

Check Mate: A Sanity Check for Trustworthy AI

Sascha Mücke¹, Lukas Pfahler¹

¹AI Group, TU Dortmund University, Dortmund, Germany

Abstract

Methods of Explainable AI (XAI) try to illuminate the decision making process of complex Machine Learning models by generating explanations. However, for most real-world data there is no “ground-truth” explanation, which makes evaluating the correctness of XAI methods and model decisions difficult. Often visual assessment or anecdotal evidence is the only type of evaluation. In this work we propose to use the game of chess as a source of “near ground-truth” (NGT) explanations, which XAI methods can be compared against using various metrics, serving as a “sanity check”. We demonstrate this process in an experiment with a deep convolutional neural network, to which we apply a range of commonly used XAI methods. As our main contribution, we publish our data set of 30 million chess positions along with their NGT explanations for free use in XAI research.

Keywords

Explainable AI, Trustworthy AI, Convolutional Neural Networks, Chess

1. Introduction

Models in machine learning (ML) and artificial intelligence (AI) have become larger and more complex over the past decades [1, 2, 3]. Long since the revival of artificial neural network (ANNs) and the advent of deep convolutional architectures [4] has the decision process of such models – Deep Learning models [5] in particular – become so utterly complex, that no human user can explain it using conventional reasoning. At the same time, more and more ML models are used for decisions that influence the health and welfare of human beings [6, 7]. Explainable AI (XAI) has emerged as a research branch that aims, among other things, to generate human-interpretable explanations from trained models [8]. However, in most cases it is unclear what a “good explanation” should be, and, if there are several, which one is better than the other.

One way to approach this problem is to look toward rule-based systems with perfect information, where every scenario can be interpreted. A great example is the game of chess, an abstract strategy board game played between two players, whose origins date back several centuries. The board consists of an 8×8 grid of squares. Each player has a set of *pieces*: a King, a Queen, two Bishops, two Knights, two Rooks, and eight Pawns. The players take turns moving one of their pieces, where each piece type has a specific movement pattern. Enemy pieces can be captured. The goal is to capture the enemy King. If the King is under attack by an enemy piece,

LWDA'22: *Lernen, Wissen, Daten, Analysen*. October 05–07, 2022, Hildesheim, Germany

✉ sascha.muecke@tu-dortmund.de (S. Mücke); lukas.pfahler@tu-dortmund.de (L. Pfahler)


🌐 <https://www-ai.cs.tu-dortmund.de/PERSONAL/muecke.html> (S. Mücke);

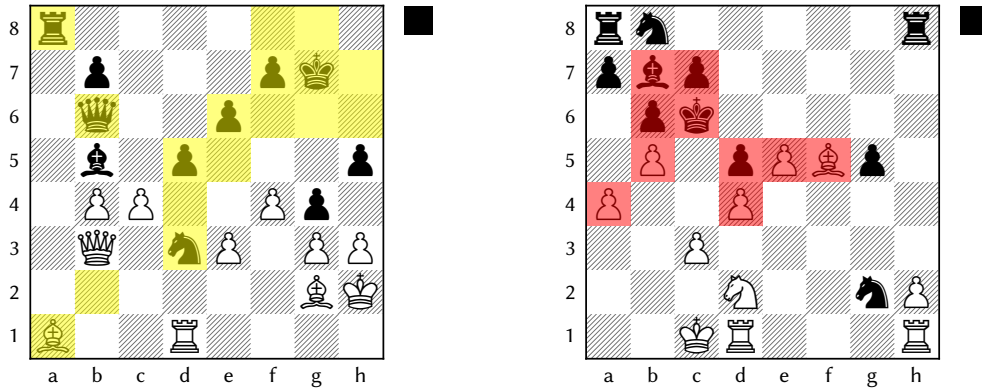
<https://www-ai.cs.tu-dortmund.de/PERSONAL/pfahler.html> (L. Pfahler)

🆔 0000-0001-8332-6169 (S. Mücke); 0000-0003-4012-4502 (L. Pfahler)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)



(a) Bishop on a1 puts black King on g7 in check

(b) Black King on c6 is checkmate

Figure 1: Two example data points from our chess data set. The colored squares represent the explanations: For check, they show legal moves. For checkmate, they highlight pieces that are essential for checkmate.

the position is called *check*. If, in addition, the player cannot avoid that their King be captured on the next move, the position is called *checkmate*, and the game is over.

