Explaining Intelligent Game-Playing Agents

by

Aðalsteinn Pálsson

Dissertation submitted to the School of Technology, Department of Computer Science
at Reykjavík University in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

October 2024

Thesis Committee:

Yngvi Björnsson, Supervisor
Professor, Reykjavík University, Iceland

María Óskarsdóttir, Committee Member
Associate Professor, Reykjavík University, Iceland

Stephan Schiffel, Committee Member
Assistant Professor, Reykjavík University, Iceland

Tómas Philip Rúnarsson, Committee Member
Professor, University of Iceland, Iceland

Mark Winands, External Examiner
Professor, Maastricht University, The Netherlands

The undersigned hereby certify that they recommend to the School of Technology, Department of Computer Scienceat Reykjavík University for acceptance this Dissertation entitled **Explaining Intelligent Game-Playing Agents** submitted by **Aðalsteinn Pálsson** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy (Ph.D.) in Computer Science**

.....
date

.....
Yngvi Björnsson, Supervisor
Professor, Reykjavík University, Iceland

.....
María Óskarsdóttir, Committee Member
Associate Professor, Reykjavík University, Iceland

.....
Stephan Schiffel, Committee Member
Assistant Professor, Reykjavík University, Iceland

.....
Tómas Philip Rúnarsson, Committee Member
Professor, University of Iceland, Iceland

.....

Mark Winands, External Examiner
Professor, Maastricht University, The Netherlands

.....
date

.....
Aðalsteinn Pálsson
Doctor of Philosophy

Explaining Intelligent Game-Playing Agents

Aðalsteinn Pálsson

October 2024

Abstract

Artificial intelligence (AI)- based systems increasingly affect our daily lives. Such intelligent computer agents are becoming increasingly complex; for example, they employ learned machine learning models and extensive lookahead search, often exploring millions of possibilities. Unfortunately, as the complexity of those systems grows, it becomes more difficult to understand the rationality behind their decisions.

In this dissertation, we researched how to incorporate explainability into the game-playing domain. Furthermore, we also evaluated to what extent the game-playing domain suits the development of explanations, where we showed that using the game-playing domain enables quantification of many aspects that otherwise are costly or infeasible. We used three games as our testbed: Sudoku, Breakthrough, and Chess. We primarily focused on model explainability, with a secondary focus on the search part of game-playing.

For model explainability, we investigated three approaches for explaining the evaluation of the model: saliency maps, surrogate models and concept probing. First, we developed evaluation methods for saliency maps to understand the reliability of the methods and quantified to what extent we can interpret them in a way we are likely to do. Furthermore, we also introduced a second explainability layer using a surrogate model, where we explain the explanation. This way, we can, on a higher level, interpret what a high saliency means for the explainability methods. Second, we developed methods to compare concept probing results and assess to what extent we can interpret the results as concept importance. We showed that concept probing results and concept importance are only moderately correlated. Third, we unveiled the concepts learned by the world-class chess-playing agent, Stockfish, and exposed differences between its neural network and hand-crafted methods.

On the search front, we altered a search-based reasoning process to generate solutions that were more explainable to humans, using the domain of Sudoku puzzles as our test bed. It is a hybrid of a heuristic- and constrained-based solver that biases the search towards finding solutions easily explainable to humans.

Keywords: Artificial Intelligence, Explainability, Game-playing, Neural-Networks, Chess

Útskýringar fyrir Gervigreinda Leikjaspilunar Agenta

Aðalsteinn Pálsson

október 2024

Útdráttur

Tölvukerfi sem byggja á gervigreind eru í síauknu mæli að hafa áhrif á okkar daglega líf. Flækjustig slíkra kerfa er einnig sífellt að aukast, m.a. vegna þess að þau nota oft flókin líkön (s.s. tauganet) og meta milljónir mögulegra lausna við ákvarðanatöku sína. Eftir því sem flækjustigið eykst, því erfiðara verður fyrir okkur að skilja rökin að baki ákvörðunum kerfanna, sem getur haft áhrif á það traust sem við berum til þeirra.

Í þessari ritgerð könnuðum við hvernig hægt er að innleiða útskýringar fyrir leikjaspilunar agenta. Ennfremur mátum við hve vel leikjaspilunaragentar henta fyrir þróun útskýringa og sýndum fram á að með þeim er hægt að mæla marga þætti sem annars eru kostnaðarsamir eða óframkvæmanlegir. Við notuðumst við þrjá leiki: Sudoku, Breakthrough, og skák. Við einbeittum okkur fyrst og fremst að útskýringum líkana, með aukaáherslu á leit.

Við rannsökuðum aðallega þrjár tegundir af útskýringum: "saliency maps", "surrogate model" og "concept probing". Við þróuðum matsaðferðir til að meta áreiðanleika "saliency maps" og mátum hvort við gætum túlkað niðurstöður þeirra á þann hátt sem við mannfólkið erum líkleg til að gera. Ennfremur þróuðum við annað útskýringarlag með "surrogate model", þar sem við útskýrum útskýringuna. Þannig getum við betur áttað okkur á því hvað felst í því þegar útskýringaraðferð metur eitthvað mikilvægt. Til að skilja betur notagildi "concept probing" aðferða þróuðum við aðferðir til að athuga hvort hægt sé að meta mikilvægi hugtakanna (e. concepts) út frá "concept probing" niðurstöðum. Þar sýndum við fram á fylgni milli mikilvægi hugtakanna og niðurstöðum "concept probing" tilraunanna. Þá afhjúpuðum við einnig þau hugtök sem heimsklassa skákagentinn Stockfish hafði lært, og vörpum ljósi muninn á þeirri þekkingu sem býr í tauganeti og handgerðu matslíkani.

Hvað leitarhlutann varðar, þá breyttum við leit agents svo hann skili frá sér lausnum sem er auðveldara fyrir okkur mannfólkið að skilja. Það gerðum við með blöndu af brjóstsvits- og skorðunarnálgun sem aðlagar leitina svo agentinn taki ekki ákvarðanir sem er erfitt eða ómögulegt að útskýra.

Efnisorð: Gervigreind, Útskýringar, Leikjaspilun, Tauganet, Skák

I dedicate this to my son.

Acknowledgements

First and foremost, I would like to express my deepest appreciation to my supervisor, Yngvi Björnsson. His guidance, expertise, and unwavering support have been invaluable throughout my research. Yngvi's insightful feedback and constant encouragement have helped me grow both academically and personally. He has been exceptionally generous with his time and always willing to engage in thoughtful discussions, and for that, I am profoundly grateful.

I would also like to acknowledge Sigurður Helgason, with whom I collaborated early in my research. I am truly grateful for his collaboration and commitment.

I extend my sincere gratitude to the Department of Computer Science for providing a great academic atmosphere and support. The sense of community within the department has been invaluable.

I am also deeply thankful to my family. Without their love and support, this would not have been possible.

Lastly, I would like to extend my heartfelt gratitude to my girlfriend, Stella Rún. Her presence and support during the final stages of this journey have been truly wonderful, making the completion of this thesis a memorable and joyous experience.

Preface

This dissertation is original work by the author, Aðalsteinn Pálsson. In this thesis we summarize the work conducted between September, 2019, to Oktober, 2024.

The document is formatted in a *collection of articles*. It includes the following articles:

A) **A. Pálsson and Y. Björnsson**, "Evaluating interpretability methods for dnns in game-playing agents," in Advances in Computer Games - 17th International Conference, ACG 2021, Virtual Event, November 23-25, 2021, Revised Selected Papers, C. Browne, A. Kishimoto, and J. Schaeffer, Eds., ser. Lecture Notes in Computer Science, vol. 13262, Springer, 2021, pp. 71–81.

B) **A. Pálsson and Y. Björnsson**, "Unveiling concepts learned by a world-class chess-playing agent," in Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China, ijcai.org, 2023, pp. 4864–4872.

C) **A. Pálsson and Y. Björnsson**, "Empirical Evaluation of Concept Probing for Game-Playing Agents" (accepted for publication at the 27th European Conference on Artificial Intelligence (ECAI-2024))

D) **Y. Björnsson, S. Helgason, and A. Pálsson**, "Searching for explainable solutions in sudoku," in 2021 IEEE Conference on Games (CoG), Copenhagen, Denmark, August 17-20, 2021, IEEE, 2021, pp. 1–8.

All code related to the publications is publicly available in the repository: `https://github.com/adalsteinnpals/Explaining-Intelligent-Game-Playing-Agents`

Contents

Acknowledgements xi

Preface xiii

Contents xiv

List of Figures xvi

List of Tables xvii

1 Introduction 1
- 1.1 Explainable AI . . . 2
- 1.2 Explainability in Game-Playing . . . 3
- 1.3 Aim and Research Questions . . . 3
- 1.4 Contribution of Thesis . . . 5
- 1.5 Thesis Structure . . . 5

2 Background 7
- 2.1 Game-Playing . . . 7
  - 2.1.1 Games . . . 7
    - 2.1.1.1 Breakthrough . . . 7
    - 2.1.1.2 Chess . . . 8
    - 2.1.1.3 Sudoku . . . 9
  - 2.1.2 Game-Playing Agents . . . 9
    - 2.1.2.1 Stockfish . . . 10
    - 2.1.2.2 AlphaZero . . . 13
- 2.2 Explainability . . . 13
  - 2.2.1 Taxonomy . . . 14
  - 2.2.2 Surrogate Models . . . 15
  - 2.2.3 Shapley Values . . . 16
  - 2.2.4 Saliency Maps . . . 16
    - 2.2.4.1 Occlusion . . . 16
    - 2.2.4.2 Gradient Methods . . . 17
    - 2.2.4.3 LIME . . . 18
    - 2.2.4.4 Deeplift . . . 18
    - 2.2.4.5 Shapley Value Sampling . . . 18
    - 2.2.4.6 Challenges Related to Saliency Maps . . . 18
  - 2.2.5 Concept Based Explanations . . . 19
    - 2.2.5.1 Concept Probing . . . 19
    - 2.2.5.2 TCAV . . . 20

2.2.5.3 Challenges Related to Concept Probing 21
2.2.6 Evaluation of Explanations 22
2.3 Explainability in Games 23
2.4 Conclusions 25

3 Summary of Contributions **27**
3.1 Understanding and Evaluating Explanations 28
3.1.1 Motivation and Goal 28
3.1.2 Contribution 28
3.1.3 Methods 29
3.1.4 Results 31
3.1.5 Summary 32
3.2 Unveiling What Game-Playing Agents Have Learnt 37
3.2.1 Motivation and Goal 37
3.2.2 Contribution 37
3.2.3 Methods 38
3.2.4 Results 40
3.2.5 Summary 41
3.3 Explaining Search 43
3.3.1 Motivation and Goal 43
3.3.2 Contribution 43
3.3.3 Methods 44
3.3.4 Results 45
3.3.5 Summary 45

