *IEEE BigData Cup* challenge.

Following that, we compare different variations of the model which were trained throughout the development of the final version. Here, we again use the mean squared error of the unnormalized Elo rating on the validation set. In each version of the model, we use the same training procedure and the same number of neurons unless stated otherwise.

The components of the model, as described in Section III-B, that we compare here are:

- *Move pred* - These models use the correct move prediction as an auxiliary task
- *Move CE* - These models use the cross-entropy between the correct move label and predicted correct move as an additional input to the Elo dense block
- *Max-pool* - These models use the maximum prediction over all puzzle positions as the prediction for the puzzle-level labels, i.e. the difficulty rating and the themes
- *W-avg-pool* - These models use the weighted average operation to aggregate position predictions into the puzzle-level prediction
- *Filters* - This states the number of filters in the Resnet blocks of the model. This is the only number of neurons we vary between models.

TABLE II
FINAL RESULTS OF THE IEEE BIGDATA CUP, SHOWING THE FIRST FIVE PLACES OUT OF 37.

Rank	MSE Holdout	MSE Final
1	49141	**104540**
2	58810	**120682**
3	69890	**123103**
4	61381	**129245**
5	65136	**132631**

TABLE III
EVALUATION RESULTS OF THE ABLATION STUDY. THE COLUMNS ARE DESCRIBED IN SECTION IV-B

Filters	*Move pred*	*Move CE*	*Max-pool*	*W-avg-pool*	MSE
64					167759
64	✓				152794
64	✓	✓			122293
64	✓	✓	✓		89144
128	✓	✓	✓		74514
128	✓	✓		✓	56537

Note here that models that used neither Max-pool nor W-avg-pool only used the first position of a puzzle as input, ignoring the positions that occur while the puzzle progresses.

C. Position-level Difficulty

In addition to the puzzle's difficulty level, our model also implicitly learned the difficulty level for each position in the puzzle. We are interested in whether the position-wise difficulty level matches with human behavior. To this end, we look through 100,000 randomly chosen puzzles in the validation set and the corresponding game where the position occurred. Each puzzle from the validation dataset is generated from a position in a real game, which can be queried through the Lichess API [35]. We analyze these positions, looking at whether the player who got into the puzzle's starting position and standing to gain an advantage, can play on accurately in his game. We refer to this player as the puzzle-taker in the following.

Out of these 100,000 games, we only consider games where the puzzle-taker made at least one correct move following the puzzle, but then also made a mistake, which corresponds to incorrectly solving the puzzle. Additionally, we only considered games with longer time controls (classical and rapid) to filter out mistakes caused by time pressure. From this, we split the positions the puzzle-taker faced into mistake positions (the player made a mistake) and correct positions (moves before the mistake). Here, we hypothesize that if our model captures the difficulty for each position in the puzzle, then the predicted difficulty of mistake positions should be higher than the correct positions. To answer this, we compare the predicted difficulty level of these positions by taking the predicted value before the weighted average aggregation block. We pair up the correct and mistake positions of the same puzzle and use the Wilcoxon signed-rank test to check whether there is a significant difference.

V. RESULTS

We submit the predictions on the test dataset for evaluation. The challenge allowed three final submissions for evaluation. Therefore, we submitted the following versions of the same neural network with different linear transformation parameters (see Equation 5):

- Gradient = 0.8, center point = 1600, shift = 0
- Gradient = 0.89, center point = 1649, shift = -7
- No transformation

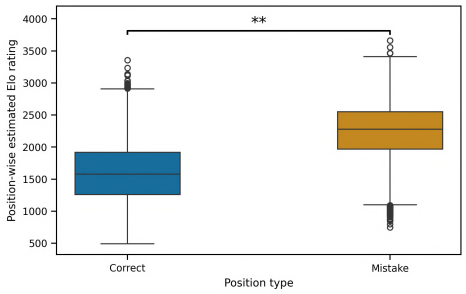

Fig. 3. Predicted difficulty ratings of positions where players make a correct move or a mistake.

Our best model out of the submitted three achieved a mean squared error of 58810 on the holdout test dataset, and 120682 on the final test dataset, landing on the second place in the challenge. Note that we do not have access to the final evaluation results from our other submissions and do not know which of our submitted versions performed best. For comparison, we provide the scores of the first 5 places in Table II.

