Unveiling Concepts Learned by a World-Class Chess-Playing Agent

Aðalsteinn Pálsson , Yngvi Björnsson
Department of Computer Science, Reykjavik University
{adalsteinn19, yngvi}@ru.is

Abstract

In recent years, the state-of-the-art agents for playing abstract board games, like chess and others, have moved from using intricate hand-crafted models for evaluating the merits of individual game states toward using neural networks (NNs). This development has eased the encapsulation of the relevant domain-specific knowledge and resulted in much-improved playing strength. However, this has come at the cost of making the resulting models ill-interpretable and challenging to understand and use for enhancing human knowledge. Using a world-class superhuman-strength chess-playing engine as our testbed, we show how recent model probing interpretability techniques can shed light on concepts learned by the engine's NN. Furthermore, to gain additional insight, we contrast the game-state evaluations of the NN to that of its counterpart hand-crafted evaluation model and identify and explain some of the main differences.

1 Introduction

Game-playing agents for abstract board games, like chess and checkers (and others), almost universally employ both a search and an evaluation components for coming up with their move decisions. The former encapsulates the thinking ahead process, exploring various possible continuations of play, whereas the latter determines the merit of individual game states explored during the search. Traditionally, the evaluation component is a carefully hand-crafted (possibly automatically tuned) function modeling the domain-specific aspects of the game, e.g., for chess, concepts like material, development, king safety, and soundness of pawn structures. The evaluation function approximates and maps those disparate concepts into a single numerical value indicating how desirable a given position is from the perspective of the side having the move (e.g., the expected game outcome given correct play by both sides).

One of the most laborious tasks when making state-of-the-art game-playing agents is developing and carefully tuning such an evaluation function. However, recent successes using evaluation functions based on (deep) neural network models (DNNs), which are learned automatically, have eased this task considerably and, more impressively, improved gameplay considerably. The most notable examples of this approach are the superhuman agents Alpha-Zero [Silver *et al.*, 2018] (for chess, Shogi, and Go), Leela Chess Zero [LeelaChessZero, 2022] (chess) and Stockfish [Stockfish, 2022c] (chess). However, such an approach comes at a cost: the learned evaluation function is not easily interpretable. For example, the agent might prefer the position for one side, however, it might be unclear for a human observer, even an expert-level one, why that is the case. Moreover, the agent has no trivial ways of explaining its preference in human terms.

For a model to be interpretable, humans should readily understand the reasoning behind its decisions. NNs are notoriously difficult for humans to interpret and are often treated as black boxes, that is, concealed functions with inputs and outputs. Such a treatment is generally not desirable because we humans may need to understand the knowledge encoded into such models, e.g., to learn and build trust.

In this work, we try to understand better the knowledge encoded in NNs used by super-human strength game-playing agents, using chess as our test-bed. More specifically, we use state-of-the-art interpretability techniques for probing and interpreting the NN model used by Stockfish, (arguably) the strongest chess-playing engine in existence. Our objectives are to identify in human understandable terms what chess concepts the networks learned and what importance it places on each of them. Furthermore, we examine how the NN's position evaluation differs from its hand-crafted counterparts, thus identifying (at least partially) influential chess concepts accountable for improved playing strength. The paper's primary contributions are: (i) we show how state-of-the-art interpretability methods can gain insights into high-level concepts learned by a world-class game-playing agent, and (ii) we use the gained insights to pinpoint the relative strengths and weaknesses of neural-network and handcrafted models. For example, we show the network's capability to statically detect certain threats. Finally, this work adds to the emerging literature on explaining models learned by game-playing agents (e.g., [Puri *et al.*, 2020; Pálsson and Björnsson, 2021; McGrath *et al.*, 2021; McGrath *et al.*, 2022]).

The paper's organization is as follows. The next section introduces the terminology and background, followed by our methods and empirical result sections, respectively, and finally, we conclude and discuss future work.

2 Background

This section provides a brief background of model interpretability and how such methods have hitherto been used to interpret game-playing agents' actions. It furthermore introduces Stockfish, our test-bed game-playing agent.

2.1 Interpretability

Interpretability of neural networks has received much-added attention in recent years due to the popularity of DNNs and the increasing use of ML in serious situations (see e.g. [Bodria *et al.*, 2021; Mi *et al.*, 2020] for a survey). Interpretability of (black-box) models may be broadly categorized as either *global* or *local* and *model-specific* or *model-agnostic*. Global methods create interpretations valid across all input instances, whereas local methods' focus is on interpreting individual instances. Model-agnostic methods explain any black-box models, while model-specific methods leverage the model's architecture (thus, in reality, not treating the model as an absolute black-box). Our focus will mostly be on global methods, both model-specific and model-agnostic.

Most prior work on local methods for interpreting models use feature-based explanations, which alter the input features (e.g., occlude or perturb them) [Lundberg and Lee, 2017; Ribeiro *et al.*, 2016]. Such approaches are especially helpful in image-based domains and can reveal how different regions in an image contribute to the network's classification. Similar approaches have been used to explain chess positions [Puri *et al.*, 2020]. However, the methods are not easily applicable to intricate concepts. Thus, to overcome this shortcoming, more recent explanation techniques, referred to as *concept-based*, use high-level human concepts as interpretable units [Alain and Bengio, 2017; Kim *et al.*, 2018; McGrath *et al.*, 2021].