4 Related Work **49**
4.1 Acquisition of Chess Knowledge in AlphaZero 49
4.2 Other Related Work in Game-Playing 51

5 Conclusion and Future Work **55**
5.1 Conclusion 55
5.2 Future Work 58

Bibliography **61**

A Evaluating Interpretability Methods for DNNs in Game-Playing Agents **69**

B Unveiling Concepts Learned by a World-Class Chess-Playing Agent **81**

C Empirical Evaluation of Concept Probing using a World-Class Game-Playing Agent **91**

D Searching For Explainable Solutions in Sudoku **101**

E Supplementary Material **111**
E.1 Training a AlphaZero-like agent to play Breakthrough 111
E.2 Hiding Concepts from a Stockfish Agent 112

List of Figures

2.1 Breakthrough: Initial board position (left); Example position (right). . . . 8
2.2 Chess: Initial board position [9]. . . . 8
2.3 Sole candidate. . . . 9
2.4 Stockfish' NNUE architecture 12
2.5 A visualization of four dimensions of neural network interpretability, extended from Zhang et al.[22]. . . . 15
2.6 Example of a saliency map using occlusion. . . . 17
2.7 A simplified figure of a probe probing for if white has a mate threat. . . . 20
2.8 The taxonomy of evaluation methods. Figure 1 from Doshi-Velez and Kim [48]. . . . 22
2.9 Example of a saliency map from Puri et al. [51]. . . . 24
2.10 What-when-where plots from Figure 2 in McGrath et al. [52]. . . . 24
2.11 Results from a behavioral test from Figure 5a in Lovering et al. [53]. . . . 25

3.1 Effect of gradually removing the least-important pieces. . . . 33
3.2 The visualization displays the distribution of saliency values assigned to the piece that secures the winning move 4-ply before the win. . . . 34
3.3 The average Shapley Values of each of the inputs to the interpretable surrogate model. . . . 34
3.4 The relationship between probing accuracy and importance. . . . 35
3.5 Probing accuracy results performed during training of the autoencoder. . . . 36
3.6 Estimated piece values using the weights of a linear surrogate model. . . . 40
3.7 Concept probing results 41
3.8 Concept probing results per bucket. . . . 41
3.9 Chess positions for qualitative analysis. . . . 42
3.10 Best cost found per difficulty level. . . . 46
3.11 Unsolved Sudoku board with 10 remaining moves. . . . 46

4.1 Figure 7 from McGrath et al. [38]. . . . 50
4.2 Figure 3 from Tomlin et al. [56]. . . . 51
4.3 Figure 1 from Schut et al. 2023 [57]. . . . 52
4.4 Figure 4 from Hammersborg and Strumke, 2022 [58]. . . . 53
4.5 Figure 1 from Ruoss and Delétang [59]. . . . 53

E.1 Visualization of the training framework 111

List of Tables

1.1 An overview of the research question and topic of each paper. 4

2.1 A comparison between AlphaZero and Stockfish. 10
2.2 List of hand-crafted concepts used by the classical evaluation model of Stockfish. 12
2.3 A table from [21] showing the categorization of the objective of different stakeholders. 14

3.1 Description of concepts used by The Explaining the Explanation method. . 31
3.2 Win ratios after removing highest and lowest-ranked piece according to each explainability method. 33
3.3 Pearson correlation between concept probing results and importance. . . . 33
3.4 Description of Δ piece value concepts. 38
3.5 Shapley values of hand-crafted concepts. 39
3.6 Strategy specific cost, w_m. 44
3.7 Least costly path found to solve the board in Figure 3.11. Cost 33. 45
3.8 Most costly path found to solve the board in Figure 3.11. Cost 342. 47

E.1 Hyperparameters used to train Stockfish Agents 113

Chapter 1

Introduction

This thesis is about Artificial Intelligence (AI) and explaining. AI has become ubiquitous; touching most areas of our daily lives. We interact with algorithms through chatbots, professionals use them for assistance, and they run much of our infrastructure [1].

Artificial Intelligence and its subfield, Machine Learning, has an emerging area of research that aims to create self-explainable algorithms and develop techniques to help decipher otherwise non-understandable algorithms. This area is called Explainable Artificial Intelligence (XAI). The interest in XAI has steadily increased since the publication of the first breakthrough deep learning model, Alexnet [2], published in 2012. However, explanations have also historically been a research topic for philosophers who have tried to define the theoretical foundation of explanations.

We currently use AI algorithms in many high-stakes situations where trust, transparency, and reliability of the algorithms are crucial to understanding the ultimate tradeoffs of AI automation. We currently deploy automated decision-making in areas such as health care, finance, and criminal justice and manual task automation such as driving vehicles - all of which impact people's lives. When humans operate in these situations, our legal and regulatory system has defined appropriate responses when things go wrong. However, concerning AI automation, the regulatory system is far behind, and accountability needs to be well-defined when things go astray. These issues are currently being addressed globally, with efforts such as the AI Act [3] from the European Union. Even though automation may result in more efficiency or higher accuracy than manual undertaking, the circumstances where it goes astray may be due to unwanted or indecipherable behavior of the model. In these situations, there is often a tradeoff between explainability and accuracy. More complex models frequently outperform simpler, more explainable models, where the causality of the decision is more transparent. The tasks and the models we use to automate are becoming increasingly complex, making them harder to interpret.

Multiple issues can arise, even when we are deploying models that outperform any other predecessor. For example, issues related to bias and fairness, particularly concerning gender or race, are especially sensitive when automating decisions that may have had historical discrimination [4]. There are also issues related to data quality and quantity; here, the data may not correctly represent out-of-sample distribution and thus can give an inaccurate idea of how the model will perform in the real world. Is the model robust, that is, albeit most often correct? Does it sometimes go astray? How can the model decisions be explained in meaningful and human-understandable terms when that happens? Furthermore, in cases where models outperform domain experts,

we may seek to develop methods to decipher the model's reasoning and identify its acquired knowledge to increase our own understanding of the domain. All of these methods fall into the category of generating explanations of the model behavior.

1.1 Explainable AI

So, what constitutes a good explanation? This topic is somewhat complicated because, although fundamental to us, we can argue that even humans are often quite bad at explaining the decisions they arrive at; for example, we might make decisions based on intuition, and when challenged to explain why, we might be inclined to refer to reasons we did not consider at the time of the decision. Explanations play a big role in our communication and influencing other people, Lombrozo [5] states: "Explanations are more than a human preoccupation – they are central to our sense of understanding, and the currency in which we exchange beliefs." When we seek explanations from a model, we seek to fit some mental model about the reasoning behind the decision. Kim et al. [6] elegantly defines this as a mapping from model vector space, E_m, to human vector space, E_h. E_m operates in the space of features, such as image pixels, while E_h spans high-level, human-understandable concepts. Theories from philosophy, such as the Deductive-Nomological Model [7], address that when we explain, we seek a logical proof or to provide a mechanism behind the explanation. What results is an intriguing phenomenon: The Symmetry Thesis [8], which states that prediction and explanation differ only in terms of when they are given, meaning they should adhere to the same structure. This idea that an explanation should be used to build a proper argument is very appealing. Still, in practice, the kind of explanations developed in the field of Explainable AI (XAI) are usually quite different. When explaining data, models, and results from the model, there are multiple areas of focus from the explainability community. For example, the sheer amount of data, features, solutions, or combinations explored to arrive at an answer can be so much that it is impossible to create a mental model of the problem. Solving this is usually approached by abstracting to a more understandable form, either by highlighting relevant data or intelligently presenting it.

In this thesis, we focus primarily on model explainability, aiming to decipher the reasoning and knowledge acquired by deep neural networks. Due to their intricacy, deep neural networks are usually considered black boxes, i.e., concealed functions with inputs and outputs. An alternative approach is to develop more inherently explainable models, which we briefly explore in this thesis by restricting an agent to consider explainable actions only. The approaches this research explores to explain neural networks fall into two categories. First, we will consider saliency maps (covered in Section 2.2.4), a method of visually highlighting the importance of different model elements, usually applied to the input. The resulting explanation from these methods is a visualization of the input with a transparent overlay indicating the importance of each input feature. Second, we develop explanations using high-level, human-understandable concepts. Deep neural networks (DNN) learn intermediate representations of the data before predicting. Thus, we might be curious to determine if the DNN represents some human-understandable concepts in its intermediate representations. *Concept probing* (covered in Section 2.2.5.1) is an example of a method used to detect the presence of human-understandable concepts.

1.2 Explainability in Game-Playing

We use the domain of game-playing for this research, which is an intriguing area of research for explainability. For the first part, we have an extensive history of explanations from the literature where the explanations have even been tailored to each user's knowledge. That is, we can only explain something in terms that the user understands. Furthermore, from a theoretical point of view, the game-playing domain has many interesting research opportunities. Such as: i) the decisions are made by combining a model and a search algorithm, ii) we have a closed system where there are no uncontrolled confounders, iii) decisions are (usually) not made by one observation but multiple ones, iv) in adversarial games we need not only explain the reasoning of a single agent, but the adversary as well, v) explanations can be validated by observing future states, and vi) it is a domain where humans learn abstract concepts, and thus it is of particular interest to determine if the super-human state-of-the-art agents are using similar concepts or learning something new.

More specifically, the explainability problems we address in this thesis focus on developing explanations and ways to evaluate if the explanations are trustworthy. A significant issue regarding explanations is ensuring that they are correctly interpreted, e.g., does the user have all the necessary knowledge to interpret the explanation?. There are many scenarios where explanations are non-conclusive, fail to expose valuable insight, and are even subject to confirmation bias. Thus, they are often hard to rely on in cases where one would like explanations to help build trust. Furthermore, one would prefer the explanations to shed light on the model's mechanism in such a way that it is also helpful to identify false positives.

In this research, we focus specifically on three games: Breakthrough, Chess, and Sudoku. The characteristics of each make for an ideal test bed for each problem we address in the thesis. Breakthrough, a pawn racing game, has a well-defined strategy, and its simplicity makes it well-suited for analysis of saliency maps. Chess has a rich literature of theoretical and teaching material. The concepts from the literature have been implemented as evaluation functions, which makes chess an ideal test-bed for concept-based explanations. And finally, Sudoku is a single-player game with simple rules, which is naturally formalized as a constraint satisfaction problem (CSP), which makes it a good starting point to explain the search process. An interesting contrast exists between how humans and computers solve Sudoku, which makes for exciting challenges because humans solve Sudoku with minimal backtracking. At the same time, CSP solvers rely heavily on it.

