Some Chess-Specific Improvements for Perturbation-Based Saliency Maps

Jessica Fritz
Johannes Kepler University
Linz, Austria
jessy.fritz@gmx.at

Johannes Fürnkranz
Johannes Kepler University
Linz, Austria
juffi@faw.jku.at

***Abstract*—State-of-the-art game playing programs do not provide an explanation for their move choice beyond a numerical score for the best and alternative moves. However, human players want to understand the reasons why a certain move is considered to be the best. Saliency maps are a useful general technique for this purpose, because they allow visualizing relevant aspects of a given input to the produced output. While such saliency maps are commonly used in the field of image classification, their adaptation to chess engines like Stockfish or Leela has not yet seen much work. This paper takes one such approach, Specific and Relevant Feature Attribution (SARFA), which has previously been proposed for a variety of game settings, and analyzes it specifically for the game of chess. In particular, we investigate differences with respect to its use with different chess engines, to different types of positions (tactical vs. positional as well as middle-game vs. endgame), and point out some of the approach's down-sides. Ultimately, we also suggest and evaluate a few improvements of the basic algorithm that are able to address some of the found shortcomings.**

***Index Terms*—Explainable AI, Chess, Machine learning**

I. Introduction

Computer chess is one of the first and most prominent success stories in artificial intelligence [13], culminating in Deep Blue's victory over the reigning world chess champion [14]. In their second encounter, Gary Kasparov was unnerved by an unexpected move of the program, which he thought was too human-like. As a result, in an attempt to understand how the program arrived at that choice, he requested to see the printouts of the program. Of course, printing out the entire variation tree with millions of moves was infeasible for the Deep Blue team, and they denied the request. What could have helped here, is an algorithm for explaining the move choice, which goes beyond simply returning the numerical score of the move and its alternatives.

Nowadays, there is no doubt that the best chess engines are far stronger than any human chess players, and chess engines have also become an undispensable training tool for chess grandmasters. However, they have been constructed as black boxes, unable to explain their decision making processes, so that it still requires expert-level chess knowledge in order to interpret the moves made by leading chess engines [18]. A crucial step to bridge this gap between humans and AI is to provide useful explanations to proposed decision-making agents. While much of research in explainable AI is centered around machine learning [12] and the interpretation of deep neural networks [19], there is also great demand for explainable AI models in other areas of AI, including game playing.

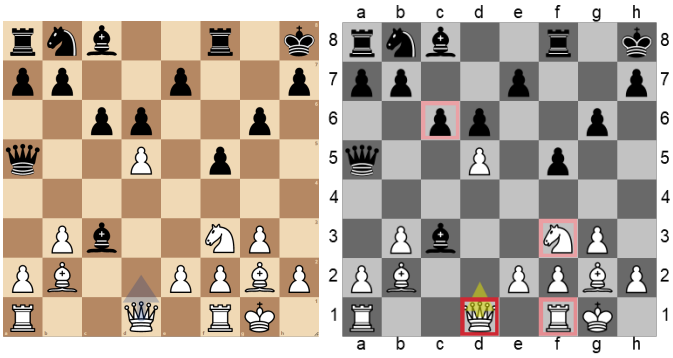


Fig. 1: SARFA Example

An approach that has recently been proposed specifically for games and related agent architectures is *specific and relevant feature attribution* (SARFA) [15].[1] It has been shown that it can assist human chess players in assessing important squares, which improved their success rates in solving chess puzzles [15]. However, SARFA being conceived as a general tool for assessing feature importance in black-box models, we also noticed that the approach has some weaknesses when it is applied specifically to chess. Consider, e.g., Figure 1, which is a position from the Bratko-Kopec test [5]. The Stockfish 11 engine played the excellent move ♕d2, which is only possible because the black bishop on c3 is pinned, and, moreover, it adds another pin on the bishop, which can now no longer take the white bishop on b2 without losing the queen on a5. The right part of the figure shows the output of SARFA, where the different shades of red illustrate the gradience in importance, with the darkest tone of red being the most important piece. It can be seen that none of the pieces mentioned above are considered to be salient. Moreover, by design, SARFA is also not able to recognize that the empty squares on the long diagonal are important (if there was a piece on of these squares, the pin would no longer work and black could take the white queen), Thus, another goal of this work will be to analyze such weaknesses, in particular with respect to positional and endgame play, and suggest and evaluate some simple improvements.

[1] Available on Github at https://github.com/nikaashpuri/sarfa-saliency.

978-1-6654-3886-5/21/$31.00 ©2021 IEEE

The goal of our work is to thoroughly evaluate SARFA for the game of chess. For example, in Section III, we will evaluate SARFA on eight chess engines of different playing strengths and styles. While the original analysis was confined to tactical positions, which can typically be very well solved by conventional, search-based chess engines, we will also take a look at its performance on strategic and endgame positions, which require a deeper chess understanding (Section IV). Finally, we also noticed a few limitations of SARFA (Section V), which we tried to address with some chess-specific modifications in Section VI. For example, SARFA originally only focussed on evaluating the importance of pieces on the board, whereas in many positions it is also important that certain squares are empty. All executed dataset runs, evaluation classifiers and the proposed adaptation files are publicly available.[2]

II. Specific and Relevant Feature Attribution

A particularly popular technique, when it comes to the interpretation of inscrutable black-box AI models, are *saliency maps*. Their key idea is to determine the relevance of the input features, with respect to the computed output. For example, layered relevance propagation (LRP) [10] propagates the output activation of a neural network backwards through the weighted connection in its layers, so that its distribution in the input layer eventually reflects the respective relevance of the input features. While such approaches are typically applied to image-based inputs and depend on the concrete architecture of the learned black-box model (typically a deep neural network), they can also be adapted for other input representations. An obvious idea is to perform random perturbations of the inputs, with the expectation that relevant features will result in large changes in the outcome, whereas a perturbation of irrelevant features will leave the output unchanged (cf., e.g., [16], [7]).