A popular concept-based model-specific interpretability methodology for "peeking" into DNNs is *probing* [Alain and Bengio, 2017]. Based on the intuition that deep neural networks are primarily about distilling computationally useful representations, one can monitor the output of different layers within the network for how well they represent various (high-level) concepts. One can measure how much information a layer carries for a given concept by training classifiers or regression models (on a dataset separate from the one used for training the network) to predict a given concept from a layer's activations; these models are called *probes*. The higher the prediction accuracy of the probe, the more information that layer carries for representing the concept.

2.2 Interpretability and Games

Research into intelligent game-playing agents has mainly focused on new algorithms and learning techniques for improved playing strength, with little attention to their capability for explaining the reasoning behind their actions.

Early work on intelligent chess-tutoring systems is scant and somewhat preliminary [HaCohen-Kerner, 1994; HaCohen-Kerner, 1995; Guid *et al.*, 2013; Sadikov *et al.*, 2006] and limited follow-up work. More recent work has instead focused on the interpretability of models, particularly neural networks, e.g., using saliency maps [Pálsson and Björnsson, 2021; Puri *et al.*, 2020] or concept probing [McGrath *et al.*, 2022; Lovering *et al.*, 2022].

The concept-probing work reported in [McGrath *et al.*, 2021; McGrath *et al.*, 2022], which attempts to explain the concepts learned by AlphaZero's neural network, is particularly relevant. As a proxy for human-understandable concepts, it uses, among others, the Stockfish's classical concepts. Using concept probing, they show that many human-understandable concepts get represented by the network, and since it is trained using self-play, the order of the *discovery* of these concepts is particularly interesting. They demonstrate that material concepts are represented well early in the training process, while more complex and subtle ones emerge later. Also, treating AlphaZero as a black-box, they show how concepts such as piece values and Stockfish's classical concepts relate to the output. By training a linear surrogate model predicting AlphaZero's output from a set of concepts, they show the relative importance of the concepts to one another.

Online sites for playing and looking at chess games do many provide the option to have computers analyze one's games; however, this is first and foremost in the form of a computer engine analysis simply pinpointing mistakes based on the engine's numerical evaluation. One notable exception to this is DecodeChess [Decodea, 2022], which explains using human-understandable chess concepts; however, the level of the explanations is still somewhat rudimentary. Moreover, the techniques used are proprietary and not published.

2.3 The Stockfish Game-Playing Agent

Stockfish [Stockfish, 2022c] is a free and open-source chess engine written in C++ and is available for various computing platforms. Today's top chess-playing engines all play at a super-human strength, and, historically, Stockfish has been the most victorious, with the most recent version leading most independent rating lists. In contrast to earlier versions, which use a hand-crafted evaluation function, the more recent versions of the engine can employ either a NN or a hand-crafted evaluation. Using the NN significantly improves playing strength even though it slows down the thinking-a-head search slightly. Stockfish, being open-source, state-of-the-art, and allowing both model-types of evaluation, is thus an ideal candidate to use for exploring interpretability and contrasting hand-crafted and NN based evaluations.

3 Methods

Here, we first describe Stockfish's evaluation models, both the classical and the neural network. Then, we detail the interpretability methods used and how we applied them.

3.1 Classical Model

The classical evaluation function uses carefully hand-crafted higher-level concepts (also called features) that are linearly combined to form an evaluation, i.e.:

$$f_{classical}(s) = \sum_{i=1}^{N} w_i \times c_i(s) \tag{1}$$

Concept	Type	Weight
Material	material	1.0
Winnable	material/positional	1.0
Passed-pawns	positional	1.0
Imbalance	material	1.0
Mobility	positional	1.0
King-safety	positional	1.0
Threats	positional	1.0
Space	positional	1.0
Pawns	positional	1.0
Knights	positional	1.0
Bishops	positional	1.0
Rooks	positional	1.0
Queens	positional	1.0

Table 1: The high-level concepts of the classical evaluation model of Stockfish. They all compute using a *centi-pawn* (1/100 of a pawn's value) as its unit metric; however, the resulting values may be on a different scale (i.e. *Material* may result in values in the thousands, *King-safety* and *Passed-pawns* in the hundreds, and the others typically less. A positive value indicates an advantage for the player to move and a negative one advantage for the other player. The scales of the concept values are pre-tuned to avoid additional weighing — thus, all weights are 1.0 in the linear combination.

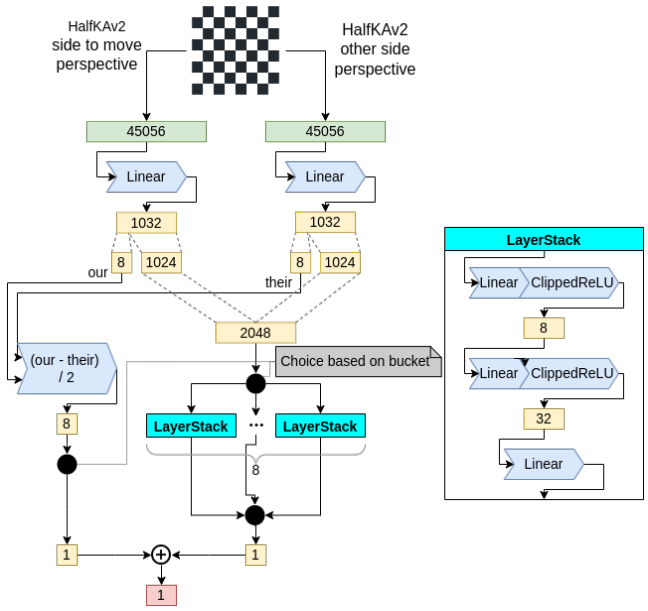

Figure 1: Stockfish' NNUE architecture [Stockfish, 2022b]. It processes the values depending on the game phase (8 buckets, depending on the number of pieces), i.e., it learns different PSQT values and uses a different layer stack (sub-network) for each bucket.