1.3 Aim and Research Questions

The thesis had two primary goals. Firstly, we research how we can incorporate explainability into the game-playing domain. And secondly, how can the game-playing domain help us understand the explainability methods better. In this thesis we approached that in a few different ways. We will be developing and assessing explanations, as well as developing explainable agents. Furthermore, we will also expose the knowledge that the agents have learned.

We leverage the unique opportunities of our domain to define and address domain-specific problems, but the insight may be more broadly applicable. The research questions addressed are the following:

Table 1.1: An overview of the research question and topic of each paper.

Paper	Overall RQ	Topic
A	RQ1, RQ3	Breakthrough, Saliency Maps
B	RQ3	Chess, Unveiling Learned Concepts
C	RQ1	Chess, Concept Probes
D	RQ4	Sudoku, Explainable Search

- RQ1: How effective/reliable are commonly used saliency map explainability methods in the game-playing domain?
- RQ2: How to use explainability methods to unveil the concepts learned by game-playing agents?
- RQ3: What are the pitfalls and best practices in interpreting concept probing results in the game-playing domain?
- RQ4: How to bias the search process of game-playing agents to make more human-understandable decisions?

This thesis is a compilation of four research papers, which we reference as Papers A-D.

The first paper, Paper A, aimed to understand how practical commonly used saliency map explainability methods are in our domain. For this paper, we trained (using self-play) an AlphaZero-like model that predicts an evaluation and a policy. We used these methods to assess how important both players' pieces are to our model's evaluation. The main research question was: *How effective/reliable are commonly used saliency map explainability methods in the game-playing domain?*

The second paper, Paper B, aimed to investigate how we can apply state-of-the-art explainability methods to unveil what human-understandable concepts a world-class chess-playing agent (Stockfish) has learned. The goal is to quantify and expose to what extent the agent's neural network uses known chess concepts when evaluating chess positions. The research question explored in this paper is: *How to use explainability methods to unveil the concepts learned by game-playing agents?*

The third paper, Paper C, investigated the reliability and best practices of concept probing in our domain. In the game-playing domain, researchers (including us in Paper B) have been applying linear probes and interpreting their results in their research. However, researchers have questioned their reliability, and some have suggested using more complex probes. In this paper, we investigate the suitability of concept probing in the game-playing domain and evaluate, for example, whether we can infer concept importance from concept probing results. The research question addressed in this paper is: *What are the pitfalls and best practices in interpreting concept probing results in the game-playing domain?*

In the fourth and final paper, Paper D, we address the concerns that search algorithm results do not sufficiently match human intuition. The aim is to make the search process and an agent's final decision more human-understandable while looking for an acceptable solution to the problem (quality wise). Such a problem-solving approach is relevant in settings where we expect a human to verify or execute the resulting solution and where a human hopes to learn from the computer's solving process. The research

question the paper explores is: *How to bias the search process of game-playing agents to make more human-understandable decisions?*

1.4 Contribution of Thesis

In this section, we give a brief overview of the contributions of the thesis; further details can be found in Chapter 3 and Papers A - D.

The main contributions of this thesis revolve around researching how XAI can be used in game-playing. However, we also identified some takeaways that may broadly apply to explainability research, first and foremost, by demonstrating the effectiveness of the game-playing domain as a test bed for explainability research and showing how well it is suited for building functionally-grounded evaluation methods that map well with how we are likely to interpret their results.

This research presents multiple evaluation methods that all share a distinctive trait. The evaluation methods utilize a unique aspect of the game-playing domain: We can look into the future and assess whether an explanation correlates with our anticipation of future events. That means we can create functionally grounded tests and evaluate explainability methods without relying on human feedback other than designing the evaluation test. Using the evaluation methods, we compare commonly used explainability methods and discover that some, such as Shapley Value Sampling, better align with our interpretation.

Furthermore, using game-playing, we can precisely evaluate the playing strengths of different agents by having them compete in a tournament. Thus, by selectively removing knowledge from the agents, we can quantify how important the removed knowledge is. Therefore, we have a foundation for quantifiable analysis of concept probing results. Using our method, we show the shortcomings of commonly used linear probes and suggest more reliable experiments to reveal feature importance from concept probing experiments.

We show how explainability methods can unveil the concepts learned by game-playing agents. We unveiled the strategy of Breakthrough using a surrogate method that explains the explanation; it interprets the saliency maps using higher-level concepts. Furthermore, we showcase some interesting human-understandable knowledge the agent Stockfish has learned. We exposed the difference between hand-crafted and learned concepts.

Using Sudoku, we develop an agent that optimizes its solutions for explainability. The solver is a depth-first algorithm guided by pre-defined human-like strategies and an explainability heuristic. Instead of searching for moves in model space, we showed how it is possible to constrain it to search in human-understandable space of concepts and pick the most easily explainable solution to present to the user (depending on the user's expert level).

1.5 Thesis Structure

The rest of the thesis is structured as follows: In Chapter 2, we cover the necessary background information, such as algorithms, games, and relevant terminology. In Chapter 3, we summarise in more detail the contributions made in the thesis. We summarize related work in Chapter 4. In Chapter 5 we conclude the document and

discuss future work. Then, in Appendixes A, B, C, and D, we list each of the four papers the thesis is based on. Finally, in Appendix E, we give the supplementary material.

Chapter 2

Background

This chapter covers relevant background information needed to understand the contributions of the thesis. In the first section, we will cover game-playing; in the second section, we will cover relevant Explainability AI terminology and methods; in the third and final section, we will cover related work on game-playing in the explainability domain. Further coverage of related work can be found in Chapter 4.

2.1 Game-Playing

In this section, we describing the games, agents, and search algorithms used in the thesis. The research utilized three games: Breakthrough, Sudoku, and Chess. The agents used are Stockfish, which relies on alpha-beta search, and AlphaZero, which uses Monte Carlo Tree search. First, we will describe the games, followed by the agents.

2.1.1 Games

In the thesis, we investigated three games: Breakthrough, Chess, and Sudoku. We describe the games in the following subsections.

2.1.1.1 Breakthrough

The game Breakthrough is an abstract strategy board game, originally played on a 7x7 board but later popularized to an 8x8 board. The game can be played on different-sized (not necessarily squared) boards. In this work we use a smaller variant of 5x6, mainly for the ease of demonstration.

The game is a two-player turn-taking game. The players are referred to as White and Black. The board is initially set up by placing White's pieces along the first two rows and the Black pieces on the last two rows, as shown in Figure 2.1 (left). White goes first and then players alternate, with each player moving one of their pieces per turn. A piece moves one square straight or diagonally forward (relative to the player). A straight forward move is allowed to an empty square only, but a diagonally forward moves may also capture an opponent's piece. For example, in Figure 2.1 (right) the white pawn on $d2$ has two moves, to $d3$ or $e3$, and the piece on $d4$ also has two moves, to $c5$ or $e5$, both with a capture. The first player to get a piece across the board wins: White wins by moving a piece onto the last row, and likewise, Black wins by moving

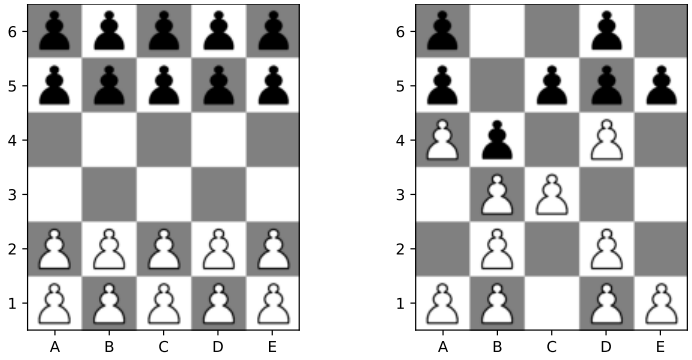


Figure 2.1: Breakthrough: Initial board position (left); Example position (right).

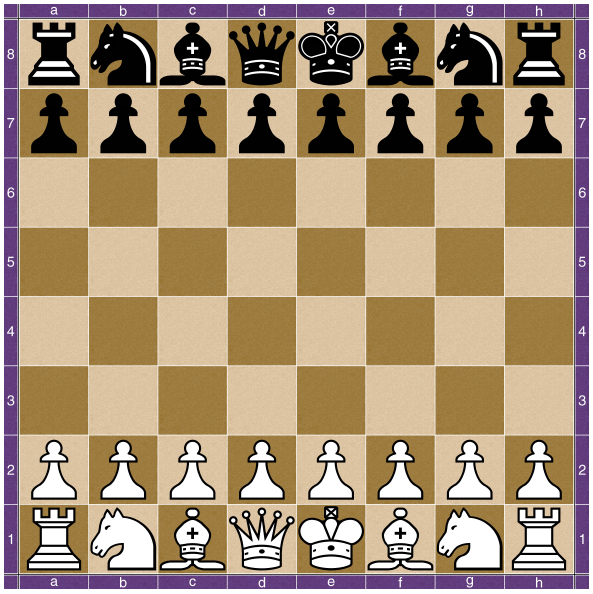


Figure 2.2: Chess: Initial board position [9].

a piece onto the first row. If all pieces of a player are captured, that player loses. It follows from the rules that one of the players always wins (there are no draws).

2.1.1.2 Chess

Chess is a two-player, perfect information turn-taking game [9], although the game has relatively straighforward rules, there are complex strategies. The initial board position can be seen in Figure 2.2. Unlike the game Breakthrough, Chess can end in a draw. The objective is to checkmate the opponent's king, but there also other ways that the game can end, such as in a stalemate or due to the three-fold repetition rule. Due to the popularity of chess we will not cover the rules of the game in the thesis (but they can be found here [10]).

One of the aspects of chess that makes it interesting for XAI is that it has centuries of theory literature, where chess experts define and explain abstract (chess) concepts to other players of varying levels of expertise.