A recently proposed technique along these lines, *specific and relevant feature attribution* (SARFA) [15], has previously been analyzed for various application fields, including common Atari games such as Breakout or board games like chess. It relies on an evaluation function $Q(s, a)$ for a given move a in a position s, which can, e.g., be provided by a common chess engine, for estimating the importance of features in s. SARFA iterates over the chess board and perturbs its pieces by removing them one at a time, retrieving the Q-values for the resulting positions s', with the intention of monitoring the change of the evaluation in the perturbed state.

The key insight of [15] was that not the change $\Delta Q = Q(s, a) - Q(s', a)$ between s and a perturbation s' is relevant, as has, e.g., been proposed in [7], but instead the *relative* change has to be used. For example, if a generally important piece like the queen is removed from the board, ΔQ will change substantially, regardless of whether the position of the queen is important for the played move a or not. To capture such a relative change, SARFA computes a *specificity* score

$$\Delta P = P(s, a) - P(s', a) \tag{1}$$

[2]https://github.com/JessyFritz/sarfa-saliency

where $\mathbf{P}(s, A)$ is a softmax probability distribution over all moves A in a position s, based on the obtained evaluations $Q(s, a)$. A high ΔP score indicates that the piece, which has been removed when changing position s to s', was important for the evaluation of action a, because its relative importance in the two positions changed substantially.

While only pieces whose removal directly affects the investigated move receive a high specificity score ΔP, they should also have a minimal impact on the distribution of scores of the remaining moves, i.e., the softmax distribution $\mathbf{P}(s, A)$ should change as little as possible, as when moving from s to s'. To capture this, a *relevancy score*

$$K = \frac{1}{1 + (\mathbf{P}(s', A) \mid\mid \mathbf{P}(s, A))} \tag{2}$$

is calculated, where $||$ denotes the KL-divergence.

The *saliency* S of a piece is then estimated with the harmonic mean of the specificity ΔP and relevance K.

$$S = \frac{2K\Delta P}{K + \Delta P} \tag{3}$$

Pieces with a high saliency will therefore have a high impact on the evaluation of a move.

III. Testing different engines

This section aims at identifying differences in saliency maps generated by various chess engines.

A. Chess Engines Used

For our evaluation we selected several chess playing engines, with a wide variety of playing strengths and playing styles. Table I shows the tested engines, sorted by recent Elo ratings from the December 2020 Chess Engines Grand Tournament.[3] The most popular among these are probably the rivaling Stockfish and Leela engines, which have come forward with some recent advances in reinforcement learning. Especially the Stockfish 12 release, compared to its predecessor, made an impressive leap ahead by including a neural network into their already strong supervised learning approach. The previous version, 11, uses heuristic functions for evaluating positions, combined with a classic alpha-beta game tree search [4]. Like AlphaZero, Leela relies on a deep neural network paired with Monte Carlo tree search [2], and thus has a very different and more human-like playing style than conventional search-based chess engines [18].

[3]http://www.cegt.net/rating.htm

TABLE I: Selected Engines

Engine Version	Release Year	Elo Rating
Stockfish 12	2020	3578
Lc0 v0.26.3 - net J92-270	2020	3534
Stockfish 11	2020	3505
SlowChess Classic 2.4	2020	3275
Komodo 12.1.1	2018	2939
Rybka 2.3.2a	2020	2814
Octochess r5190	2013	2618
Fruit 2.2.1	2005	2569

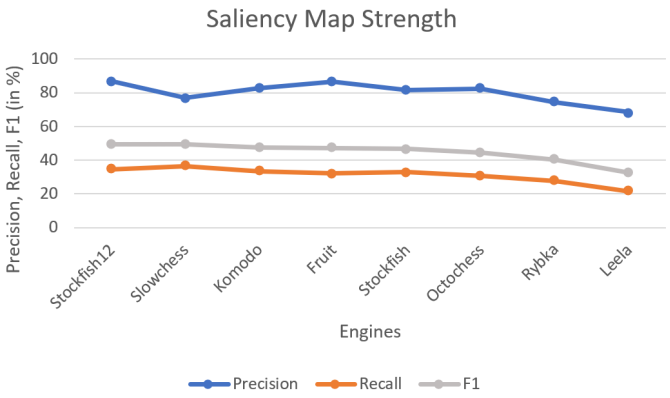


Fig. 2: Precision, recall, and F1 of saliency maps for different engines.

We use the default centipawn evaluation as the Q-value, which is the standard evaluation among basic chess engines. Once the engine is successfully initialized using the UCI[4], the program's main function can be called with a FEN[5] for a chess board as parameter, which then calculates the board's best move for a given amount of time, in our case 5 seconds. After that, the program perturbs every piece, each time retrieving a new set of Q-values in order to calculate the saliency (3).