where s is the game state, N is the number of features, and c_i computes the value of concept i.[1]

As listed in Table 1, Stockfish's classical evaluation function uses several higher-level concepts. These concepts are computed for each game position and linearly combined to form the final evaluation (from the player's perspective having the move). The *Material*, *Imbalance*, and (in part) *Winnable* concepts are material based. *Material* accumulates the value of the pieces based on their type and location, *Imbalance* gives a bonus for specific piece configurations (most notably the bishop pair), and *Winnable* scales down the score for specific endgames that are known to be difficult to win (e.g., queen vs. rook and opposite color bishop endings). The remaining concepts are positional. The *Pawns*, *Knights*, *Bishops*, *Rooks*, and *Queens* features give bonuses to the respective piece types based on how good or bad a piece is in a given position. For example, the minor pieces (knights and bishops) get a bonus if on a good central outpost, and the major pieces (rooks and queens) get a bonus if on an open or a semi-open file. Pawns are penalized for weaknesses such as being isolated or doubled, which can be a serious weakness, especially in the endgame where pawns typically play a key role because of their ability to promote. The *Passed-pawns* concept aims at capturing the potential for pawns to promote in the endgame. The *Mobility* and *Space* concepts estimate in different ways how easily one can maneuver own forces on the board. The former uses the number of safe squares a piece can move to as an approximation of its mobility, whereas the latter estimates how much space (many squares) in the center is secure for our pieces. The *Threats* feature estimates potential threats in the position (e.g., attacks on the opponent's weak squares). *King-safety* explicitly handles threats against the king, which may be critical.

[1]More specifically, Stockfish uses a so-called phased-evaluation where it computes the value of each feature differently for the middle- and end-game, and then linearly weights the two evaluation based on the approximated game-phase (determined from the material on the board). For our intended purposes, this detail is unimportant and f_i represents the resulting phase-weighted value.

3.2 Stockfish' Neural Network

Stockfish (since version 12) uses a neural network called NNUE [Nasu, 2018] (EUNN Efficiently Updatable Neural Network) for evaluating game states. The network architecture was invented for the game Shogi but later ported to chess/Stockfish, immediately resulting in an 80 ELO point increase in playing strength (and more since then) [Stockfish, 2022a; Stockfish, 2022b].

The NNUE architecture uses a (shallow) design with linear and clipped ReLU layers, as depicted in Figure 1. Notable design choices are routing the inference through different layer stacks, or sub-networks, depending on the phase (number of pieces) of the game. Another interesting design choice is to feed piece-square-table-values (PSQT) directly to the output after a single linear layer.

3.3 Global Explanations

NNUE is a black-box in the context of explainability research. Although it is shallow, it is non-linear and does not fall in the category of interpretable models such as tree/rule-based or linear models. The interpretable models either learn more structured representations or enable tracing of causal relationships [Schwalbe and Finzel, 2021]. A general approach uses an interpretable model as a surrogate model to explain the black-box model's overall logic. This approach is also valid for explaining the local behavior, as done in the local

Concept	Description
*_bishop_pair	True if * has a bishop pair
*_knight_pair	True if * has a knight pair
*_double_pawn	True if * has a bishop pair
*_isolated_pawns	True if * has isolated pawns
*_connected_rooks	True if * has connected rooks
*_has_control_of_open_file	True if * has control of open file
has_contested_open_file	True if there is a contested open file
#_queens	The difference in number of queens
#_pawns	The difference in number of pawns
#_rooks	The difference in number of rooks
#_knights	The difference in number of knights
#_bishops	The difference in number of bishops

Table 2: The custom concepts (* stands for white or black)

surrogate approach LIME [Ribeiro *et al.*, 2016]. The surrogate model is supposed to mimic the decisions of the black-box model but transparently and should be judged on its fidelity [Bodria *et al.*, 2021], i.e. how well the surrogate model explains the black-box model.

We use a linear surrogate model mainly because it is intrinsic in the design of the classical concepts that their sum equals the classical evaluation. The linear model used minimizes the residual sum of squares (Ordinary Least Squares). However, it is non-trivial to choose the appropriate loss, e.g., [McGrath *et al.*, 2021] chose to minimize the L_1 instead of the L_2 loss, because they found the L_2 loss to systematically underestimate the piece weights. We tried both loss functions, and although the latter resulted in slightly lower piece values, they were both in proximity to the literature, and the dynamic between both evaluation methods remained the same.