The first computer agents used hand-crafted heuristic functions accompanied by sophisticated search algorithms. This approach ultimately resulted in superhuman playing strength and implemented much of the chess theory from the literature. Con-

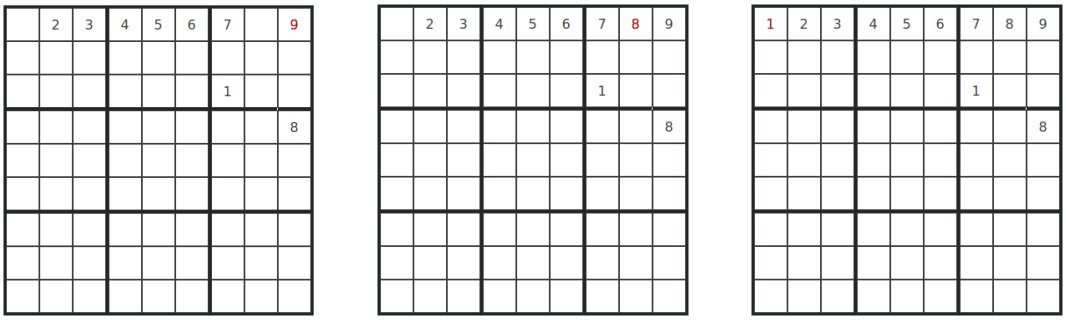


Figure 2.3: Sole candidates: In the topmost row: 9 is a sole candidate (left), 8 becomes a sole candidate (middle), and finally the 1 becomes one too (right).

veniently, in XAI research, we can now use these heuristic functions as a proxy for human-understandable chess concepts.

A competition was formed in 1974, called World Computer Chess Championship [11], where chess agents compete against each other. Nowadays, the Top Chess Engine Championship (founded in 2010) is regarded as the unofficial world championship for chess engines, where Stockfish [12] (which we use in this thesis) is the reigning champion.

2.1.1.3 Sudoku

Sudoku is a single-agent puzzle game. The game of Sudoku is played on a rectangular grid, usually at order 3, meaning that the grid is of size $3^2 \times 3^2 = 9 \times 9$, and overlaid by mutually exclusive boxes (also called blocks or regions) of size 3×3 (see Figure 2.3). We interchangeably refer to grid cells as variables, as this is how they would be represented in the CSP formalism. The domain of each variable (grid-cell) is 1 to 9 (3^2).

Initially, a partially filled puzzle is provided and the task of the player is to fill out the remaining grid cells such that each number occurs exactly once in each row, column, and box. Sudoku puzzles are traditionally generated such that they have exactly one solution. The puzzles can be of a varied level of difficulty, ranging from being easily solved by humans using only trivial deduction techniques to requiring quite sophisticated levels of reasoning.

For example, the simplest one, called *sole candidate*, as seen in Figure 2.3, is a method where a candidate number is assigned to a square by eliminating all other candidates.

2.1.2 Game-Playing Agents

Game-playing agents for abstract board games, like chess and checkers (and others), almost universally employ both a search and an evaluation component for making their move decisions. The former encapsulates the thinking ahead process, exploring various possible continuations of play. In contrast, 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

Table 2.1: A comparison between AlphaZero and Stockfish.

	AlphaZero	Stockfish
Neural network size	Large	Small
Outpur	Value and Policy	Value
Search method	Monte Carlo Tree Search	Alpha-beta pruning
Training	Self-play	Supervised learning
Number of positions evaluated per move	20,000	20,000,000
Accessibility	Closed-source	Open-source

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).

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 game-play considerably. These DNNs learn from training data, which are either engineered or generated (and possibly simultaneously learned from) via self-play. One approach is to train the model to predict a scalar value v, which estimates the expected outcome of the game. The training data is, therefore, state and target pairs, where the target can either be the game's final outcome or, alternatively, utilize an evaluation of a secondary (strong) game-playing agent given significant search depth. The scalar value can then guide the search process (e.g., using alpha-beta search) and suggest the strongest move. Alternatively, the agent can additionally learn to predict move probabilities, p, giving a probability distribution over possible moves[13]. The move probabilities can then assist by guiding the search process (e.g., Monte Carlo Tree Search) .

We investigated two game-playing agents in this thesis: Stockfish, a chess agent trained to only predict a value estimation, v, using supervised learning, and AlphaZero, which outputs both a value estimation, v, and move probabilities, p, trained using self-play.

For this thesis we trained AlphaZero on very simplified chess-like game called Breakthrough, training it to play regular chess is extremely hardware intensive.

The other agent, Stockfish, it is currently the strongest chess agent in existence, but is quite different from AlphaZero.

Stockfish is very computationally efficient, it is also open source, and trained using supervised learning and with all training data available online. And due to its small size and training procedure, we are able to train it on just a single desktop computer - making it the perfect test bed for the type of experiments we conduct in this thesis. Further comparison between the agents can be seen in Table 2.1. In the following subsections, we will further describe the two agents.

2.1.2.1 Stockfish

Stockfish [12] 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. It relies on the same alpha-beta search using both methods. 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.

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.:

$$f_{classical}(s) = \sum_{i=1}^{N} w_i \times c_i(s) \tag{2.1}$$

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 2.2, 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.

Stockfish' Neural Network Stockfish (since version 12) uses a neural network called NNUE [15] (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) [14], [16].

[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.

Table 2.2: 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.

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

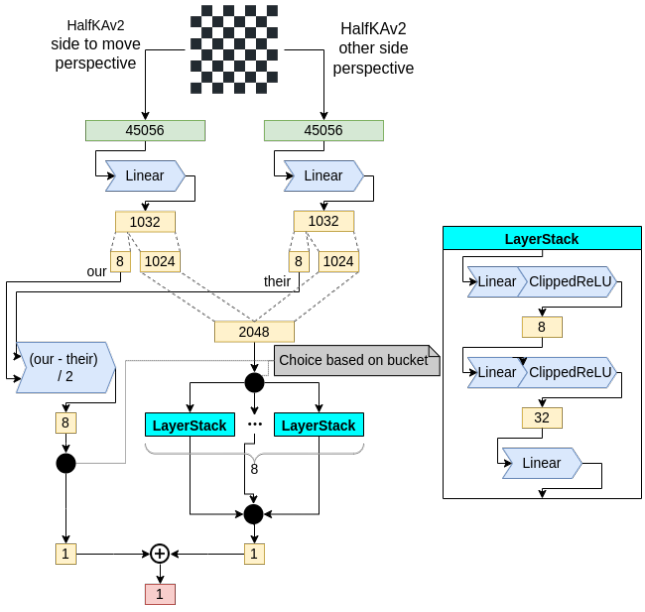


Figure 2.4: Stockfish' NNUE architecture [14]. Depending on the game phase (8 buckets, depending on the number of pieces) it will route the calculations differently through the network.

The NNUE architecture uses a (shallow) design with linear and clipped ReLU layers, as depicted in Figure 2.4. 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.

2.1.2.2 AlphaZero

AlphaZero [13] is the successor of Deepmind's AlphaGo [17]; it is a more generalized approach than its predecessor and can achieve superhuman performance in several games.

Given the game state s and neural network f with parameters θ we get

$$(\boldsymbol{p}, v) = f_\theta(s) \tag{2.2}$$

where $\boldsymbol{p}$ is a vector of move probabilities $p_a = Pr(a|s)$ for moves a, and v is a scalar value estimation that estimates the expected outcome z from position s, i.e. $s, v \approx \mathbb{E}[z|s]$. The loss, l, for updating the network parameters is

$$l = (z - v)^2 - \boldsymbol{\pi}^T log\boldsymbol{p} + c||\theta|| \tag{2.3}$$

where c controls the L_2 weight regularization. This algorithm is trained on games generated via self-play, where each move is chosen by Monte Carlo tree search (MCTS) [18]. Each game starts at the initial board position, then by repeatedly executing a MCTS to choose the next move until the game is terminated.

Each round of MCTS consists of a specified number of simulations or using a time-constraint, traversing through the search tree according to a formula called PUCT [19], selecting action a

$$a = argmax_a(Q(s, a) + U(s, a)) \tag{2.4}$$

where $Q(s, a)$ is the mean action value and

$$U(s, a) = C(s)P(s, a)\sqrt{N(s)}/(1 + N(s, a)) \tag{2.5}$$

where $N(s)$ is the parent visit count, $P(s, a)$ is the prior propability of selecting a in s and $C(s)$ is the exploration rate. All statistics are backpropagated to the root after each simulation, updating $N(s, a)$, $W(s, a)$, $Q(s, a)$ and $P(s, a)$ of all parent nodes, where $W(s, a)$ and $Q(s, a)$ are the total and mean action-values respectively.

After the game has finished, it is added to a replay buffer, from which game sequences are sampled during training to update the neural network.

2.2 Explainability

In this section, we will cover the necessary terminology and describe all relevant explainability methods used in the thesis.

An explanation can be defined as the interaction between three entities: an explanandum (what is to be explained), an explanator (which generates the explanations), and the explainee (the one to whom the explanandum is explained) [20].

The explanandum can, in theory, be any automated decision system, but in our case, we will focus primarily on neural networks.

The explanator, whose task is to explain to the explainee, gives insight or answers questions. The questions can be "how?" and "why?" - where generally the "how?" questions are inclined to focus more on giving some intuition on the *global* behavior of the model, generating explanations valid across all input instances. While the "why?" questions are usually directed at a single or a group of instances [20].

Table 2.3: A table from [21] showing the categorization of the objective of different stakeholders.

Target audience	Description	Explainability purposes	Pursued goals
Experts	Domain experts, model users (e.g. medical doctors, insurance agents)	Trust the model itself, gain scientific knowledge	Trustworthiness, causality, transferability, informativeness, confidence, interactivity
Users	Users affected by model decisions	Understand their situation, verify fair decisions.	Trustworthiness, informativeness, fairness, accessibility, interactivity, privacy awareness
Developers	Developers, researchers, data scientists, product owners...	Ensure and improve product efficiency, research, new functionalities...	Transferability, informativeness, confidence
Executives	Managers, executive board members...	Assess regulatory compliance, understand corporate AI applications...	Causality, informativeness, confidence
Regulation	Regulatory entities/agencies	Certify model compliance with the legislation in force, audits, . . .	Causality, informativeness, confidence, fairness, privacy awareness

The input into the explanator can be any combination of the following: the model, the data, user feedback, or context. On the other hand, the output is defined by three aspects: i) what is explained? ii) how is it explained? iii) how is it presented?

The explainee can be any person interacting with the AI system. For example, a developer or a non-technical person inside the same organization, such as an executive. The user can be inside or outside the organization developing the algorithm; even regulatory entities might seek explanations from the system. However, the reasons these persons seek an explanation broadly differ, as seen in Table 2.3. For example, a user might seek an explanation to understand their situation better, while a developer is motivated to ensure and improve the product efficiency.

In the following subsections, we will explore the different types of explanators and further cover the taxonomy of XAI algorithms. First, we will cover the taxonomy. Then, we will cover Shapley values, a concept many of the methods covered later rely on. We will cover the explainability methods used in the thesis in three sections: surrogate models, saliency maps, and concept-based explanations. Finally, we will explore approaches to evaluate the explainability methods.