To check the quality of our predictions locally, we evaluated the predictions on the local validation set without any transformation. We achieved a mean squared error of 0.1916 on the normalized Elo rating, corresponding to a mean squared error of 56537 on the unnormalized Elo rating. For context, the square root of 56537 is about 237, which is close to the gap of 200 Elo points that the US Chess Federation uses to separate players into classes [1].

Examining the model more closely, we present the results from the ablation study in Table III. Finally, we present the results comparing the predicted difficulty level of positions where players made the correct move or a mistake in a real game in Figure 3. We filtered only the interesting games, as described in Section IV-C. This resulted in 3,489 data points for comparison. The Wilcoxon signed-rank test shows a significant difference between the two groups ($z = -44.21$, $p < 0.001$). The test statistic corresponds to a coefficient of determination $r^2 = 0.56$, indicating a large effect size [7].

VI. DISCUSSION

With regards to the *IEEE BigData Cup*, our model has achieved second place, showing that our model has a competitive edge. While not the winner of the challenge, the ranking shows that the model included many aspects important to delivering good predictions for chess puzzle difficulty.

The experimental results in Table III show that our human problem-solving inspired architectural decisions improve performance greatly. The auxiliary tasks and the correct move cross-entropy block helped in processing the positions more precisely. This aligns with the insights from [11], that the uncertainty of the correct move, i.e. the number of reasonable-looking moves, is a good indicator of a puzzle's difficulty. Furthermore, the aggregation methods show the importance of considering the puzzle as a sequence of positions. This coincides with the example we raised in Section I, that the idea of the puzzle might not be apparent from the starting position.

In the same direction, Figure 3 suggests that the network has not only learned to estimate the difficulty of the whole puzzle, but also to distinguish the difficulty of positions within the same puzzle. These results, along with those from the ablation study, indicate that our network has incorporated the human problem-solving process into its prediction, coinciding with previous works that show that neural networks can mimic human chess playing [16], [17].

In future works, it would be interesting to extend the difficulty estimation network to also consider how human players handle tactical positions. One potential question from this is whether there is more than one dimension to the difficulty rating. For example, some players might be extremely good at certain themes but fail at others, similar to [15]. In addition to estimating chess puzzle difficulty, the proposed network could also contribute to problem-solving tasks in mathematics and computer science [18], [22], [24], where students make a sequence of decisions to solve their task, similar to chess puzzles. Having a good estimate of difficulty levels could help in sequencing the exercises that students should practice [23]. The insights from our human problem-solving inspired work could help in understanding the difficulty rating of such exercises and the student's learning progression.

VII. Conclusion

In this work, we proposed a novel neural network architecture for predicting the difficulty rating of chess puzzles without historical data. We demonstrated the prediction performance, receiving second place in the *IEEE BigData Cup 2024* challenge. We also showed that our human puzzle-solving inspired additions to the network architecture improved the performance of the network. This affords further research to consider human problem-solving characteristics when analyzing difficulty levels. Finally, we demonstrated that our network not only estimates the difficulty of the whole chess puzzles, but also identifies difficult positions within the puzzles.

References

[1] US Chess Ratings Committee. The US chess title system. http://www.glicko.net/ratings/titles.pdf, 2016. [Online; accessed 4 October 2024].

[2] Mihaly Csikszentmihalyi. *Flow: The psychology of optimal experience*, volume 1990. Harper & Row New York, 1990.

[3] Philipp Eisen. Simulating human game play for level difficulty estimation with convolutional neural networks, 2017.

[4] Arpad E. Elo. *The Rating of Chessplayers, Past and Present*. Arco Pub., New York, 1978.

[5] Stefan Engeser and Falko Rheinberg. Flow, performance and moderators of challenge-skill balance. *Motivation and emotion*, 32:158–172, 2008.

[6] Frank Erwich. *1001 chess exercises for club players: The Tactics Workbook that also explains all the key concepts*. New in Chess, 2019.

[7] Catherine O Fritz, Peter E Morris, and Jennifer J Richler. Effect size estimates: current use, calculations, and interpretation. *Journal of experimental psychology: General*, 141(1):2, 2012.

[8] Mark E Glickman. The Glicko system. *Boston University*, 16(8):9, 1995.