To evaluate the importance of the concepts in Stockfish's two evaluation models we estimate the Shapley value of each concept. Shapley value evaluates the contribution of each concept over all possible combinations of concepts [Lipovetsky and Conklin, 2001]. In this research we use Shapley value sampling [Castro *et al.*, 2009] to evaluate the Shapley values. The metric for contribution is measured in r^2 accuracy.

3.4 Concept Probing

Concept probing aims to determine the emergence of human concepts in deep learning models [Alain and Bengio, 2017]. It is a global method used to shed light on how well the concept is represented in the network's activation, z, but not to explain individual samples. If we suspect that the concept information is linearly separable from negative samples (where the concept is not present), we can train models such as

$$\begin{aligned} g_l^j(z^l) &= w_{jl}^T z^l + b_{jl} \qquad \text{(continuous concepts)} \\ g_l^j(z^l) &= \sigma(w_{jl}^T z^l + b_{jl}) \qquad \text{(binary concepts)} \end{aligned} \tag{2}$$

Then, the accuracy of $g_l^j(z^l)$ on the held-out test set will indicate how much information the activations at layer l carry regarding the concept j. In [McGrath *et al.*, 2021] they mention the challenge of choosing the correct architecture for concept probing. For such a small network as Stockfish, new challenges arise where the information is so compressed that it is hard to linearly separate the concept from negative (random) samples. Furthermore, we use the high-level concepts defined in the Stockfish hand-crafted evaluation as our explaining vocabulary, as well as some lower-level binary concepts such as whether pawns are doubled or isolated and rooks are connected or not.

4 Results

In the following subsections, we describe the results gathered. The goal is to understand better the performance increase gained by introducing Stockfish's NNUE evaluation function. We begin by describing the experimental setup, and then in the second subsection, we use model agnostic methods to explain Stockfish's NNUE evaluation without analyzing the model's internals. In the third subsection, we analyze the internal representation of the model. Here we aim to determine how well Stockfish's NNUE model represents given human-understandable concepts. The fourth and final subsection highlights some of the main differences between the classical and NNUE evaluations.

4.1 Experimental Setup

We use version 14.1 of Stockfish, which was the latest release at the time of the research.

Our experiments need an external dataset to generate the concept probes. For that we use a dataset generated by *Leela Chess Zero* that is listed as a quality dataset (*training_data* at [Stockfish, 2022d]), from which we randomly sampled 100k positions. Henceforth, we defer to this dataset simply as D. For each position s in D, we compute all relevant concepts ($c_i(s)$) and Stockfish's static evaluations by both the classical and NNUE models, $f_{classical}(s)$ and $f_{NNUE}(s)$. The resulting data is used in our experiments to generate the concept-based regression surrogate models.

In our concept probing experiments, we use ridge regression, a linear model that minimizes the squared error with L2 regularization. For each probe, we perform a hyperparameter search over alpha values (the L2 term multiplier) of [0.01, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000]. The error bars of Figures 6 and 7 show the standard error of the mean of cross-validation results over five splits.

4.2 Model-Agnostic Interpretation

The results we present here treat the NNUE model as a black box, interpreting its output in terms of its inputs only.

One of the first thing newcomers to chess learn, besides the rules, is the value of the pieces, presented relative to a pawn's value. Assuming a pawn value of one, the chess literature most commonly gives a knight and a bishop a value of three, the rook the value of five, and the queen the value of nine. Of course, many positional factors also determine how effective the pieces are in different situations, but this assignment is a good rule of thumb for the pieces' intrinsic value.

Figure 2 shows how Stockfish's models value the pieces, normalized such that a pawn's value is one. Both models are mostly in agreement with the chess literature values; however, seemingly, the classical model values its minor and major pieces (especially the queen) slightly more, whereas the

	NNUE	Classical	ratio
Material	0.412	0.573	0.719
Winnable	0.187	0.147	1.269
Passed pawns	0.055	0.045	1.234
Imbalance	0.040	0.076	0.524
Mobility	0.023	0.035	0.642
King safety	0.019	0.086	0.226
Threats	0.004	0.012	0.365
Rooks	0.003	0.009	0.385
Bishops	0.002	0.003	0.508
Space	0.002	0.008	0.244
Pawns	0.001	0.004	0.337
Knights	0.001	0.001	0.466
Queens	0.000	0.001	0.318

Table 3: Shapley values estimated using Shapley value sampling with a linear model. Ratio is calculated as the value of NNUE divided by the value of Classical.

NNUE model values them slightly less. In Figure 3, shows how the models evaluate the relative piece values during different game phases. The overall trend for both models is that as the game progresses, the relative difference between the pawns and the other pieces decreases. This may be explained by the pawns becoming more awake in the endgame, where being even a single pawn up is often a decisive advantage.