2.2.1 Taxonomy

We can categorize model explainability methods in terms of four dimensions as shown in Figure 2.5. This categorization is the one introduced by Zhang et al. [22], we have extended it by one extra dimension (Dimension 4) to capture more distinctions that are relevant to the research.

The first dimension tells whether the method is active or passive. For an active approach we can actively change the architecture or training process to increase or aid in the interpretability of the model. For example, in this thesis we retrain Stockfish with a modified training set to shed a light on the importance of the removed data. Alternatively, we can develop post hoc methods that explain the decisions of already trained neural networks without modifying the model or the training process.

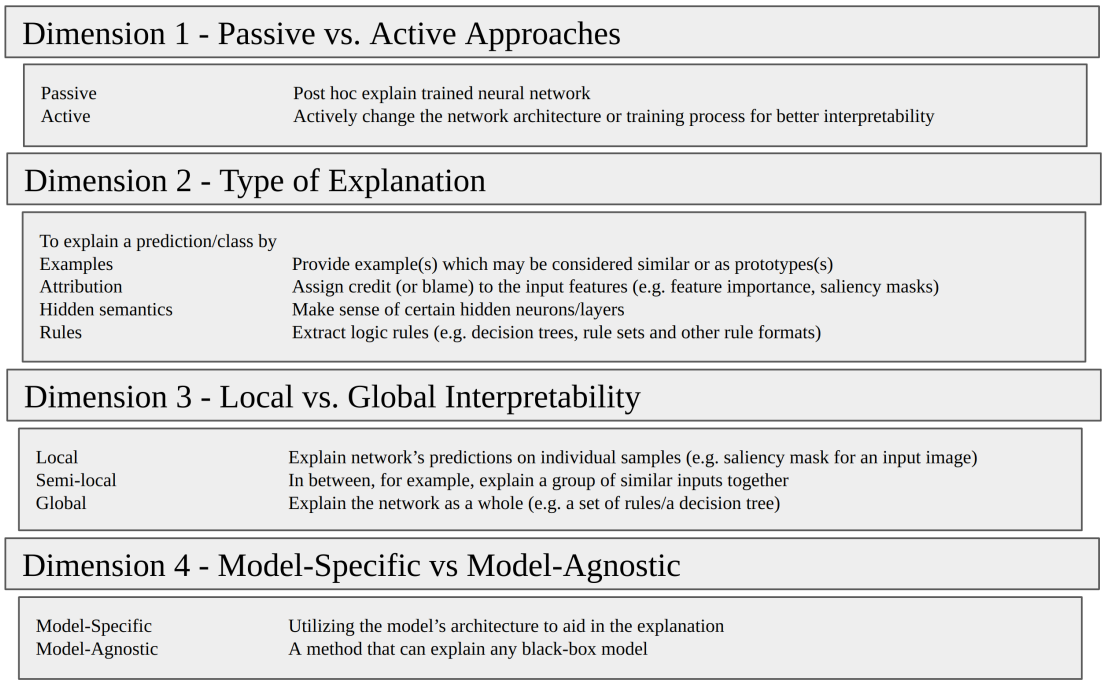


Figure 2.5: A visualization of four dimensions of neural network interpretability, extended from Zhang et al.[22].

The second dimension determines the building blocks of the explanation, that is, what we use to construct the explanation. Are we explaining using examples or assigning attribution to input features? Or are we extracting hidden semantics from hidden neurons or extracting logic rules? This dimension defines the type of explanation, i.e., what is presented to the explainee.

The third dimension tells whether we are explaining the local or global behavior of the model. A local explanation is only valid for a single decision, while a global explanation is valid across all instances. For example, if we are identifying how important a queen is in a single position, we are using a local explanation; however, when we assess how valuable, on average, the queen is for an agent (as we do in Paper B), we are applying a global explanation. Furthermore, it can also be semi-local if we simultaneously explain a group of similar inputs.

The fourth dimension tells whether the type of explanation is model-specific or model-agnostic. Model-agnostic methods explain any black-box models, while contrarily, model-specific methods may leverage the model's architecture to aid their explanations.

2.2.2 Surrogate Models

Dimension	Dimension 1	Dimension 2	Dimension 3	Dimension 4
Value	Passive	Attribution	Global/Local	Model-Agnostic

To interpret a black-box model, it is possible to train a interpretable model to mimic the behaviour of the black-box model, i.e., not predicting the true label, instead it predicts the output from the black-box. As for many other XAI methods, there exists

a Fidelity - Interpretability trade off, where a more complex model will better mimic the black-box model (resulting in a higher fidelity, further described in Section 2.2.6), but at the cost of interpretability.

2.2.3 Shapley Values

Dimension	Dimension 1	Dimension 2	Dimension 3	Dimension 4
Value	Passive	Attribution	Global/Local	Model-Agnostic

The Shapley value [23] estimates the marginal contribution of an agent, averaged over all possible coalitions. In the explainability domain we can treat features or concepts as agents, and evaluate their contribution as such.

The formula for Shapley values is:

$$\phi_i(N, v) = \frac{1}{N!} \sum_{S \subseteq N \setminus \{i\}} |S|!(|N| - |S| - 1)![v(S \cup \{i\}) - v(S)] \tag{2.6}$$

where ϕ is the Shapley Value, v is, in our case, the black-box function (neural network) that we aim to explain, and N is the set of all agents. Then, the Shapley Value is the normalized sum of the contribution that agent i adds to all possible sets S that don't contain i. Thus, when we use this method to estimate feature contribution (treating features as agents), it has two favorable properties: (i) if a feature does not affect the model prediction, its impact will be zero, and (ii) if two features similarly impact the model, their contribution will be similar.

This formulation can be extended to evaluate the Shapley Values of input features, concepts, and gradients, as described in the following sections. Generally it is a global method, but it can be applied to evaluate local contributions as well.

2.2.4 Saliency Maps

Saliency maps, also sometimes known as pixel attribution methods in image classification [24], are a form of an explanation which highlights visually the attribution of the input. In image classification we highlight input pixels, and when we apply this to games, we highlight the importance of squares or pieces, furthermore this method can also be used to visually highlight the importance of other aspects of the model such as neurons. Examples of a saliency maps for games can be seen in Figures 2.6 and 2.9. Although the presentation of the resulting explanations from saliency map methods is identical, the way these methods calculate the importance of each part of the input differ. Some consider the interplay between different parts of the input, while others do not. And some leverage the neural network architecture, while others do not.

2.2.4.1 Occlusion

Dimension	Dimension 1	Dimension 2	Dimension 3	Dimension 4
Value	Passive	Attribution	Local	Model-Agnostic

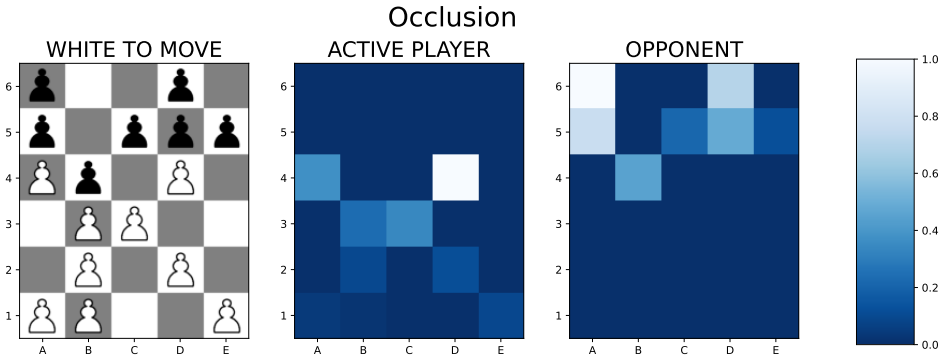


Figure 2.6: Example of a saliency map using occlusion.

The most straightforward local method is *occlusion*, where the model's output sensitivity to leaving out (zeroing) arbitrary input parameters is investigated [25]. Such an approach, where applicable, is appealing as it is both model-agnostic and straightforward to implement.

2.2.4.2 Gradient Methods

Dimension	Dimension 1	Dimension 2	Dimension 3	Dimension 4
Value	Passive	Attribution	Local	Model-Specific

A method that is local and model-specific but still requires no ad-hoc work is to analyze the gradient [26]. That is, evaluating the gradient with respect to the image at the input:

$$Gradient_i = \frac{\partial F(x)}{\partial x_j} \tag{2.7}$$

where x is the input, F is the neural network and j is the j-th dimension of input x. In practice, multiplying the input with the gradient is also often preferable because it leverages the strength of the input.

A further extension on the gradient method is Integrated Gradients [27], which relies on a baseline and interprets the input feature attribution as the integration of gradients on the straight-line path between the input and the baseline.

$$IntegratedGrads_j(x) ::= (x_j - x'_j) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_j} d\alpha \tag{2.8}$$

where α is the scaling coefficient. GradientShap [28] is an extension of Integrated Gradients, but instead of using a baseline of zero, GradientShap enables the use of a dataset as a baseline. It adds noise to the input, then samples a points between the input and reference/baseline and computes the gradient for those points. Thus, for a non-informative baseline, they arrive at very similar solutions.

2.2.4.3 LIME

Dimension	Dimension 1	Dimension 2	Dimension 3	Dimension 4
Value	Passive	Attribution	Local	Model-Agnostic

LIME (Local Interpretable Model-Agnostic Explanations) [29] is a local model-agnostic method, it uses an interpretable surrogate model to explain the black-box model, F. The procedure when explaining local behaviour around input sample x_i while using a linear surrogate model is as follows: First, sample data around x_i, second, evaluate sampled points using model F, third, train a linear surrogate model using the sampled points as input and the model evaluations as target. Finally, the weights from the linear surrogate model serve as the saliency values.

2.2.4.4 Deeplift

Dimension	Dimension 1	Dimension 2	Dimension 3	Dimension 4
Value	Passive	Attribution	Local	Model-Specific

DeepLIFT [30] is a model-specific explanation method that does not rely on the gradient. It overcomes the limitations of loss of information when the gradient is zero because the signal might still be meaningful. It calculates the importance in a backward fashion by distributing attributions, or blame, in terms of difference-from-reference. For all neurons, a difference-from-reference is calculated by passing through the input sample and the reference. Finally, it calculates the importance using predefined rules, such as the linear- or reveal cancel rule.

2.2.4.5 Shapley Value Sampling

Dimension	Dimension 1	Dimension 2	Dimension 3	Dimension 4
Value	Passive	Attribution	Global	Model-Agnostic

In practice, Shapley Values can be estimated using Shapley Value Sampling (SVS) [31], especially when it is infeasible to evaluate all possible combinations of contributions. SVS is implemented by randomly ordering the a list of input features N times. For each ordering we start with a baseline (e.g., with all values set as zero) and iteratively add one feature at a time. For every addition, we evaluate the impact or contribution of the newly added feature. Finally, the approximated Shapley Value is the average contribution of the feature across all random orders.