Chess is strictly rule-based, yet highly complex regarding the number of possible game positions. It has historical significance for AI, being perceived as an important milestone in approaching (and surpassing) human-level intelligence, which was reached when IBM’s chess computer Deep Blue defeated then world chess champion Garry Kasparov in 1997 [9]. Today, chess computers are far stronger than any human player.

However, in this paper we follow a different path and use chess as a data source for an XAI benchmark. To this end, we extract millions of chess positions from a large database of real chess games between human players. We sort all positions into one of three classes (no check, check or checkmate) and generate several binary 8×8 masks each, indicating which squares are important to explain the position at hand (see fig. 1, which will be explained in detail in section 2). We call these *near ground-truth* (NGT) explanations, as there are numerous equally valid representations of an explanation, and we select some of them.

Specifically, in this work we restrict the explanations to binary 8×8 masks, which highlight certain squares. The advantage is that feature attributions produced by common XAI methods can easily be compared to these masks by means of distance metrics such as Euclidean or cosine. If one finds a high correspondence between XAI output and any of our NGT explanations, one can be more confident that (i) their model pays attention to useful features, and (ii) the XAI method at hand works as intended.

1.1. Related Work

Explainable Machine Learning is a very active field of research, and a multitude of post-hoc model explanation methods has been proposed [10, 11, 12, 13, 14, 15]. We should note that evaluating explanations is a non-trivial task on its own [16, 17, 18]. Particularly visual explanations

and example-based explanations [19] are often subjective and defining quantitative metrics is difficult. Datasets where expert explanations are provided as an additional layer of annotation may allow quantitative evaluations. As such, the benchmark we propose can be classified as a “Controlled Synthetic Data Check” according to [20], though our data is not synthetic but is taken from actual chess games played by humans.

However, it is not always apparent that there is a single, unique explanation. Synthetic data can ensure this. For instance, Ismail et al. use synthetic time-series data [21] and Sanchez-Lengeling et al. [22] provide a set of synthetic graph-classification tasks where labeled sets of nodes are responsible for the target variable. Tritscher et al. generate datasets of synthetic categorical features and use randomly generated boolean functions as labeling functions with 3 to 10 important features [23]. In these checks, the XAI methods should identify only truly relevant parts of the input as relevant. More recently, Tritscher et al. provide a more realistic dataset for fraud detection where domain experts have manually annotated the “problematic” features of simulated fraud cases [24].

Our contribution is somewhat more general, as we provide a dataset of raw chess positions for a generic classification task, which can be approached with a wide variety of feature extraction methods and model types. Also, to the best of our knowledge, the domain of chess has not yet been explored as an XAI benchmark.

Other related research focuses on evaluating how well XAI methods perform from a user’s perspective [25]. We focus instead on the objective similarity between generated and NGT explanations, which are independent from human users.

1.2. Structure of this Paper

In section 2 we introduce our dataset of chess positions and NGT explanations, detailing our pre-processing steps and our reasoning behind the specific explanations we chose. In section 3 we train an ML model to classify the chess positions, apply post-hoc XAI methods to the trained model and compare the output to our NGT explanations using various metrics. Finally, in section 4 we discuss other possible applications of our data set, and further research directions we identified. Information about how to access our data set can be found there, as well.

2. Data Set

Our data set is based on the Lichess open database¹, which contains records of over 3 billion games of chess played online by human players on the free chess website lichess.org. The records are saved in *Portable Game Notation* (PGN) format, containing the series of moves played as well as meta information, such as player ratings, timings and computer evaluations. The database is split into downloadable files for each month since January 2013. For our data set, we arbitrarily chose May 2021, as it contains over 100 million games. To read and process the games and to create the explanations, we used the Python package `chess`².