B. Evaluation Measures and Datasets

We evaluated the engines in terms of precision and recall on a variety of different sets of chess positions, including the dataset that was used in the original SARFA paper [15], the Bratko-Kopec test [5], and a set of positional and strategic endgame positions [1], which will be analyzed in detail in later sections. For computing precision and recall, the marked-up labels need to be compared to ground truth labels for important squares. For obtaining these, we re-used the labels wherever available (e.g., from the SARFA datasets) and relied on the squares mentioned in move annotations by chess experts, supplemented by our own analyses[6]. The precision value describes the fraction of relevant squares among the retrieved squares, while recall describes the fraction of retrieved relevant squares among all squares annotated as relevant. To express saliency maps' accuracy in a single term, we will also use the F1-measure, which is the harmonic mean of precision and recall.

C. Results

Figure 2 shows the results averaged over all tested datasets, expressed by precision and recall and sorted by their F1-score. It is evident that some engines produce better output than others, as the F1 measure ranges from 50 % with Stockfish 12 to 33 % with Leela. These exceeding deviations across different engines saliency map computations also illustrate how little they are linked to their age or Elo rating. It may be true that this perturbation-based approach produces focused saliency maps, however, this does not guarantee that it works for every agent.

Particularly surprising is that the the neural-network based Leela engine, which is based on AlphaZero and can thus be expected to play more human-like chess than conventional chess engines [18], produces the worst saliency maps over all eight engines, despite having the second highest Elo rating. One could have assumed, that evaluations from a neural network based engine are simply more difficult to analyze and interpret, as they do not employ human-like heuristics to make its decisions like basic AI engines do. However, following this assumption, the evaluation above should have exposed similar problems for all reinforcement learning agents, which it clearly did not. It can be seen that SARFA worked far better for the Stockfish 12 engine, which uses an efficiently updatable neural network (NNUE) in combination with their classical human knowledge evaluations and plays significantly stronger than any of its predecessors [4]. However, of all applied agents, Leela is the only one that uses no human knowledge at all, so maybe this could be a clue for this weakness.

IV. Different Types of Chess Positions

In the following, we will use the SARFA saliency map implementation against several datasets to evaluate these engines based on the underlying chess positions, whether the program works best for tactical in comparison to strategic positions, or for endgame setups.

A. Bratko-Kopec Test

The first dataset, which will be used for assessing the quality of given engines, is the well-known Bratko-Kopec test,[7] which has been a standard in computer chess for nearly 20 years, as it equally balances two diverse kinds of puzzles, positional and tactical ones, and predicts your Elo rating based on your answers. The original experiment from Bratko and Kopec in the 1980s [5] hereby concluded that traditional search-based chess playing computer programs excel at tactical puzzles, where it is important to find a surprising variation that promises a short-term gain (such as a queen sacrifice, which ends in a forced mate a few moves later), and are weaker in positional puzzles, where the task is to estimate long-term strategic effects of moves. The positional and tactical profiles of humans, on the other hand, are more evenly distributed. The Bratko-Kopec test contains 12 puzzles of each kind and should correlate to the actual Elo ratings of humans and chess programs. Each puzzle has a unique solution, but the user can enter up to four ranked moves, which are scored with 1, 0.5, 0.33 or 0.25 points respectively, in case one of the inserted moves matches the given solution move.

Table II shows the results of the test for all 8 engines, where Leela has the highest score of all. Its score of 20.83 points means that it made the fewest mistakes, as its four

[4]The Universal Chess Interface (UCI) is a standardized interface for chess engines, cf. https://backscattering.de/chess/uci/.

[5]The Forsyth-Edwards Notation (FEN) is a compact representation code for chess positions, cf. https://www.chessgames.com/fenhelp.html.

[6]The second author is a chess expert (Elo rating approx. 2100). Of course, it was ensured that the annotation was performed independently of the program development, which was driven by the first author.

[7]http://www.kopecchess.com/bratko-kopec-test/

TABLE II: Bratko-Kopec Test Scores

Engine	Total Score	Positional	Tactical
Leela	20.83	8.83	12.0
Octochess	19.58	7.58	12.0
Stockfish 12	19.25	8.25	11.0
Fruit	19.00	7.50	11.5
Stockfish 11	18.83	8.33	10.5
Rybka	18.16	7.66	10.5
Komodo	18.00	8.50	9.5
SlowChess	18.00	9.00	9.0

best moves did not contain the solution only 4 out of 24 times. All other engines proposed wrong moves 5, 6 or even 7 times, which was the case for Stockfish, Rybka and Komodo. Considering that Stockfish 12 has the highest Elo of all tested engines, it would have been logical that it achieves the highest score, but interestingly, it is behind Octochess, which has the second weakest Elo rating. If we take a look at positional and tactical puzzles separately, the table depicts that both Leela and Octochess managed to list all 12 solution moves for the tactical puzzles, whereas both Stockfish engines missed either one or two of those moves. Generally, except for Slowchess, the score for tactical puzzles is always higher.

For estimating the precision, recall, and F1 for these positions, we manually analyzed all 24 puzzles' best moves, identifying some of the most important squares as the ground truth that should be uncovered by the saliency maps. Figure 3 shows the results of the test for all 8 engines, separated into positional and tactical puzzles for our ground-truth analysis. Generally, except for Stockfish 12 and Leela, it can be seen that the precision for positional puzzles is always higher than the one for tactical ones, and that, in general, precision is much higher than recall. This, however, is a bit deceptive, as for some of the engines, the predictions are very conservative. For example, Octochess', Fruit's and Rybka's positional saliency maps show a very high precision of 100%, but a recall below 15%, because they all only mark exactly one piece per board, which is the starting square of the solution move. The average number of marked squares for positional puzzles over all engines is 2.31 squares per board, where Stockfish 12 offers the highest average of 5.29 squares. Similarly, the average over tactical puzzles over all engines is 3.38, so more than one square higher per board than at positional puzzles, where Stockfish 12 again marks the most squares per board. If we take a look at the calculated F1-measure, Stockfish 12 clearly produces the best saliency maps for the Bratko-Kopec test, followed by the slightly older, other Stockfish engine.