2.2.4.6 Challenges Related to Saliency Maps

One of the challenges related to saliency maps is that they are (typically [2]) only conditioned on one sample (local), thus they lack the generalization of other methods.

[2]There exist saliency methods such as GradientShap [32] that can be conditioned on a other samples

Furthermore, they are presented by the model input (such as image pixels), which is very low level of abstraction and thus usually not very useful for human intuition. Potentially due to that, saliency maps have been shown to be very misleading - we can interpret them overly confidently. One study showed, when users where given the task to classify concept importance based on saliency maps, it only resulted in 52% accuracy, although the users felt very confident [6]. Further doubts have been identified because when analyzing saliency maps of some randomly initialized neural networks, they appear quite similar to trained ones [6].

2.2.5 Concept Based Explanations

Concept based explanations aim to capture information at a high level of abstraction to better mimic human intuition. One example of such an explanation are the stripes of zebras in image classification or the concept of king safety in game-playing. In this section, we will cover two concept based explanation methods, and discuss general challenges related to concept probing.

2.2.5.1 Concept Probing

Dimension	Dimension 1	Dimension 2	Dimension 3	Dimension 4
Value	Passive	Hidden-semantics	Global	Model-Specific

A recent concept-based model-specific interpretability methodology for "peeking" into DNNs is *probing* [33]. Based on the intuition that DNNs 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. By training classification or regression surrogate models (on a dataset not used for training the network itself) — called probes — to predict a given concept from a layer's activations, one can measure how much information that layer carries regarding the given concept. The higher the prediction accuracy of the probe, the more information that layer carries for representing the concept. A simplified drawing of such probe can be seen in Figure 2.7.

When concept probing, we train a model or a probe g to map the intermediate representations of a neural network, f, to a concept z. We map representations $f_l(x)$ using input x evaluated at layer l to concepts z. As a proxy for how well f represents the concepts, one evaluates

$$performance\_metric(g(f_l(x)), z) \tag{2.9}$$

on unseen test data, and report the results using a relevant performance metric, such as accuracy. This means that an accuracy value of 0.5 infers layer l does not contain information regarding concept z. In contrast, an accuracy value of one indicates that concept z can be fully decoded from layer l in model f. The choice of probing architecture, g, will depend on the experiment and whether the concept is binary or continuous. Although researchers still debate the specific choices of probing architecture: some advocate for simpler linear probes, while others advocate for more complex ones, such as neural networks. The detailed nuances related to the specific choice of probing architecture will be discussed further in Section 2.2.5.3.

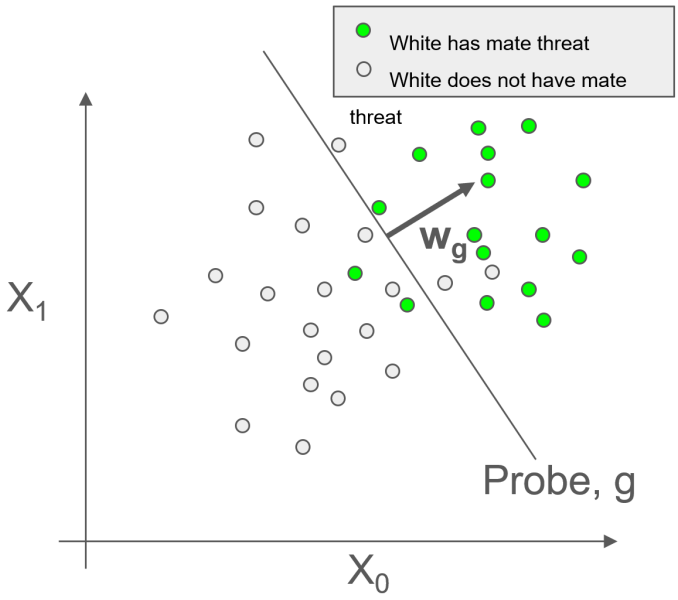


Figure 2.7: A simplified figure of a probe probing for if white has a mate threat.

The model training and probing process requires two separate datasets: the training dataset $D_t = \{x^i, y^i\}$ used to train the model, f, and a separate probing dataset $D_p = \{f_l(x^i), z^i\}$, used to train the probe g. Where $f_l(x^i)$ is the internal represetions in layer l for sample x^i and z^i is the concept's value for the corresponding sample. The probe is usually trained on a balanced dataset for binary concepts, meaning that the majority class is undersampled.

When designing the probing dataset D_p, a few things need to be considered, e.g., the diversity and amount of data needed. Similarly, as Voita and Titov [34], model the effort needed to achieve a particular probing accuracy, the amount of data needed to achieve the highest accuracy may vary. As one increases the size of D_p, the accuracy will increase until it saturates. Thus, when probing with a limited amount of data, comparing two probes for a fixed amount of training data may give misleading results if they represent two different points on a saturation curve.

2.2.5.2 TCAV

Dimension	Dimension 1	Dimension 2	Dimension 3	Dimension 4
Value	Passive	Hidden-semantics	Global	Model-Specific

A methodology introduced by Kim et al. [6], called Testing with Concept Activation Vectors, or TCAV, is a methodology where predefined concepts can be measured within an image classifier at prediction time. Using TCAV we can measure how relevant various concepts were to a given classification, e.g. how relevant stripes are to the classification of a zebra.

First we create the Concept Activation Vector, or the CAV, which, as described in the previous subsection, is the weights from a linear classifier. The classification of random samples and concept samples in the activation space of the model where the concept should be measured. Using the TCAV method we can measure the activation

of the concepts in any hidden layer of the model. The sensitivity $S_{Z,k,l}$ of classification of class k to concept Z, at layer l is then

$$\begin{aligned} S_{Z,k,l} &= \lim_{\epsilon \to 0} \frac{h_{l,k}(f_l(x) + \epsilon v_Z^l) - h_{l,k}(f_l(x))}{\epsilon} \\ &= \nabla h_{l,k}(f_l(x)) \cdot v_Z^l \end{aligned}$$

where $h_{l,k}$ is the logit for class k, at layer l and v_Z^l is the concept vector for concept Z in layer l. They also present an extension they call a relative TCAV, that is for when you seek to compare a concept that may be correlated or overlapping. E.g. when comparing three concepts Z_1, Z_2 and Z_3. Then it is possible, when training the v_{Z_1} to replace the previously random Negative set N, by setting the requirement $\{Z_2 \cup Z_3\}$ for the Negative set.

2.2.5.3 Challenges Related to Concept Probing

Researchers have arrived at contradicting conclusions on the kind of probe most useful for concept probing. Some have advocated for simpler probes [33], [35]–[38], while others have advocated for more complex probes [39]–[41]. The debate is mainly focused on what kind of results we can safely interpret; for example, it can be argued that too powerful a probe could learn its own mappings between the representations and the concept, and conversely, for too simple probes, it might not accurately reveal complex representation that the model has learned.

The underlying debate boils down to determining whether it is more meaningful that a model represents information linearly or non-linearly because the probes are not inherently faulty or incorrect. They are merely tools we don't yet fully know how to interpret.

To gain further insight into the probed model, some additional experiments have been proposed to identify to what extent the model is learning a concept. For example, evaluating a lower- and upper bound for the value has been proposed to put the probing accuracy's numerical value into perspective. A lower bound can be found by probing an untrained network; this can help identify a few concerns, for example, to help attribute to which extent the probing accuracy is because of model learning [39] rather than other factors.

The probing accuracy of concepts in untrained models is often remarkably high [39], which supports the findings that untrained models often work exceptionally well as feature extractors [42], [43]. This is, however, highly dependent on the architecture of the models; for example, a NN with fully connected layers does a better job at scrambling information than convolutional neural networks (CNNs). Meanwhile, CNNs preserve the structure of input a lot better.

An upper bound can be seen as a way to show how well, theoretically, a mapping from the input to the concept could perform; this can be achieved by training a dedicated model $h(x)$ to predict the concept z given the input, x. Although, in theory, this might seem like a viable option, one drawback is that it is likely very resource-intensive because the models required for such an experiment are likely larger than needed for regular probing and are more data-intensive. After all, they might require a similar amount of data as used to train the original model f.

If concept probing is a reliable tool for identifying the presence of concepts, in theory, it should be able to guide the removal of information from the network. However,

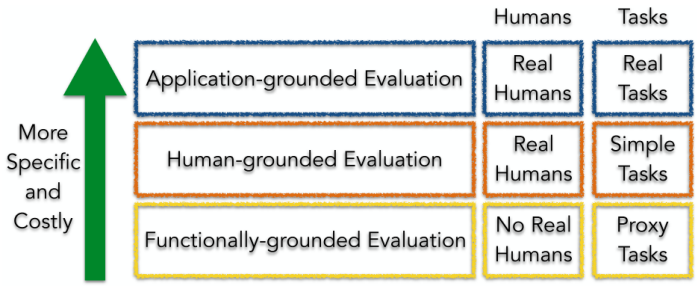


Figure 2.8: The taxonomy of evaluation methods. Figure 1 from Doshi-Velez and Kim [48].

the work in [44] demonstrate that it fails to remove the concepts entirely and, in some cases, destroys other task-relevant features because the probing classifier is likely to use non-concept features to predict the presence of the concept.

Although concept probing only measures how well model representations can be mapped to a concept, it is tempting to infer that the model is using the concept. However, researchers have found some evidence of false positives, meaning that high-accuracy probing does not necessarily mean that the concept benefits the task of model f [45]. In that respect, one attractive property of the game-playing domain we use is that we can quantify via self-play how useful a concept really is for a model.

To further investigate how to interpret these experiments, researchers have proposed various interventions that provide further contrast to the results; one such experiment is to erase the concept from the model training data (as we also do in this paper) $D_o = \{x^i, y^i\}$ [46]. By erasing the concept from the training data, we can monitor performance degradation on the original task, which puts the importance of the concept into perspective, as well as changes in probing accuracy. Interestingly, Ravichander et al. [46] showed that by removing a concept from the training data, the probe may still be able to predict some properties of the concept.

Elazar et al. [45] coined the term *amnesic probing*, an extension to conventional probing, to describe an approach for studying behavioral changes of a model after removing concept information from its representations using Iterative Nullspace Projection (INLP) [47]. They show that even after removal, subsequent layers can recover some of the removed properties which, to some extent, speaks to the importance of non-linear information within the model because INLP only removes its linear components. In this thesis we will refer to probing for concepts removed from the training set as *amnesic-like probing* due to its similarity with amnesic probing [47] – both method probe for removed information.