¹<https://database.lichess.org/>

²<https://python-chess.readthedocs.io/en/latest/>

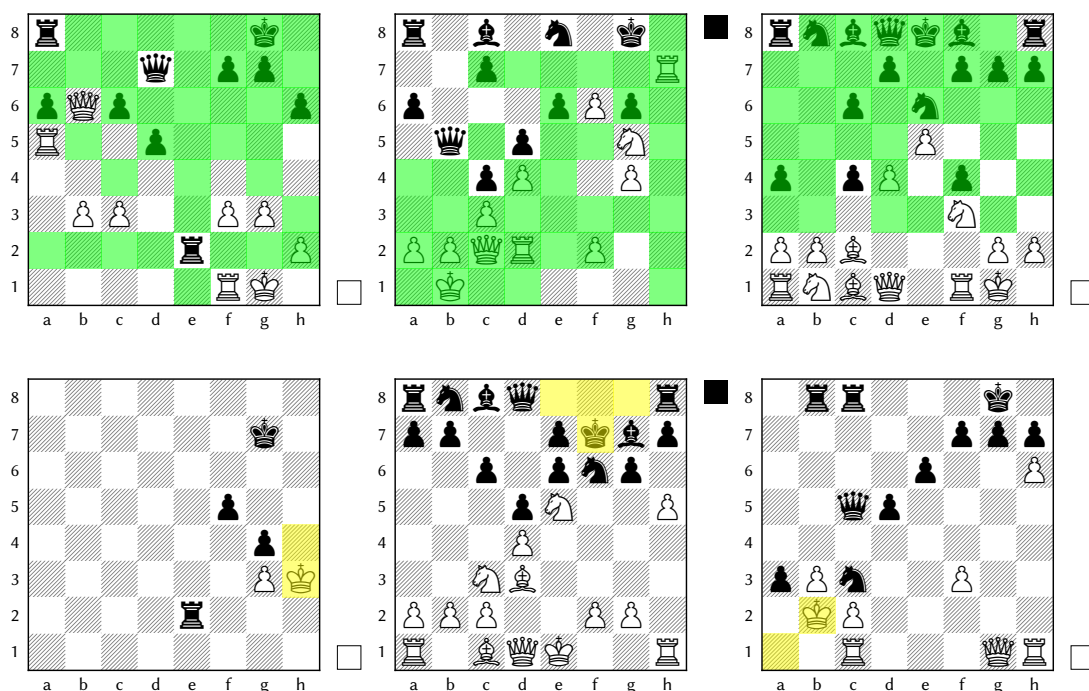


Figure 2: Some data points from our data set, consisting of chess positions along with explanations (highlighted squares; green means “no check”, yellow means “check”). The black and white squares to the right of the chess boards indicate whose turn it is. The top row shows positions with no check, where one possible explanation is the squares that the opponent controls/attacks. As the King is not on one of these squares, there is no check. The bottom row shows positions where the King is in check, but not checkmate. The explanation shows all squares (origin and target) that have legal moves.

Table 1
Summary of data set

label	class	number of positions
0	no check	9,998,984
1	check	9,996,870
2	checkmate	10,004,146
total		30,000,000

We selected only those games that end in checkmate, excluding those that end by timeout or resignation. In a first pre-processing step, we iterate through the games and extract random positions that occur during the game. Each position falls into one of three classes: *no check* (0), *check* (1) or *checkmate* (2).

There are many more non-check positions than check positions in a typical game of chess, and again many more checks than checkmates. To approximately balance the classes while iterating over the positions, we accept each position x with label y randomly with probability

- *No check* (0): All squares that are controlled by the enemy player, i.e., all squares that can be reached or captured on by any enemy piece. The fact that the King is not on any of these squares proves that the position is not check.
- *Check* (1): All squares (origin or target) of legal moves. As a checkmate is a check where the player under attack has no more legal moves, highlighting legal moves is sufficient to disprove a checkmate. Note that the piece giving check is only highlighted when it can be legally captured.
- *Checkmate* (2): All squares with pieces that are essential for creating the checkmate. This includes attackers, friendly pieces blocking the King, enemy pieces guarding escape squares and enemy pieces protecting attackers.

Examples of explanations for 0 and 1 can be seen in fig. 2, for 2 in fig. 3. As discussed in section 1, none of these explanations is *the* perfect explanation. For instance, the explanation given for class 0 is also a valid explanation for class 1, and vice versa. However, in this work we aim to provide a variety of explanations with different semantics, and the choice which explanation to apply to which class is up to the users of our data set. We therefore generate each type of explanation mentioned above for each data point, regardless of its class.

We save the explanations in a CSV file as unsigned integers representing 64-bit binary masks. Using the `SquareSet` class from the `chess` package, these integers can be converted back to binary masks (or other representations). Along with our data set, we provide code to convert between different representations.