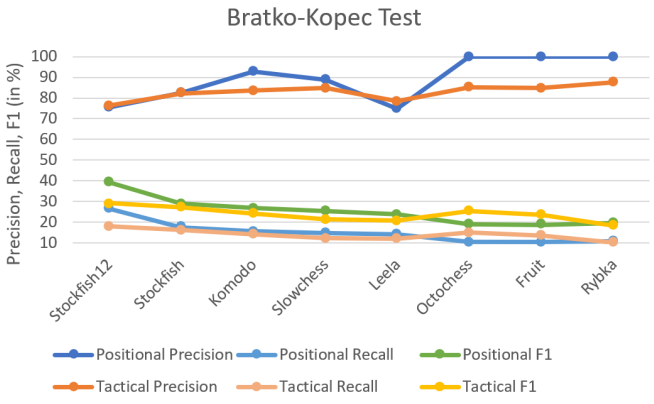


Fig. 3: Bratko-Kopec Test Saliency Maps

Overall, it can be said that the SARFA approach works slightly better for positional than for tactical puzzles, as can, e.g., be seen from the positional average score over all eight engines, which is somewhat higher for positional (25.1%) than for tactical puzzles (23.7%). This is surprising, as chess engines are typically better in finding tactical moves than positional moves, which is also witnessed by the scores the engines obtained at the puzzles (Table II). Apparently, the estimation of the influence of a piece on the board on the final evaluation of the position is fairly independent of the type of the position, and might even work better in quiet, non-tactical positions.

B. *Endgame Puzzles*

For testing the performance of SARFA on endgames, we used 10 endgame puzzles from Sune Larsson [1]. His test suite does not provide a solution move for every puzzle, yet it mentions its expected result as win, draw, or loss for every puzzle and even states the difficulty of some hard to find moves. For the cases where the author did provide a solution move, we analyzed the moves qualitatively and added ground-truth squares, which should be highlighted by the engines. Here, the number of squares marked as relevant was typically higher than in other puzzles, as we, e.g., often had to mark entire blocked pawn formations with adjacent empty squares.

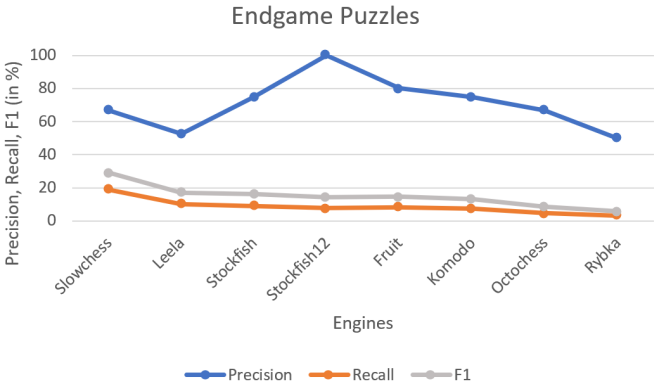
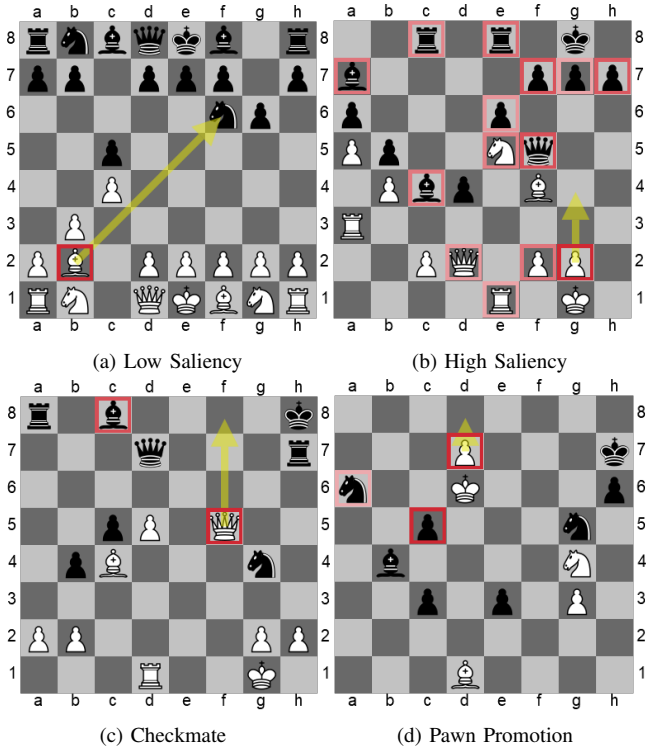


Fig. 4: Endgame Puzzles Saliency Maps

Figure 4 illustrates that Slowchess has the highest F1-measure, which it achieved through its recall fraction of approximately 19 %. Rybka is last, which only marked one true positive square and thus has a recall below 5 %.

V. Limitations of SARFA

In this section, we will list and analyze some shortcomings of SARFA, which we noticed in our experiments. We primarily focus on Stockfish 12, as this agent proved to produce some of the most promising saliency maps.