2.2.6 Evaluation of Explanations

The evaluation approaches can be coarsely split into three categories [48] (visualized in Figure 2.8):

(i) Functionally-grounded, where the explanation is evaluated without human, using a proxy as an indication of explainability; (ii) human-grounded, requiring a human with non-expertise to evaluate a simple explanation and; (iii) application-grounded, which requires a human-expert, evaluating an explanation for a real-world task.

The functionally-grounded approaches rely on metrics that measure the properties of the explanator. The field is still maturing, and there is no clear consensus about

terms to use, resulting in more than one term describing the same concept. In this research, we rely on the definitions found in Schwalbe and Finzel [20], the most extensive survey to date.

To measure how accurately an explainability method follows the explanandum (the black box), we measure its faithfulness (also known as fidelity, soundness, or causality, often used interchangeably). Thus, for a high faithfulness explanator, for example, when we use a surrogate model (described in Section 2.2.2), if the surrogate model can 100% mimic the black box model, we have 100% faithfulness. However, we usually have to manage a fidelity-interpretability trade-off. A simpler surrogate model will result in higher interpretability, often at the cost of lower fidelity. When the architectural complexity of the surrogate model increases, we perceive it as more complex.

The term *completeness* (or coverage, used in the same way as faithfulness for global explanations) is an extension of fidelity, describing where we can expect high fidelity. That is, quantifying if we can expect high fidelity throughout the whole input distribution.

An example of a functionally-grounded approach is to determine if the input that an explainability method perceives to be important is, in fact, important to remove the information from the input. This can be approached in two ways [49], finding the smallest destroying subset by iteratively removing (defaulting to a baseline) the information that the explainability method considers most important, or conversely, finding the smallest sufficient subset. We explored this approach for game-playing agents in Paper B, where we analysed how well each explainability method identified the pieces needed to retain an advantage.

An example of a costly human-grounded approach was explored for game-playing agents where the accuracy and solving time was monitored for players solving chess puzzles with and without the help of the explainability method [50].

2.3 Explainability in Games

This section briefly reviews the related research needed to understand the contributions, describing key findings and showing examples of results. A discussion of other related work is given in Chapter 4, which will provide more context to the contributions of the thesis.

Research into explanations in game-playing is increasing; researchers are developing and adapting current explainability methods to our domain, seeking a better understanding of their algorithms. Furthermore, we can identify an alternative avenue of research related to game-playing, which we explore in this thesis: using game-playing to better understand the explainability methods.

Puri et al. [50] developed a saliency map method to improve on existing (non game-playing specific) methods. Their goal is to highlight what parts of the input (e.g., chess pieces) mostly influence a specific (chosen) move. They address the problem that existing methods often highlight irrelevant input regions to the proposed action. To address that, they develop a method that balances specificity and relevancy. Figure 2.9 shows an example of such a saliency map. This method identifies the importance of the presence of attributes, thus not being able to accurately identify the importance of squares being empty, which is often crucial in games such as chess where an open file is meaningful. They assessed the explainability method using an application-grounded method; they showed chess players (Elo 1600-2000) chess puzzles and measured the

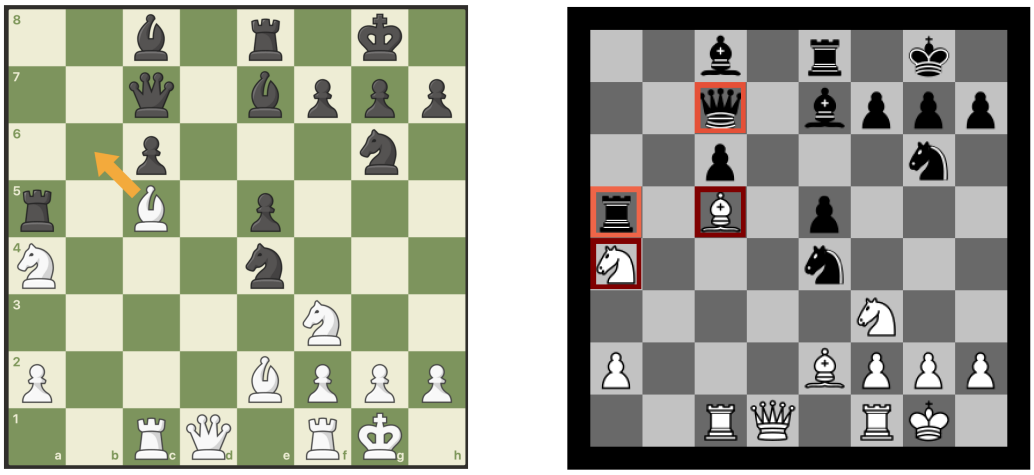
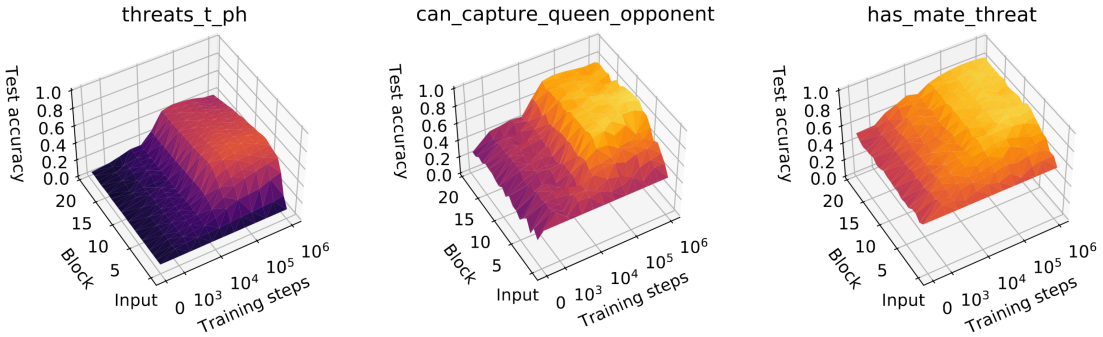


Figure 2.9: Example of a saliency map from Puri et al. [51].



(d) Stockfish 8's evaluation of threats.

(e) Can the playing side capture their opponent's queen?

(f) Could the opposing side checkmate the playing side in one move?

Figure 2.10: What-when-where plots from Figure 2 in McGrath et al. [52].

impact of the explanation in average time and accuracy when solving the puzzle. Highlighting relevant pieces as done using their method, helped chess players find the correct move and reduced the time needed to do so, compared to other explainability methods.

Two recent papers [52] [53] in the game-playing domain use concept-probing; both of them use linear probes, which is one of the topics discussed in this thesis. We investigate how the results from such probes can be interpreted and if we should consider other probing methods.

McGrath et al. [52] researched numerous ways to identify the knowledge AlphaZero acquired. They focus on three avenues: i) probing for concepts, ii) studying behavioral changes, and iii) investigating activations directly. The research related to pre-defined concepts, including concept-probing, is most relevant to this research. Still, their study of behavioral changes is also of some relevance; it is an example of an interesting application-grounded evaluation with the assistance of chess Grandmaster Vladimir Kramnik. They introduce what-when-where plots, visualizing concept probing results. *What*, indicating what concept, *when*, indicating when in the training process, and *where*, indicating where in the network the concept is being measured. The results (especially the *when* dimension) from this kind of map are particularly interesting for agents trained using self-playing because the agents need to discover the concepts. An example of such results can be seen in Figure 2.10, there we see a general increase until saturation in test accuracy over training steps and increased depth. Agents trained

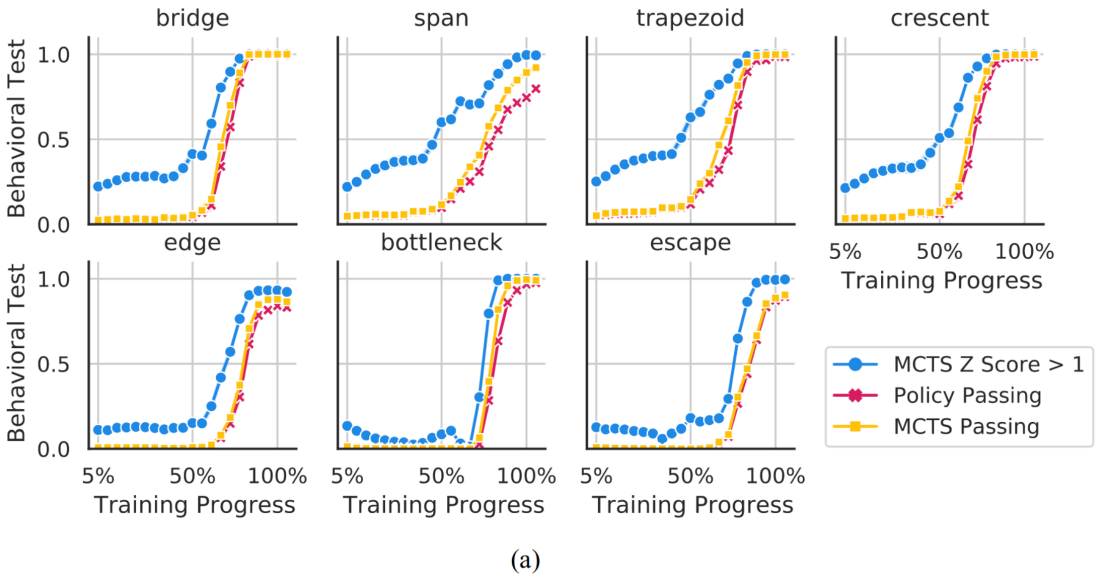


Figure 2.11: Results from a behavioral test from Figure 5a in Lovering et al. [53].

supervised, such as Stockfish, show much less nuance on the *when* dimension, because they are not introduced to new data during the training process. Furthermore, they also examine the value estimation of AlphaZero using a global surrogate model. They show how the relevancy of high-level concepts changes during the training phase.

Meanwhile, Lovering et al. [53] used concept probing and behavioral tests to analyze Hex-playing AlphaZero agent. The behavioral tests are designed so that the agent needs to understand a concept in order to play the correct move. Thus, by playing the correct move, we assume that the agent understands the concept and thus passes the test. An example of such results can be seen in Figure 2.11 where we see the results of behavioural tests where the network's ability to correctly use the concepts increases over time. The probing analysis's foundation is a linear concept probe, where they compare the concept probing results from different checkpoints and layers in the model. They showed that long-term concepts are best represented in the middle layers of the model, while short-term concepts are best represented in the final layers of the model. Comparing the concept probing and behavioral tests, they make an interesting discovery: in the checkpoints where they observed an increase in concept probing accuracy, they had already observed an increase in behavioral test accuracy using MCTS. This means that MCTS discovers concepts before the network learns to encode them.