3. Experiments

We transformed the FEN representation of each chess position into a 12×8 binary feature tensor x , where the first dimension represents all combinations of 6 chess piece types and 2 colors, and the remaining dimensions are board coordinates. Feature $x_{ijk} = 1$ if there is a piece of type i on square (j, k) . We used this representation because it contains a minimal amount of domain knowledge, though other representations are possible, too (e.g., 8 categorical features).

We train a convolutional neural network with 4 convolution layers with a total of 782,851 trainable parameters on the given 3-class decision problem. A detailed description can be found in appendix A. The convolutions use filter widths 7, 5 and 3, respectively, with centered padding. The larger filters help the network detect how far-away positions interplay. We add weight decay of 0.001 to combat overfitting in light of these large filter matrices. We use the Adam optimizer for a total of 400,000 weight updates with an initial learning rate of 0.001 that is reduced by a factor of 0.1 four times during training. The final model achieves a classification accuracy of 99.02% on 10,000 hold-out examples. The training accuracy also converges at 99%. It is not entirely surprising that a sufficiently large deep network can learn a deterministic function given millions of data points.

We want to stress that the point of this work is *not* to propose to use ML to classify chess positions. A deterministic, polynomial-time algorithm can do that just fine, such as the one included in the `chess` Python package we used. Instead we want to explore how some commonly

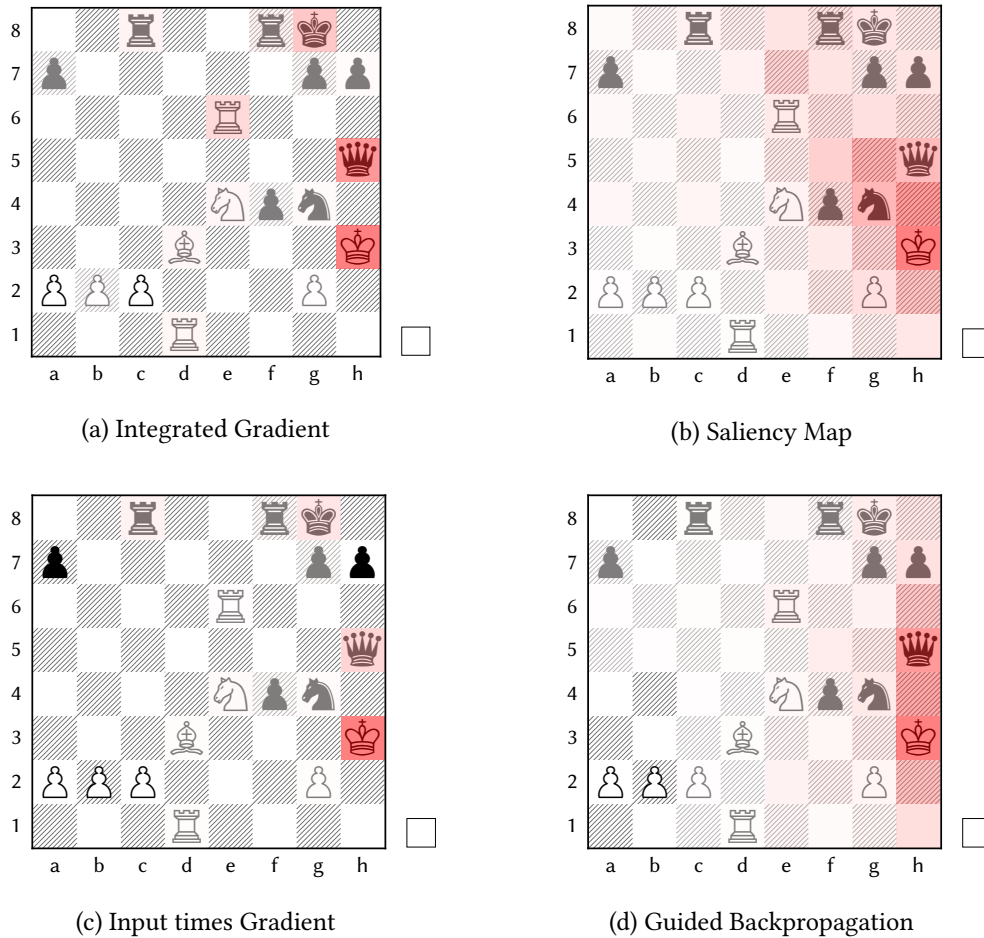


Figure 4: Examples of four different attribution methods on the first checkmate example from fig. 3. The $12 \times 8 \times 8$ feature attribution maps were first reduced to 8×8 by taking the absolute maximum over the first dimension, and then rescaled to a value range of $[0, 1]$. Darker red means higher attribution value. There is a clear overlap between the explanation given in fig. 3 and the attributions shown here. However, only the King itself and the attacking Queen are clearly highlighted in all cases, while the pawns on f4 (guarding the King’s escape square g3) and on g2 (blocking the King) have overall low attribution, even though they are essential for checkmate.