(a) Low Saliency (b) High Saliency

(c) Checkmate (d) Pawn Promotion

Fig. 5: Saliency Maps

A. Overly Specific and Overly General Saliency Maps

Figure 5 gives an overview of some unsatisfactory saliency maps on different types of chess boards, whether they are focused too much or too little, or are simply no help for inexperienced chess players at identifying relevant pieces in certain positions. In Figure 5 (a), one can see a map in which the engine did not consider a single additional square as salient, except for the performed best move's starting square b2. Obviously, this saliency map is too simple and fails at its intended task to gain insight into the engine's playing behavior. Conversely, the next map, (b), highlights too many squares than presumably desired, which is also not transparent or helpful. Given this inconsistency of highlighting many irrelevant squares at once, whereas in other maps only one square is salient, a good goal could be to avoid both these kinds of maps by walking the line between not marking anything and the whole board.

B. Empty and Blocked Squares

The following two Figures 5 (c) and (d) show two cases that are more a matter of opinion than really a flaw. This played check by the white queen on map (c) could be considered salient by highlighting the black king and maybe even its neighboring rook, as it blocks the king's escape, ultimately leading to a checkmate. The map only marks the black bishop on c8, which is important to the played move, as it blocks the white queen from being taken by the black rook on a8. However, the value of the move would clearly be diminished if the black rook would not block the black king's exit square, so this should be marked as important as well. The next map, (d), shows a pawn promotion d7-d8♕, where it might be useful to highlight the empty square d8, with promoting the pawn being most likely the only reason for this move. Regarding this issue, the SARFA authors stated that their implementation currently cannot highlight the importance of absence of certain attributes, i.e., the saliency of empty squares [15]. Taking the chess saliency map's purpose of gaining insight into the playing behavior of an engine into aspect, it could be meaningful for the interpretation to extend this implementation to consider empty squares as salient, as they certainly can play a crucial role for an intended move.

C. Strategic Positions

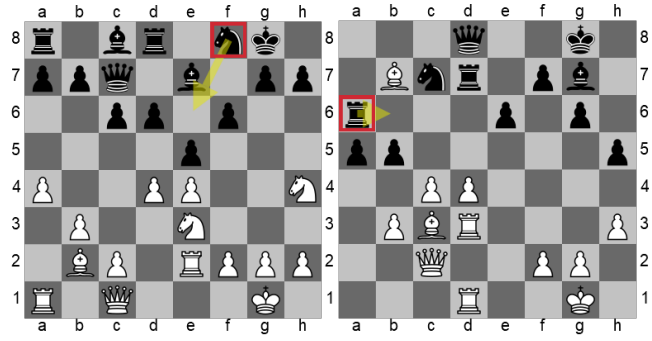
The original SARFA study [15] was limited to tactical positions, where the effect of the presence or absence of a piece will have a more pronounced effect. Nevertheless, SARFA also performed well on the strategical positions of the Bratko-Kopec test (Figure 8). In this section, we take a closer look at some strategic positions, using saliency maps produced by Stockfish 11, Stockfish 12 and Komodo.

Figure 6 shows some saliency maps on additional strategic positions. Obviously, they all have in common that they essentially only highlight the moving piece, but do not provide any help for understanding the strategical patterns involved. The first one, Figure 6 (a), shows a positional puzzle that is all about creating weak squares in the course of the game. After playing ♘e6, the following moves should be d4×e5, d6×e5, ♘f5 and ♗b4, after which White dominates white squares [3]. The second puzzle (b), produced by Stockfish, shows that a small material advantage can create a positional plus for the opponent. In the actual game, White sacrificed the exchange against for a pawn (♖b6, c4-c5, ♖×b7, c5-c6, ♖a7, c6×♖d7, ♕×d7) [3]. Interestingly, the engine did not even mark the white bishop on b7 as salient for the move ♖b6, although it is both the reason why the rook moved (the bishop attacked the rook) as well as the reason why the rook moved to b6 (the rook attacks the bishop). In position (c), White played ♗b5, with the intention of controlling the hole on c6, truly a salient square. This puzzle shows that it is good to embark upon the weak points of the opponent, in this case leading to material win after the line ♗b7, ♗×♘f6, ♗×♗f6, ♕a4, ♖e7 and, finally, ♗c6 [3]. The last puzzle, (d), is not ideal for black because of the uncoordinated black pieces. This can be partly resolved by trading bishops on g5, and then following up with ♘e7, making up space for the other pieces. Again, this should be considered important for this move.[8]

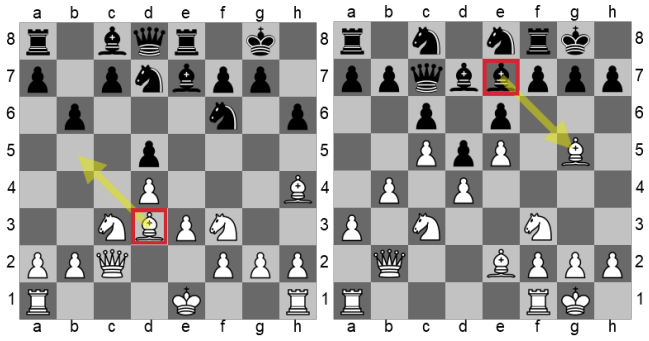
VI. Chess Specific Improvements

Section V pointed out some limitations of SARFA, which we will aim to improve in this section by adapting the original SARFA with some more chess-specific aspects. We will illustrate these improvements with the same puzzles as in Figure 6, and show the updated maps in Figure 7. A more principled evaluation can eventually be found in Section VI-C.