As chess players progress in strength, they start to take numerous other non-material-based concepts into account when evaluating the merits of chess positions, for example, in line with the higher-level hand-crafted concepts used in the classical model. It is thus interesting to look at how well those concepts explain the NNUE model evaluations. The graph in Figure 4 shows the result of a concept regression using the classical high-level concepts to explain the output of the NNUE model. First, we notice a relatively low fidelity, as witnessed by the fact that a linear combination of the classical concepts explains less than 50% of the variation of the NNUE model (i.e., r^2-score), indicating considerable disagreement between the two models. We also see, when inspecting the

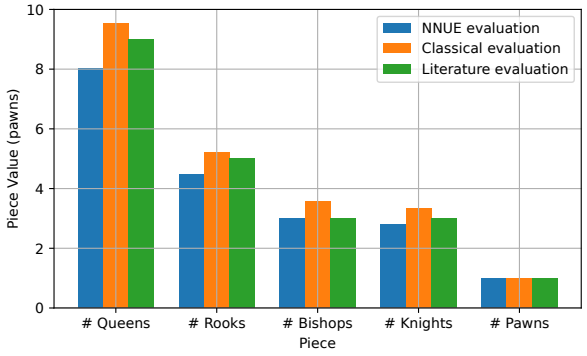

Figure 2: Piece values using Concept Regression. Each feature corresponds to #_* features in Table 2, i.e. its value is the difference in number of each piece type on the board.

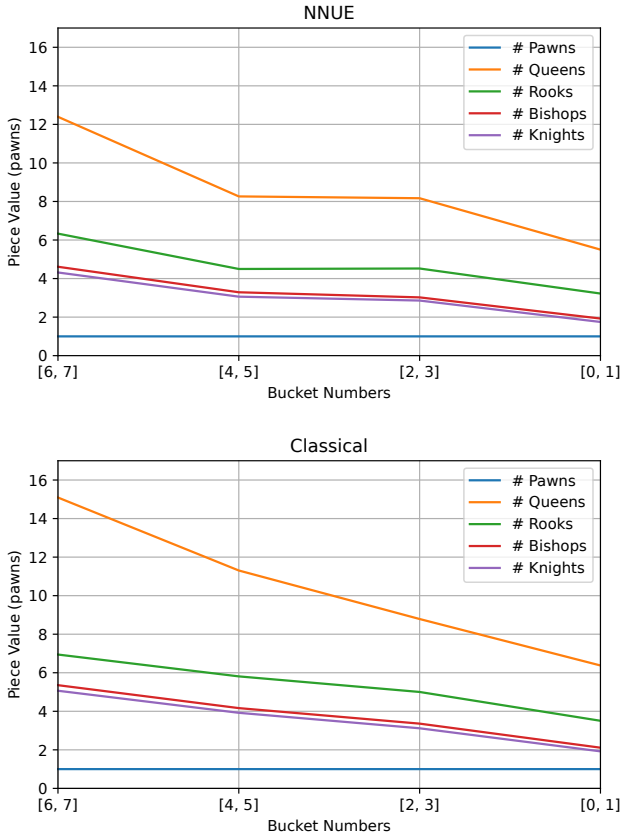

Figure 3: Comparison between the piece values depending on the phase of the game. The bucket number indicates which layer stack will be used; the number of pieces on the board determines the bucket number, which is calculated by $(piece\_count - 1)/4$.

weights of the linear surrogate model, that they disagree the most about the *relative* importance of the *Winnable*, *Space*, and *King-Safety* concepts, with the NNUE model placing less weight on those concepts than the classical one. However, this does not tell us about these concepts' *absolute* importance in the final evaluation.

Table 3 gives us a better grasp of the *absolute* importance using Shapley values, which we evaluated using Shapley value sampling. Both models see *Material* as the most critical concept, followed by *Winnable* and *Passed Pawns* (both of them being more critical for NNUE than the classical evaluation). Of the concepts we looked at, taking into account their overall importance, it seems as *King-Safety* — *the way the classical model computes it* — is not too useful for the NNUE model. Supposedly, the model has found a more meaningful way of evaluating king safety.

To assess the contribution of the low-level concepts in Table 2, we create a concept vector $c(z_0)$ using all concepts in the table except $material$ and $imbalance$. We are assessing the contribution using a linear model, thus we exclude those concepts to avoid interactions between features representing the same thing. E.g., the feature $imbalance$ awards a bishop pair versus having a bishop and a knight. Instead of using the

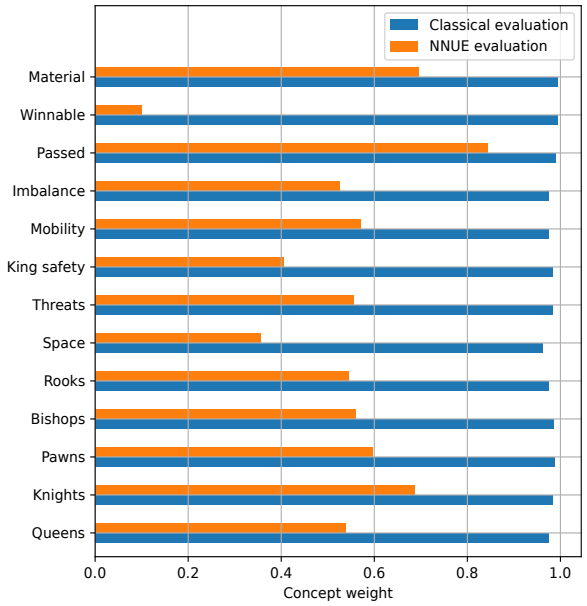

Figure 4: This figure shows the weights of two linear surrogate models used to explain the Classical and NNUE evaluations, respectively. The fitted models give a r^2-score of 0.999 for the classical evaluation and 0.497 for the NNUE evaluation, respectively.