used post-hoc explanation approaches applied to our convolutional neural network explain the model’s decision, showcasing a possible use case of our data set. Because we have domain knowledge on the rules of chess and access to the true decision boundary, we can inspect the explanations and see if they are sound and in concordance with the rules of chess. In addition, our NGT explanations allow us to compute distances, reducing the necessity of subjective evaluation through domain experts.

The explanation methods we consider are Saliency maps and Gradient times Input [10], Feature Permutation [11], Guided Grad-CAM [12], Guided Backprop [13], Integrated Gradient

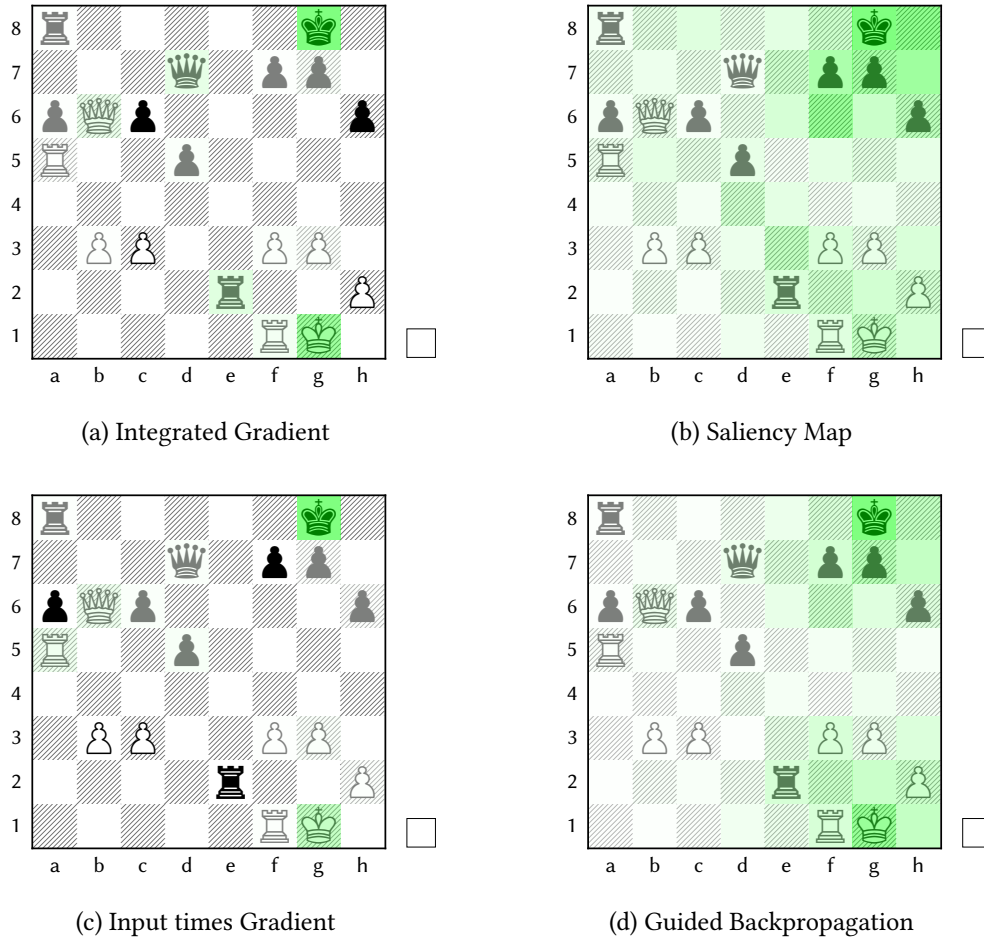


Figure 5: Examples of four different attribution methods on the first “no check” example from fig. 2. The same pre-processing steps were taken as in fig. 4. Darker green means higher attribution value. The overlap with the explanation given by the enemy-controlled squares in fig. 2 is far less pronounced as for “checkmate”.