[8]https://chesstempo.com/positional-motifs

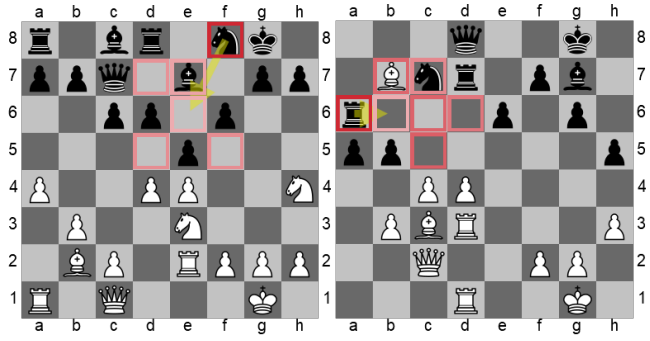


(a) Weak Squares (b) Small Advantage

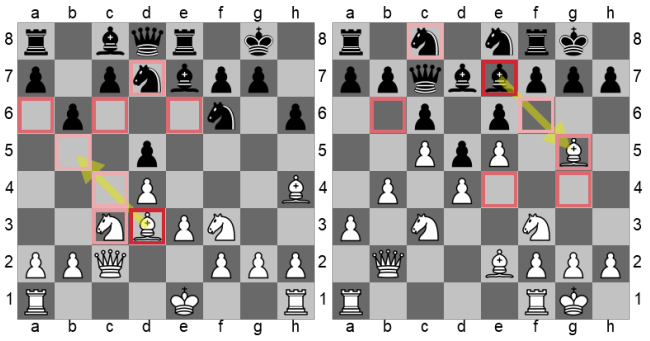


(c) Weak Point (d) Offer Exchange

Fig. 6: Strategic Chess Puzzles (SARFA)



(a) Weak Squares (b) Small Advantage



(c) Weak Point (d) Offer Exchange

Fig. 7: Strategic Chess Puzzles (Improved)

A. *Empty Squares*

First, we tried to solve the problem that empty squares often appear to be relevant but are not considered in the original SARFA version. To that end, we perturbed empty squares between ranks 2 to 7 as follows:

(i) Place a pawn of the player on the empty square, unless it would put the opponent's king into check.
(ii) Compute new Q-values and, in consequence, saliency scores for the resulting position.
(iii) Repeat (i) and (ii) with a pawn of the opponent's color.
(iv) Estimate the saliency of the empty square with the maximum of both saliencies.

In addition, we generally also consider the empty squares that are necessary for the played move as important. A fixed saliency that corresponds to the set saliency threshold has been hard-coded to these squares, such that they would be just important enough to appear in the heat map.

B. *Important Pieces*

The next feature targets increasing the saliency of pieces, based on the board's specific situation.

a) Promoting Square: In pawn promotions, typically not many pieces are salient, except the promoting pawn itself. In such cases, we increase the saliency of the destination square to 1 (as, e.g., for the square d8 in Figure 5d). The promotion puzzle now additionally highlights the promoting square d8, as well as the Knight on g5, which will be threatened after promoting to a queen.

b) King in Check: A simple, but important chess-specific modification is that the purpose of a move in chess is often to deliver a check. Here, the original implementation always assigns a saliency of 0, as the king can't be removed or replaced for perturbation. However, this is problematic, as any board where the king makes a move will mark no squares at all, because the threshold will also be calculated as 0. Nonetheless, in particular in endgames, the king is an important piece. In a case of a check, the king, as well as the square from which the check is delivered, are assigned a saliency of 1. The checkmate map from Figure 5c adds the king, the squares included in the best move and opponent pieces that are threatened by the queen after her move.

c) Threatened and Guarding Pieces: It is really interesting that the map in Figure 6 (b) did not mark the bishop on b7 as salient, although the rook, which is on b6 after his move, directly threatens this completely unguarded piece. As such pieces, which could be taken by the piece of the original action after its move, could be important, an increment of these pieces' saliency through exploring the new available moves after actually making the best move seems appropriate. Additionally, this is now also applied for the opponent, such that a piece's saliency, which threatens the original action on its destination square, is increased as well. Also, we increased the saliency of pieces which protect the best move at its destination square.

d) Blocked Pieces: As seen in Figure 6 (d), sometimes certain pieces are blocked and have to be freed by moving other pieces in order to create an escape path for the piece in need. In the given map, this is the case for the knight on c8, which has clearly no way to go, without being taken by the pawn on c5. This puzzle's move ♗g5 frees the square