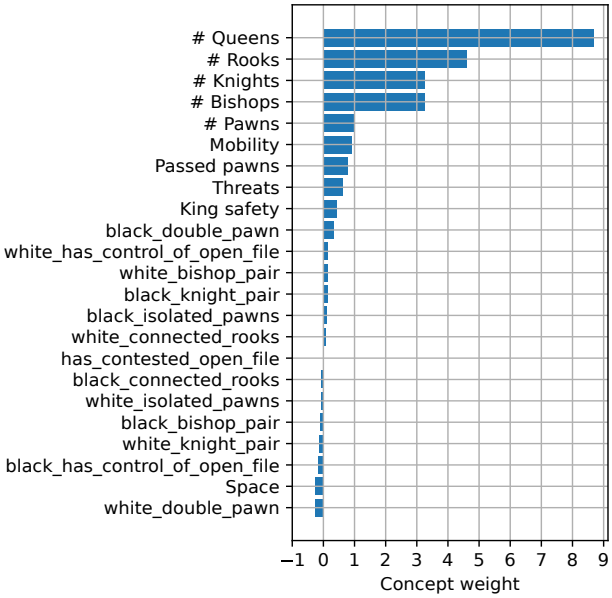

Figure 5: Surrogate model predicting NNUE's evaluation weights. We exclude *material* and *imbalance* to avoid concept interaction. It is advantageous to have a *bishop pair*, to have *control of an open file* and *connected rooks*, and disadvantageous to have *doubled pawns*.

concept $material$ we use the piece concepts used in previous experiments to describe the difference in the number of pieces. Figure 5 shows the result, where it becomes apparent that the NNUE favors, other things being equal, concepts such as the bishop pair, having control of an open file, doubling the opponent's pawns, and connected rooks.

4.3 Model-Specific Interpretation

In this subsection, we will dissect the model to understand its reasoning better. We use concept probing to identify if and how well the model represents those human-understandable concepts. We probe the model in two places, *i)* after preprocessing (i.e. the input, in the HalfKAv2_hm format) and *ii)* after the first hidden layer. Comparing the two helps distinguish attribution between the model learning and the expressive input representation. Here, we do not look deeper into the model because the information becomes much more compressed and is processed differently depending on the game phase, i.e., only one of 8 layer stacks is used each time.

In Figure 6, we show concept probing results performed on the input and after the first hidden layer. It is interesting to see that all concepts increase in probing accuracy after the first linear layer, meaning that the network is learning how to represent those higher-level concepts. Also, the contrast between the concepts *winnable* and *space* highlights an interesting intuition; *winnable* is the second most important concept (according to Table 3); the probing accuracy after the first layer is more than four times higher than on the input. In contrast, *space* is the least important concept and receives almost no increase in probing accuracy.

The relevance of the concepts may change with the game phase (e.g., an isolated pawn is more of a weakness in the endgame). Figure 7 shows the concept probing accuracy for each of the 8 phases (or buckets, with bucket 0 having the fewest pieces). Here we see that *i)* the concepts *winnable* and *passed pawns* become more relevant for the model as fewer pieces are left on the board, *ii)* the concept *material* is always well represented by the model, and *iii) king safety* and *threats* generally have a low agreement with the model but, proportionately, are better represented earlier in the game and increase again during the end-game in bucket 1.

4.4 Classical vs NNUE Evaluation

Finally, we conducted a qualitative analysis to gain even further insights into the differences in the evaluation of the two models. We filtered the dataset D for game positions where the two models were on the opposite view of which side had a clear advantage. Manual inspection unveiled some common motives as demonstrated in Figure 8. The overarching theme resulting from this inspection seems to be that the NNUE model better evaluates various types of threats (which may take several moves to materialize), like forks, mating-attacks, and the potential for promoting pawns.

The king-safety feature is one such example and of particular interest. Although intricate in the classical model, considering weak squares, pawn shelter around the king, and the attacking potentials of the opponent's pieces, it nonetheless poorly agrees with the learned model. A more in-depth qualitative inspection showed the classical model often overestimating the dangers of being able to immediately attack the opponent's king with a Rook or a Queen, for example, as seen in the bottom diagrams in Figure 8, whereas the NNUE model

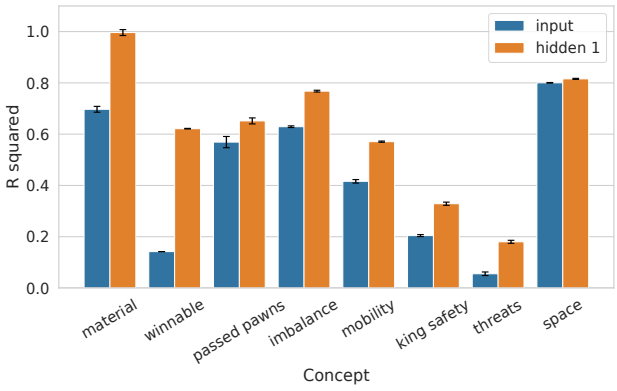

Figure 6: Concept probing results on i) the input features and ii) after the first hidden layer, using the official NNUE weights.