[14] and Occlusion [15]. All these methods are implemented in the Captum software library³ for explainable machine learning. We apply Guided Grad-CAM to the first and last convolution layers of our model, respectively.

As distance measures, we choose Chebyshev (ℓ_∞), correlation, cosine and Euclidean (ℓ_2) distance. To make the $12 \times 8 \times 8$ feature attributions compatible with our 8×8 NGT explanations, we take the absolute maximum $\|\cdot\|_\infty$ over the first dimension, which gives us the strongest attribution found for each square.

We apply each distance measure to each combination of XAI method and class-specific NGT explanation (as described in section 2).

³<https://captum.ai>

Table 2

Distances between different feature attribution methods and our ground-truth explanation for the “no check” class, with mean and standard deviation taken over all test examples of this class. Various distance measures were used.

Measure XAI method	Chebyshev distance	Correlation distance	Euclidean distance
Feature Ablation	3.6245±1.9160	1.0163±0.0884	6.9408±1.6052
Guided Backpropagation	6.3602±1.5418	0.9528±0.1590	12.2528±2.7651
Guided Grad-CAM (first conv.)	1.0044±0.0690	—	5.4767±0.8110
Guided Grad-CAM (last conv.)	1.4833±0.8212	0.9117±0.2075	5.0706±0.7382
Input Times Gradient	2.6702±1.2361	1.0455±0.0921	6.2472±1.0679
Integrated Gradient	1.9684±0.3893	1.0465±0.0761	5.8674±0.5746
Occlusion	3.6245±1.9160	1.0163±0.0884	6.9408±1.6052
Saliency Map	3.3353±1.6954	0.9206±0.2007	8.7560±4.0765

Table 3

Same as table 2, but for class “check”.

Measure XAI method	Chebyshev distance	Correlation distance	Euclidean distance
Feature Ablation	7.5070±1.3288	0.6233±0.1680	10.6495±1.7607
Guided Backpropagation	23.2978±6.8497	0.4929±0.1271	37.5013±9.6879
Guided Grad-CAM (first conv.)	2.8089±2.3667	—	3.8415±2.5108
Guided Grad-CAM (last conv.)	6.9291±3.7849	—	10.4738±5.1589
Input Times Gradient	6.0985±3.3081	0.6052±0.1717	7.5247±3.4519
Integrated Gradient	4.1820±1.0129	0.5767±0.1656	5.2523±0.9886
Occlusion	7.5070±1.3288	0.6233±0.1680	10.6495±1.7607
Saliency Map	7.4877±3.3650	0.5753±0.1459	18.4707±8.0551

3.1. Results

We report the results of our experiment in tables 2 to 4. Cosine distance turned out to be almost equal to 1 in all cases, which is why we dropped it from the result tables. Although there is no clear overall trend, Guided Grad-CAM is often closest to NGT among the XAI methods we tested on classes 0 and 1. On class 2, Integrated Gradient is closest, though Guided Grad-CAM performs also very well.

Figures 4 and 5 show some exemplary visualizations of explanations generated by various methods. In fig. 4 we observe high attributions in the general vicinity of the King and its attacker (the Queen on h5), although important squares that are contained in our NGT explanation (Pawns on f4 and g2 blocking the King’s escape squares) have no significant attribution. A possible explanation is that XAI methods may recognize that the Queen is often involved in checkmate positions when it is in the King’s vicinity, but fails to capture the more subtle rules of the game, which make f4 and g2 essential pieces for this checkmate. To understand why they are essential, one has to understand the movement patterns of both the King and the pawn, while also looking one move ahead. Understanding the Queen attacking the King requires only

Table 4

Same as table 2, but for class “checkmate”.

Measure XAI method	Chebyshev distance	Correlation distance	Euclidean distance
Feature Ablation	6.7008±1.3060	0.0485±0.0448	10.2244±2.2771
Guided Backpropagation	19.0280±5.2093	0.3939±0.0841	29.2725±7.0950
Guided Grad-CAM (first conv.)	2.1203±1.2152	0.4097±0.1642	3.0124±1.3102
Guided Grad-CAM (last conv.)	7.8247±3.6550	0.3996±0.1029	11.6994±4.7532
Input Times Gradient	3.8990±1.8152	0.2807±0.1206	4.8469±2.0273
Integrated Gradient	1.9470±0.5789	0.3005±0.1082	2.7842±0.5976
Occlusion	6.7008±1.3060	0.0485±0.0448	10.2244±2.2771
Saliency Map	5.3583±2.0448	0.5341±0.1250	14.1654±6.1172

the movement pattern of one piece, and no looking ahead, which makes it arguably easier to recognize as a reason for check or checkmate.