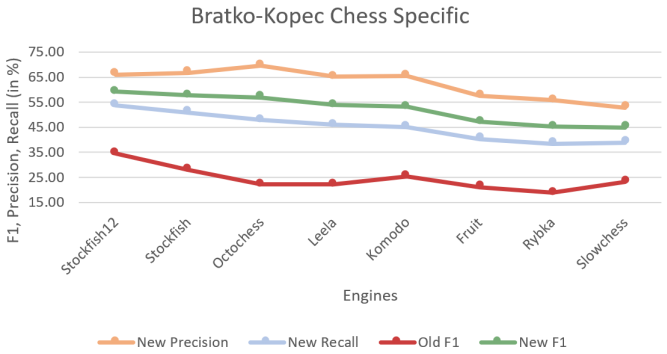


Fig. 8: Improved Bratko-Kopec Test Saliency Maps

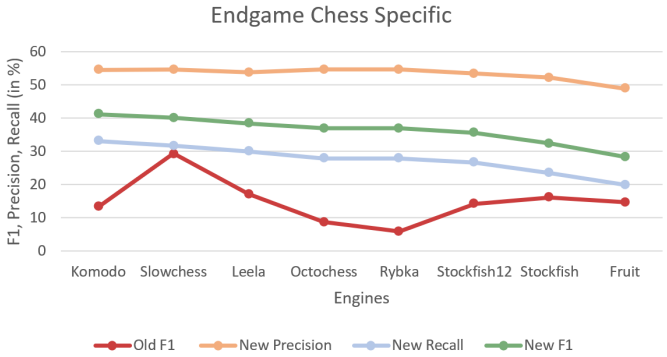


Fig. 9: Improved Endgame Dataset Saliency Maps

e7 for the important knight. For these pieces, the program calculates the difference in legal moves before and after the best move and examines if there were new moves for pieces, other than the piece of the best move, available. If this is the case, it iterates the previous moves with the destination as that piece and checks whether all of them were of opponent's color, meaning this piece was trapped before this positional move.

C. Evaluation of Improvements

In order to evaluate the suggested improvements, we re-ran the datasets from Section IV to evaluate how the proposed changes influence the overall saliency map generation for different engines. Regarding the empty squares feature, as for some maps the implementation would still mark way too many empty squares, we limited the overall number of empty squares, which are not included in the best move, to 3.

If we take a look at the reiterated Bratko-Kopec test seen in Figure 8, we perceive a considerable improvement over the original SARFA version (Figure 3): now, no F1 value is below 44%, whereas previously all were below 35%. Quite similar, Figure 9 shows an improvement in the average from 15% in Figure 4 to 36% here.

In Figure 10, we also evaluated the sensitivity of the F1 scores, averaged over all engines, to the specific choice of the number of marked squares. For the Bratko-Kopec test, three turned out to be the actual optimal choice, whereas in the endgames, higher values tend to score better. This is not surprising, because, as previously noted, there the number of labeled empty squares tends to be larger than in other positions, in particular in tactical positions.

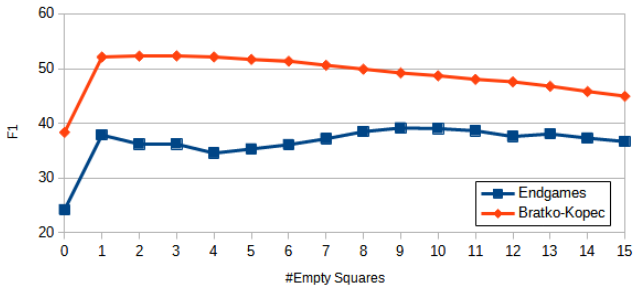


Fig. 10: F1 over number of empty squares, average of all engines.

In summary, the suggested chess-specific improvements worked better for both the Bratko-Kopec test and the endgame dataset with an overall improvement by 20 % on average. This also resulted in more evenly distributed evaluations for all engines, which highlights that all agents' saliency maps now produce useful and interpretable outputs.

VII. Related Work

Like in many other areas, the demand for interpretable explanations has also risen in game playing, and in chess in particular. The idea of using a chess engine's evaluation function to detect changes in the position evaluation when a move is made is not new. The chess tutoring system described in [17] classified available moves as good or bad based on changes in commonly observed positional features such as king safety or knight centralization. The main drawback and distinction of this method compared to SARFA is that one needs actual access to the engine's inner working in the form of the evaluation function features. SARFA is independent of hand-crafted evaluation function features, as it only requires the overall numerical evaluation of all moves.

Some prior work in this area focused on the use of natural language processing techniques for generating game commentary. The game-aware commentaries introduced in [8] train an end-to-end neural model to generate natural language commentaries from a dataset of move/commentary pairs. It also relies on high-level features such as attacked pieces before and after the move, the change in score, etc. Along similar lines, the skilled chess commentator [20] jointly trains a neural network engine and a comment generation model for different types of natural language comments. [9] introduces a rule-based commentary generator for the chess variant Shogi that predicts words of input positions and generates comments using a language model trained on human expert comments.

More related to our pursuit is work on visualizing chess games. Recently, [6] introduced a technique for visualizing sets of chess games by bundling common move trajectories in a low-dimensional embedding of chess positions. Similarly, the authors of [11] introduced a tree-based visualization of a move and interesting alternatives. None of these techniques aims at visualizing the significant patters associated with the played move, as SARFA does.

VIII. Conclusions

Regarding the interpretability gap between humans and AI, it is quite obvious where the problem of chess engines lies. Without doubt, engine moves are correct or at least better than human choices, but it is hard to extract a concrete, simple justification for a move in the thicket of variations that the program evaluates. Modern neural network based engines like AlphaZero can not only come up with short-term tactical shots, but have also become notorious for playing long-term piece sacrifices, which can not be justified by a variation tree, but are rooted in the gain of long-term positional advantages [18]. Simple explanations, such as the numeric evaluation of the best k moves in the current position, which are now routinely provided by conventional chess engines, do not give any insight on which pieces on the board are important and led to the observed evaluation score.