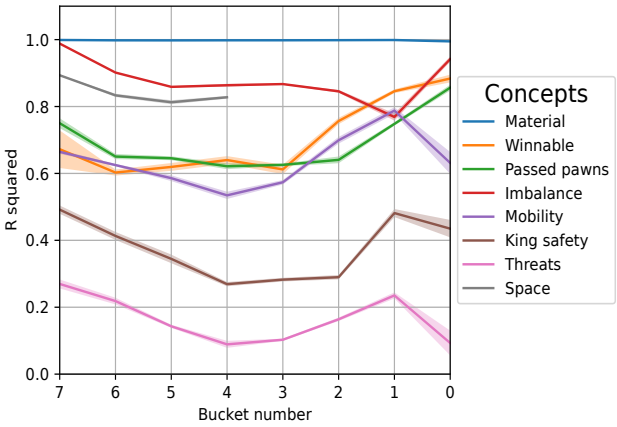

Figure 7: This figure shows concept probing results on subsets containing only a single bucket using ridge regression, evaluated at the first hidden layer of the model, using the official NNUE weights. The bucket number indicates which layer stack will be used; the number of pieces on the board determines the bucket number, which is calculated by $(piece\_count - 1)/4$.

is more realistic about the attacking prospects.

5 Conclusions and Future Work

There are several valuable takeaways from this work.

First, both model-agnostic and model-specific explainability techniques seem to, for the most part, do a reasonable job of interpreting and explaining the knowledge acquired by the game agent's deep neural network, each with its pros and cons. However, given the unique architecture of the NNUE network, some care is needed in applying especially the model-specific techniques.

Second, we highlighted crucial similarities and differences in the evaluation of Stockfish's hand-crafted and NNUE models. In contrast to the hand-crafted model, the NNUE model puts less weight on material and emphasizes more dynamic concepts like passed pawns. Also of interest is the low agreement with some high-level concepts investigated. In particular, the NNUE network clearly understands the idea of king-safety; otherwise, it would not correctly evaluate its (or the opponent's) attacking potential. However, the king-safety concept manually defined in the hand-crafted evaluation function does not have a clear correspondence in the NNUE network; instead, the NNUE model seemingly has found an alternative and more effective way of evaluating king safety. Finally, it was impressive to see how the NNUE can *statically* detect threats such as forks, promotions, and attacking potentials, allowing it to see threats right away that would require a look-ahead search in the classical version of Stockfish. It is worth noting that [McGrath *et al.*, 2021] identified similar tactical abilities in the deep neural network learned by AlphaZero; however, they attributed that to the network simultaneously learning a value function and a (look-a-head) policy. We show such tactics are learnable without simultaneous policy learning.

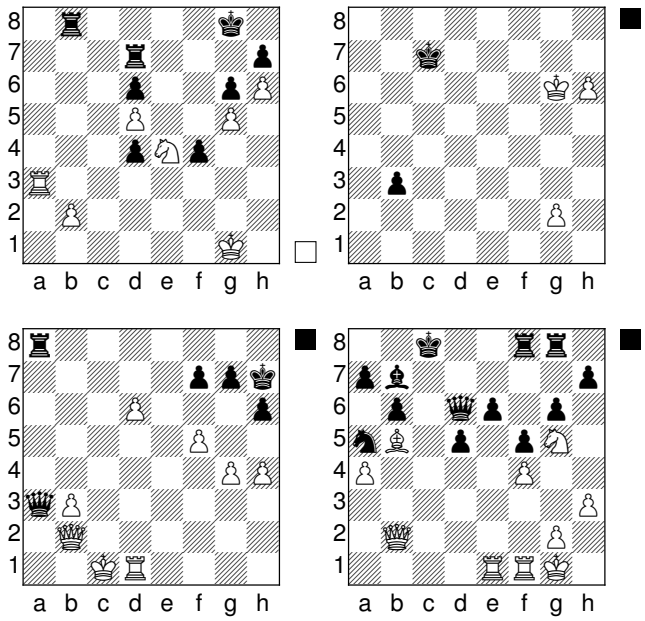

Figure 8: (Upper Left) The NNUE statically detects the *fork* Nf6 favouring White unlike the classical model; (Upper Right) Black (to move) wins because promoting with a check, seen (statically) by the NNUE but not the classical model. (Bottom) The classical model judges king safety the most critical feature favoring the attacking player (Black and White, respectively); the NNUE is unimpressed with the attacks and correctly (slightly) prefers the defending player.

As for future work, more intricate higher-level concepts are called for to understand further the evaluation differences between the two models, classical and NNUE, for example, for king-safety, as our result indicates. Although we chose to focus on chess only for this work, the concept probing methods we use are game independent and thus applicable to a wide array of games/problems. This generality is one of the appeals of the proposed approach. Thus, using these techniques to analyze, for example, a deep neural network agent learned by a general-game-playing agent would be of interest, possibly identifying some generic cross-game concepts.

References

[Alain and Bengio, 2017] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017.

[Bodria *et al.*, 2021] Francesco Bodria, Fosca Giannotti, Riccardo Guidotti, Francesca Naretto, Dino Pedreschi, and Salvatore Rinzivillo. Benchmarking and survey of explanation methods for black box models. *CoRR*, abs/2102.13076, 2021.

[Castro *et al.*, 2009] Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009. Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X).

[Decodea, 2022] Decodea. Decode chess. https://decodechess.com/, 2022. Accessed: 2022-01-30.