[9] Matej Guid and Ivan Bratko. Search-based estimation of problem difficulty for humans. In *Artificial Intelligence in Education: 16th International Conference, AIED 2013, Memphis, TN, USA, July 9-13, 2013. Proceedings 16*, pages 860–863. Springer, 2013.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Dayana Hristova, Matej Guid, and Ivan Bratko. Assessing the difficulty of chess tactical problems. *International Journal on Advances in Intelligent Systems*, 7(3):728–738, 2014.

[12] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[13] Erik Jenner, Shreyas Kapur, Vasil Georgiev, Cameron Allen, Scott Emmons, and Stuart Russell. Evidence of learned look-ahead in a chess-playing neural network. *arXiv preprint arXiv:2406.00877*, 2024.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[15] Andrew Lan, Tom Goldstein, Richard Baraniuk, and Christoph Studer. Dealbreaker: A nonlinear latent variable model for educational data. In *International Conference on Machine Learning*, pages 266–275. PMLR, 2016.

[16] Reid McIlroy-Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. 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*, pages 1677–1687, 2020.

[17] Reid McIlroy-Young, Russell Wang, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Learning personalized models of human behavior in chess. *arXiv preprint arXiv:2008.10086*, 2020.

[18] Jauwairia Nasir, Barbara Bruno, Mohamed Chetouani, and Pierre Dillenbourg. What if social robots look for productive engagement? automated assessment of goal-centric engagement in learning applications. *International Journal of Social Robotics*, 14(1):55–71, 2022.

[19] Emmanuel Neiman and Yochanan Afek. *Invisible chess moves: Discover your blind spots and stop overlooking simple wins*. New in Chess, 2014.

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019.

[21] Reyhan Patria, Sean Favian, Anggoro Caturdewa, and Derwin Suhartono. 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)*, pages 389–395. IEEE, 2021.

[22] Chris Piech, Mehran Sahami, Jonathan Huang, and Leonidas Guibas. Autonomously generating hints by inferring problem solving policies. In *Proceedings of the second (2015) ACM conference on Learning@ Scale*, pages 195–204, 2015.

[23] Mary McCaslin Rohrkemper. Self-regulated learning and academic achievement: A Vygotskian view. In *Self-regulated learning and academic achievement*, pages 143–167. Springer, 1989.

[24] Anan Schütt, Tobias Huber, Ilhan Aslan, and Elisabeth André. Fast dynamic difficulty adjustment for intelligent tutoring systems with small

datasets. In *Proceedings of the 16th International Conference on Educational Data Mining*, pages 482–489, 2023.

[25] Yasser Seirawan and Jeremy Silman. *Play winning chess*. Gloucester Pub. plc, 2007.

[26] 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.

[27] Simon Stoiljkovikj, Ivan Bratko, and Matej Guid. A computational model for estimating the difficulty of chess problems. In *Proceedings of the third annual conference on advances in cognitive systems ACS*, page 7, 2015.

[28] Tim Tijhuis, Paris Mavromoustakos Blom, and Pieter Spronck. Predicting chess player rating based on a single game. In *2023 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2023.

[29] Daniel A Wagner and Martin J Scurrah. Some characteristics of human problem-solving in chess. *Cognitive psychology*, 2(4):454–478, 1971.

[30] Xuan Wei. Difficulty level classification of sudoku puzzles based on convolutional neural network. *Academic Journal of Computing & Information Science*, 6(11):35–39, 2023.

[31] Jan Zyśko, Maciej Świechowski, Sebastian Stawicki, Katarzyna Jagieła, Andrzej Janusz, and Dominik Ślęzak. IEEE Big Data Cup 2024 report: Predicting chess puzzle difficulty at KnowledgePit.ai. In *IEEE International Conference on Big Data, Big Data 2024, Washington DC, USA, December 15-18, 2024*. IEEE, 2024.

[32] Chess - #1 platform for online chess. https//www.chess.com, 2005. [Online; accessed 1 October 2024].

[33] Lichess - a free/libre, open-source chess server. https//www.lichess.org, 2010. [Online; accessed 1 October 2024].

[34] Lichess open database. https://database.lichess.org/#puzzles, 2010. [Online; accessed 1 October 2024].

[35] Lichess.org API. https://lichess.org/api, 2010. [Online; accessed 4 October 2024].

[36] Leela Chess Zero Neural Network Topology. https://lczero.org/dev/backend/nn/, 2020. [Online; accessed 20 September 2024].