SARFA is a general method for analyzing the importance of features by observing the effect of input perturbations on the output, which we have, in this work, specifically analyzed for the game of chess. Our results showed that not all chess engines produce useful saliency maps, as witnessed by a variation in the average F-score of 17% between the best and worst engine's maps. We also found that higher playing strength does not necessarily lead to improved saliency maps. Surprisingly, Leela, a very strong neural-network based engine, performed rather poorly at the saliency map generation. On the basis of these experiments, we furthermore extended the functionalities of SARFA by including a few straight-forward, chess-specific modifications, such as an approach for perturbing the empty squares, or saliency increments for chess-specific scenarios such as checks or promotions. Our results demonstrated that these techniques can indeed increase SARFA's performance on the studied datasets.

Nevertheless, this improvement is currently limited by the decrease in precision that naturally comes with marking more squares by increasing their saliency scores. Future work could include a refinement of the empty square perturbation strategy, which currently had to be cut off at a fixed number instead of a pre-set threshold, because otherwise too many squares would be marked as empty. Additionally, one could devise different strategies for perturbing empty squares, which currently only tried to replace with pawns of either color. This proved to be particularly useful in endgame studies, where the position of the pieces (in particular pawns) is often very static. However, we did not include perturbations that place chess pieces different than pawns that, for example, replace higher value pieces by lower ones to capture the difference in importance.

A next step ahead should, in our opinion, be to move from annotating pieces (as in the original SARFA) or empty squares (as in our version), to annotating moves. Many of the empty squares in a position capture whether a certain move or even a sequence of moves are possible in a given position. A more direct way of capturing the saliency of moves could further bridge the interpretability gap between humans and programs, for which we believe SARFA is a valuable foundation.

References

[1] Endgame puzzles. [Online]. Available: https://www.stmintz.com/ccc/index.php?id=476109

[2] Leela chess. [Online]. Available: https://github.com/LeelaChessZero/lc0/wiki

[3] Positional chess puzzles. [Online]. Available: https://www.chess.com/article/view/test-your-positional-chess

[4] Stockfish chess. [Online]. Available: https://stockfishchess.org/

[5] I. Bratko and D. Kopec, "The Bratko-Kopec experiment: A comparison of human and computer performance in chess," in *Advances in Computer Chess*. Pergamon Chess Series, 1982, pp. 57–72.

[6] A. P. Hinterreiter, C. A. Steinparz, M. Schöfl, H. Stitz, and M. Streit, "Exploring visual patterns in projected human and machine decision-making paths," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2021, Special Issue on Interactive Visual Analytics for Making Explainable and Accountable Decisions.

[7] R. Iyer, Y. Li, H. Li, M. Lewis, R. Sundar, and K. P. Sycara, "Transparency and explanation in deep reinforcement learning neural networks," in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, J. Furman, G. E. Marchant, H. Price, and F. Rossi, Eds. New Orleans, LA, USA: ACM, 2018, pp. 144–150.

[8] H. Jhamtani, V. Gangal, E. Hovy, G. Neubig, and T. Berg-Kirkpatrick, "Learning to generate move-by-move commentary for chess games from large-scale social forum data," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Melbourne, Australia: ACL, 2018, pp. 1661–1671.

[9] H. Kameko, S. Mori, and Y. Tsuruoka, "Learning a game commentary generator with grounded move expressions," in *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, 2015, pp. 177–184.

[10] S. Lapuschkin, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS ONE*, vol. 10, 07 2015.

[11] W. Lu, Y. Wang, and W. Lin, "Chess evolution visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 5, pp. 702–713, 2014.

[12] C. Molnar, *Interpretable Machine Learning*, 2020. [Online]. Available: https://christophm.github.io/interpretable-ml-book/

[13] K. Müller and J. Schaeffer, *Man vs. Machine: Challenging Human Supremacy at Chess*. Russell Enterprises Inc., 2018.

[14] M. Newborn, *"Kasparov Vs. Deep Blue: Computer Chess Comes of Age"*. Berlin, Heidelberg: Springer-Verlag, 1997.

[15] G. Piyush, P. Nikaash, V. Sukriti, K. Dhruv, D. Shripad, K. Balaji, and S. Sameer, "Explain your move: Understanding agent actions using specific and relevant feature attribution," in *International Conference on Learning Representations (ICLR)*, 2020.

[16] M. Robnik-Šikonja and M. Bohanec, "Perturbation-based explanations of prediction models," in *Human and Machine Learning: Visible, Explainable, Trustworthy and Transparent*, J. Zhou and F. Chen, Eds. Springer International Publishing, 2018, pp. 159–175.

[17] A. Sadikov, M. Možina, M. Guid, J. Krivec, and I. Bratko, "Automated chess tutor," in *Computers and Games*, H. J. van den Herik, P. Ciancarini, and H. H. L. M. J. Donkers, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 13–25.

[18] M. Sadler and N. Regan, *Game Changer: AlphaZero's Groundbreaking Chess Strategies and the Promise of AI*. New in Chess, 2019.

[19] W. Samek, G. Montavon, A. Vedaldi, L. Hansen, and K.-R. Müller, Eds., *"Explainable AI: Interpreting, Explaining and Visualizing Deep Learning"*. Springer, 2019, vol. 11700.

[20] H. Zang, Z. Yu, and X. Wan, "Automated chess commentator powered by neural chess engine," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 5952–5961.

Acknowledgments: We are very grateful to the authors of SARFA [15] for sharing their code, in particular to Nikaash Puri who also provided helpful comments on our experiments. Thanks also are also due to the anonymous reviewers, whose comments helped us considerably to stream-line the paper.