Figure 5 shows radically different attributions from our NGT, indicating that the model uses other criteria to judge that a position is not a check. This is not entirely surprising, as explaining why a position is *not* a check is far more difficult and vague than showing the opposite.

4. Conclusion

We have presented a data set of 30 million chess positions falling into one of three classes (*no check*, *check* or *checkmate*), for which we generated three different types of explanations. These explanations are based on domain knowledge and can be considered “near ground-truth”, as the game of chess has perfect information, and every position can be classified according to a fixed set of rules. Such a data set of explanations is a valuable benchmark for evaluating post-hoc explanation methods which are developed within the research area of XAI. We showed this by training a deep convolutional neural network on the classification problem, applying a range of feature attribution methods and comparing the results to our NGT explanations using various distance metrics. This practically eliminates the need for a domain expert assessment, which is still the most prevalent way to judge the quality of XAI methods. We found that, among the XAI methods we looked into, Guided Grad-CAM was closest to NGT according to our metrics.

With the application we showed in this paper, we have barely scratched the surface of possible use cases for our data set. Our NGT explanations may be used to guide model training to obtain more interpretable models [26]. This applies to all gradient-based methods that we used for our experiments. By using an oracle instead of a trained model, one could evaluate the performance of black-box XAI methods, factoring out the model performance. Instead of varying the explanation method, one could vary the model type in order to investigate the overall deciding making strategies of different models and architectures.

The three classes we chose (*no check*, *check* and *checkmate*) are more or less arbitrary, and any other class that is deterministically decidable could be used in a similar way, e.g. which player has more material on the board, which player controls more squares, etc. Also it is easily conceivable to apply our method to other games or rule-based systems in general as a

similar data source. Board games such as Go or Chinese Chess are straightforward examples. Exploring more such NGT explanations from various domains could lead to even more thorough automated performance evaluation of XAI methods.

4.1. Using our Data Set

We strongly encourage active use of our data set for evaluating XAI methods or any other research project. We have published our data set along with some code for processing and feature-extraction on <https://www.kaggle.com/datasets/smuecke/chess-xai-benchmark>.

Acknowledgments

This research has been funded by the Federal Ministry of Education and Research of Germany and the state of North-Rhine Westphalia as part of the Lamarr-Institute for Machine Learning and Artificial Intelligence, LAMARR22B. This work has further been supported by Deutsche Forschungsgemeinschaft (DFG), as part of the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", project A1.

References

- [1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, et al., Language Models are Few-Shot Learners, in: Proceedings of NeurIPS 2020, 2020.
- [2] W. Fedus, B. Zoph, N. Shazeer, Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, arXiv preprint arXiv:2101.03961 (2021).
- [3] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, et al., Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model, arXiv preprint arXiv:2201.11990 (2022).
- [4] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in neural information processing systems 25 (2012).
- [5] Y. LeCun, Y. Bengio, G. E. Hinton, Deep learning, Nat. 521 (2015) 436–444.
- [6] S. Dua, U. R. Acharya, P. Dua (Eds.), Machine Learning in Healthcare Informatics, volume 56 of *Intelligent Systems Reference Library*, Springer, 2014.
- [7] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, et al., A guide to deep learning in healthcare, Nature medicine 25 (2019).
- [8] P. Linardatos, V. Papastefanopoulos, S. Kotsiantis, Explainable ai: A review of machine learning interpretability methods, Entropy 23 (2020) 18.
- [9] M. Campbell, A. J. Hoane Jr, F.-h. Hsu, Deep blue, Artificial intelligence 134 (2002) 57–83.
- [10] K. Simonyan, A. Vedaldi, A. Zisserman, Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, arXiv preprint arXiv:1312.6034 (2014).
- [11] C. Molnar, Interpretable Machine Learning, 2020. URL: <https://christophm.github.io/interpretable-ml-book/>.
- [12] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: Visual

Explanations from Deep Networks via Gradient-Based Localization, *International Journal of Computer Vision* 128 (2019) 336–359.