[Guid *et al.*, 2013] Matej Guid, Martin Mozina, Ciril Bohak, Aleksander Sadikov, and Ivan Bratko. Building an intelligent tutoring system for chess endgames. In Owen Foley, Maria Teresa Restivo, James Onohuome Uhomoibhi, and Markus Helfert, editors, *CSEDU 2013 - Proceedings of the 5th International Conference on Computer Supported Education, Aachen, Germany, 6-8 May, 2013*, pages 263–266. SciTePress, 2013.

[HaCohen-Kerner, 1994] Yaakov HaCohen-Kerner. Case-based evaluation in computer chess. In Jean Paul Haton, Mark T. Keane, and Michel Manago, editors, *Advances in Case-Based Reasoning, Second European Workshop, EWCBR-94, Chantilly, France, November 7-10, 1994, Selected Papers*, volume 984 of *Lecture Notes in Computer Science*, pages 240–254. Springer, 1994.

[HaCohen-Kerner, 1995] Yaakov HaCohen-Kerner. Learning strategies for explanation patterns: Basic game patterns with application to chess. In Manuela M. Veloso and Agnar Aamodt, editors, *Case-Based Reasoning Research and Development, First International Conference, ICCBR-95, Sesimbra, Portugal, October 23-26, 1995, Proceedings*, volume 1010 of *Lecture Notes in Computer Science*, pages 491–500. Springer, 1995.

[Kim *et al.*, 2018] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viégas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2673–2682. PMLR, 2018.

[LeelaChessZero, 2022] LeelaChessZero. Leela chess zero. https://lczero.org/, 2022. Accessed: 2022-01-30.

[Lipovetsky and Conklin, 2001] Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17(4):319–330, 2001.

[Lovering *et al.*, 2022] Charles Lovering, Jessica Forde, George Konidaris, Ellie Pavlick, and Michael Littman. Evaluation beyond task performance: Analyzing concepts in alphazero in hex. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 25992–26006. Curran Associates, Inc., 2022.

[Lundberg and Lee, 2017] Scott M Lundberg and Su In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 4766–4775, 2017.

[McGrath *et al.*, 2021] Thomas McGrath, Andrei Kapishnikov, Nenad Tomasev, Adam Pearce, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. Acquisition of chess knowledge in alphazero. *CoRR*, abs/2111.09259, 2021.

[McGrath *et al.*, 2022] Thomas McGrath, Andrei Kapishnikov, Nenad Tomašev, Adam Pearce, Martin Wattenberg, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. Acquisition of chess knowledge in alphazero. *Proceedings of the National Academy of Sciences*, 119(47):e2206625119, 2022.

[Mi *et al.*, 2020] Jian Xun Mi, An Di Li, and Li Fang Zhou. Review study of interpretation methods for future interpretable machine learning. *IEEE Access*, 8:191969–191985, 2020.

[Nasu, 2018] Yu Nasu. Efficiently updatable neural-network-based evaluation functions for computer shogi. *The 28th World Computer Shogi Championship Appeal Document*, 2018.

[Pálsson and Björnsson, 2021] Aðalsteinn Pálsson and Yngvi Björnsson. Evaluating interpretability methods for dnns in game-playing agents. In Cameron Browne, Akihiro Kishimoto, and Jonathan Schaeffer, editors, *Advances in Computer Games - 17th International Conference, ACG 2021, Virtual Event, November 23-25, 2021, Revised Selected Papers*, volume 13262 of *Lecture Notes in Computer Science*, pages 71–81. Springer, 2021.

[Puri *et al.*, 2020] Nikaash Puri, Sukriti Verma, Piyush Gupta, Dhruv Kayastha, Shripad Deshmukh, Balaji Krishnamurthy, and Sameer Singh. Explain your move: Understanding agent actions using specific and relevant feature attribution. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[Ribeiro *et al.*, 2016] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?" Explaining the predictions of any classifier. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 13-17-Augu:1135–1144, 2016.

[Sadikov *et al.*, 2006] Aleksander Sadikov, Martin Mozina, Matej Guid, Jana Krivec, and Ivan Bratko. Automated

chess tutor. In H. Jaap van den Herik, Paolo Ciancarini, and H. H. L. M. Donkers, editors, *Computers and Games*, volume 4630 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 2006.

[Schwalbe and Finzel, 2021] Gesina Schwalbe and Bettina Finzel. XAI method properties: A (meta-)study. *CoRR*, abs/2105.07190, 2021.

[Silver *et al.*, 2018] 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.

[Stockfish, 2022a] Stockfish. Introducing NNUE evaluation. https://blog.stockfishchess.org/post/625828091343896577/introducing-nnue-evaluation, 2022. Accessed: 2022-04-05.

[Stockfish, 2022b] Stockfish. NNUE readme. https://github.com/glinscott/nnue-pytorch/blob/master/docs/nnue.md, 2022. Accessed: 2022-04-05.

[Stockfish, 2022c] Stockfish. Stockfish: Strong open source chess engine. https://stockfishchess.org/, 2022. Accessed: 2022-01-30.

[Stockfish, 2022d] Stockfish. Training datasets. https://github.com/glinscott/nnue-pytorch/wiki/Training-datasets, 2022. Accessed: 2022-01-30.