- [13] J. T. Springenberg, A. Dosovitskiy, T. Brox, M. A. Riedmiller, Striving for simplicity: The all convolutional net, in: *Workshop Track Proceedings of ICLR 2015*, 2015.
- [14] M. Sundararajan, A. Taly, Q. Yan, Axiomatic attribution for deep networks, in: *Proceedings of ICML 2017*, JMLR.org, 2017, pp. 3319–3328.
- [15] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: *Proceedings of ECCV 2014*, *Lecture Notes in Computer Science*, Springer, 2014.
- [16] J. Zhou, A. H. Gandomi, F. Chen, A. Holzinger, Evaluating the Quality of Machine Learning Explanations : A Survey on Methods and Metrics (2021) 1–19.
- [17] F. Bodria, F. Giannotti, R. Guidotti, F. Naretto, D. Pedreschi, S. Rinzivillo, A Survey Of Methods For Explaining Black-Box Models 51 (2021) 1–33.
- [18] M. Nauta, J. Trienes, S. Pathak, E. Nguyen, M. Peters, Y. Schmitt, J. Schlötterer, M. van Keulen, C. Seifert, From Anecdotal Evidence to Quantitative Evaluation Methods: A Systematic Review on Evaluating Explainable AI, *Technical Report 1*, 2022.
- [19] A.-p. Nguyen, M. R. Martínez, On quantitative aspects of model interpretability, *Technical Report*, 2020.
- [20] M. Nauta, J. Trienes, S. Pathak, E. Nguyen, M. Peters, Y. Schmitt, J. Schlötterer, M. v. Keulen, C. Seifert, From Anecdotal Evidence to Quantitative Evaluation Methods: A Systematic Review on Evaluating Explainable AI, *arXiv preprint arXiv:2201.08164* (2022).
- [21] A. A. Ismail, M. K. Gunady, H. C. Bravo, S. Feizi, Benchmarking Deep Learning Interpretability in Time Series Predictions, in: *NeurIPS 2020*, 2020.
- [22] B. Sanchez-Lengeling, J. Wei, B. Lee, E. Reif, P. Y. Wang, W. Qian, K. Mccloskey, L. Colwell, A. Wiltschko, Evaluating Attribution for Graph Neural Networks, in: *Advances In Neural Information Precessing Systems*, volume 33, Curran Associates, Inc., 2020, pp. 1–13.
- [23] J. Tritscher, M. Ring, D. Schlr, L. Hettinger, A. Hotho, Evaluation of Post-hoc XAI Approaches Through Synthetic Tabular Data, in: *Foundations of Intelligent Systems*, Springer International Publishing, Cham, 2020, pp. 422–430.
- [24] A. H. Julian Tritscher, Fabian Gwinner, Daniel Schlör, Anna Krause, Open ERP System Data For Occupational Fraud Detection Julian, *Technical Report*, 2022.
- [25] R. R. Hoffman, S. T. Mueller, G. Klein, J. Litman, Metrics for explainable AI: Challenges and prospects, *arXiv preprint arXiv:1812.04608* (2018).
- [26] K. Beckh, S. Müller, M. Jakobs, V. Toborek, H. Tan, R. Fischer, P. Welke, S. Houben, L. von Rueden, Explainable machine learning with prior knowledge: an overview, *arXiv preprint arXiv:2105.10172* (2021).

A. Convolutional Neural Network Architecture

Table 5 describes the architecture of our Convolutional Neural Network we used for section 3.

Table 5

CNN architecture used for our experiment. We found a large kernel size in the first two layers beneficial for the chess domain. The padding values lead to “center padding”.

layer	type	size	(padding)
1	2D Convolution	(12, 128, 7)	3
2	2D Batch Normalization	128	
3	ReLU		
4	2D Convolution	(128, 128, 5)	2
5	2D Batch Normalization	128	
6	ReLU		
7	2D Average Pooling	2	
8	2D Convolution	(128, 128, 3)	1
9	2D Batch Normalization	128	
10	ReLU		
11	2D Convolution	(128, 128, 3)	1
12	2D Batch Normalization	128	
13	ReLU		
14	2D Average Pooling	2	
15	Flattening Layer		
16	Linear Layer	(512, 3)	