2.4 Conclusions

In this chapter, we covered the background information related to the thesis's contributions. We also described the games and agents used, the necessary terminology, and explainability methods. Furthermore, we discussed some problems related to the current approaches and reviewed the research most relevant to this thesis.

In the next chapter, we will summarize the contributions of the thesis in three sections, outlining the contributions of each of the four included papers.

Chapter 3

Summary of Contributions

This chapter provides a brief context and summary of the contributions made in each of the resulting papers from this research.

The chapter is split into three sections. In the first section, **Understanding and Evaluating Explanations**, we group Papers A and C, and RQ1 and RQ3. It studies explainability methods, where the overarching theme is to develop evaluation methods that can expose the shortcomings of explainability methods and further understand how to interpret their results safely in the game-playing domain. The second section, **Unveiling What Game-Playing Agents Have Learnt**, covers Paper B and RQ2. It focuses on unveiling what the agents have learned. We apply state-of-the-art explainability methods and quantify to what extent the agents use human-understandable knowledge. In the final section, **Explaining Search**, we cover Paper D and RQ4. We focus on the search process, making the agent's search process and decision more human-understandable.

3.1 Understanding and Evaluating Explanations

Relevant papers:

A) **A. Pálsson and Y. Björnsson**, "Evaluating interpretability methods for DNNs in game-playing agents," in Advances in Computer Games - 17th International Conference, ACG 2021, Virtual Event, November 23-25, 2021, Revised Selected Papers, C. Browne, A. Kishimoto, and J. Schaeffer, Eds., ser. Lecture Notes in Computer Science, vol. 13262, Springer, 2021, pp. 71–81.

C) **A. Pálsson and Y. Björnsson**, "Empirical Evaluation of Concept Probing for Game-Playing Agents" (accepted for publication at the 27th European Conference on Artificial Intelligence (ECAI-2024))

3.1.1 Motivation and Goal

In this section, we address the following research questions:

- *RQ1: How effective/reliable are commonly used saliency map explainability methods in the game-playing domain?*
- *RQ3: What are the pitfalls and best practices in interpreting concept probing results in the game-playing domain?*

The advantage of deep learning algorithms is that they learn their own features or hidden representations, enabling them to draw very complex, non-linear decision boundaries without relying on hand-engineered features. However, even though we know the exact mathematical formulas behind the algorithms' decisions, their complexity makes it immensely challenging to decipher their rationale and identify what knowledge they have acquired.

A prevalent way of interpreting these models is using saliency maps described in Section 2.2.4 and concept probing described in Section 2.2.5.1. However, research has shown that these methods are prone to confirmation bias [6], misinterpretation, and even counterintuitive results [54]. The specific problems regarding saliency maps and concept probing are covered in Sections 2.2.4.6 and 2.2.5.3, respectively.

The objective of this research is to gain a better understanding of how to interpret explainability methods' outputs safely in the game-playing domain. We achieve that by leveraging unique opportunities that present themselves in game-playing. One of the unique characteristics of game-playing is the relationship between the value estimation and the principal variation (PV). The value estimation predicts the expected reward (or probability of winning) following the current policy, and thus, by explaining the value, we are, to some extent, also explaining what is about to happen (outcome of the game). Meanwhile, the principal variation shows us what is most likely to happen (given the current policy). Thus, comparing the explanation to the PV is helpful; it can help us create expectations about the resulting explanation to understand better how we can minimize misinterpretation of the results.

3.1.2 Contribution

The contributions relate to increasing our understanding of how we can reliably use explainability methods in the game-playing domain. The contributions are in the form

of novel evaluation methods and results from empirical evaluations. We can summarize the contributions as follows:

- We present three novel functionally grounded evaluation methods that evaluate the association between importance derived from saliency methods and the Principal Variation using self-play (evaluated from variations of the current state, perturbed and not).
- Empirically, we evaluate the applicability and effectiveness of several saliency-map-based methods for interpreting deep neural networks, where we expose the shortcomings of model-specific algorithms compared to model-agnostic ones.
- A novel surrogate method to "explain the explanation": a method that translates the importance values derived by saliency maps into human-understandable concepts. This method makes the game's objective (Breakthrough) apparent. The explainability methods put the most importance on the row of the pawn, but it is also helpful to know if it is the furthest pawn and if it has support from other pieces.
- We present a concept probing evaluation method where we derive the importance of concepts and compare them to concept probing results, enabling a quantitative approach for evaluating different concept probing architectures and approaches.
- We show that the widespread practice of using linear probes and interpreting their accuracy to indicate concept importance gives some, albeit limited, insights into concepts learned by the network. The probes' accuracy and feature importance is only moderately correlated. Thus, interpreting concept importance based on probe accuracy alone is unreliable. A better correlation is achievable using a more complex probe (neural network) and observing the difference in probing accuracy in the original and an impaired model.
- We present a novel evaluation method for concept-probing methods by monitoring how the autoencoder passes through and reconstructs the signal. This yields a sound expectation about the ground truth of the trend and magnitude of the signal we expect the probing algorithms to capture. These experiments showcase the limitations of the linear probing algorithm, which fails to capture information known to be in the compressed state.

3.1.3 Methods

A prerequisite for this research is to acquire models to evaluate using the proposed methods. The methods fall into two main categories, the first category is the evaluation of saliency maps, where we use an AlphaZero style algorithm which we develop and train to play Breakthrough (the game is described in Section 2.1.1.1 and its development is described in Appendix E.1). The other category is the evaluation of concept probing methods, where we use the chess agent Stockfish with official and custom weights (trained on modified datasets) and a custom autoencoder trained to encode and decode chess positions.

Saliency Maps

We used Breakthrough and the AlphaZero style algorithm to assess the saliency methods. The AlphaZero style model returns a value estimation and a policy output.

However, this research focused solely on explaining the value estimation, i.e., why the model prefers one side to the other. Game-playing is well suited for ablation studies, e.g., compared to image recognition; when we modify game-playing positions, they change but (most often) are still valid positions; conversely, if we remove the pixels containing a cat's head, it does not become something else. It is merely a cat with a missing head.

We proposed four methods to evaluate the saliency maps: two in which we associate the importance of saliency map algorithms to the outcome of self-play games before and after ablation, a third one, which associates the importance with the future role of a piece, and a fourth and final one, a surrogate model explaining the explanation.

First, we assess if the saliency method assigns the highest saliency to a critical piece and, conversely, if the lowest saliency is assigned to a non-critical piece. We evaluate the win-rate without deletion for each method and then separately remove the least- and most important piece and measure its impact on the game's outcome. We analyze the results in a total of nine bins grouped by the evaluation from the AlphaZero network before deletion; then, we can evaluate the impact of the removal depending on which player has an advantage and by how much.

The second method finds the smallest subset of pieces required to retain a winning position. First, we gather a dataset of positions where the current player considers himself to have a slight advantage (estimated by the model). Then, we iteratively remove the pieces ranked least important according to an explainability method until all of the player's pieces have been removed. After each piece removal, we evaluate the position using self-play to see how each removal affects the win-rate. It is thus desirable that the method can identify pieces that affect the win rate the least and assign the lowest importance to them. Therefore, we consider the explainability method with the highest area under the curve to have the highest quality.

The first two tasks introduced are a form of ablation, and while providing helpful insight, it is rather obstructive. Thus, in the third method, we introduce an ablation-free method that associates the importance derived from an explainability method with the future role of a piece. In this experiment, we focus only on the piece that ultimately secures the win (reaches the top row) in a self-play game. It is arguably a critical piece; thus, we would like the explainability method to align with our expectations and consider it important. We aggregate the results over multiple samples to find the average and variance of the importance value (using the explainability method) assigned to the piece.

Finally, we present a method that "explains the explanation." It is a surrogate model that predicts each piece's importance (i.e., the saliency value from the explainability algorithm), given high-level information about the piece and the current position. Accordingly, the input into the surrogate model are the features described in Table 3.1 for each piece, and the target is the importance, or saliency value, for the corresponding piece. We use LightGBM as a surrogate model and present the results as shapley values for each input feature.

Concept Probing

To analyze concept probing results, we proposed two methods: first, one where we remove knowledge about concepts from the agent to analyze its impact on performance and concept probing results, second, one where we monitor the information flow through an auto-encoder.

First, for each concept (described in Paper C), we train an agent with and without the given concept present in the training set. Then, we can quantify the concept's

Table 3.1: Description of concepts used by The Explaining the Explanation method.

Concept Name	Description
Is Furthest	True if the piece is among the furthest pieces
Player COM	Player's pieces center of mass
Opponent COM	Opponent's pieces center of mass
Piece Imbalance	Difference in number of pieces
# Potential Captures	Number of available captures of a piece
Has Support	True if a piece has support from another piece
Giving Support	True if a piece is giving support to another piece

importance by making the modified agent (trained without the given concept) play against its unmodified version. For a detailed explanation of how we hide the concepts and evaluate their importance, see Appendix E.2.

Having determined the importance of each concept, we can investigate the results of concept probing methods; i.e., what concept probing approach most closely mirrors the importance of the concepts? We used this ground truth to compare different concept probing architectures and approaches, such as probing for removed concepts and untrained lower bounds.

Second, we designed an autoencoder experiment to contrast the different probing architectures. Within this controlled environment, we can investigate whether the different probes are a good proxy for the information stored within the representations. When we encode and decode information, we can observe the reconstructed state to know how much information passes through the network - for us to decode information successfully, it must also be available in the encoded state. This method monitors the probing accuracy during training in the most compressed and reconstructed states. It allows us to observe inconsistencies in probing accuracy values and trends.

Inside the autoencoder, we expect considerable compression of information, which is relevant when probing a small neural network, such as Stockfish's NNUE, where the concept representations are likely more compressed than in a larger neural network.

The autoencoder consists of two parts, an encoder and a decoder, which are trained to compress and reconstruct the input space. In this case, the encoder and decoder mirror each other in terms of the number of layers and neurons in each layer. The design uses a sequence of fully connected linear layers with ReLU activations.

3.1.4 Results

This subsection summarizes the main results from papers A and C, where we introduced novel methods to evaluate explainability methods. In Paper A, we evaluate saliency maps; in Paper C, we focus on concept probing. First, we will cover the methods to evaluate saliency maps in Paper A:

Table 3.2 presents the results from removing the least and most important pieces. We can see here that by removing the most important piece, we almost eliminate all the advantages of the current player. Conversely, removing the least important piece is non-destructive for the position. It can even be helpful, as we can see for the Shapley value sampling method. A good tactic in this game is to force the opponent into a zugzwang position. So, we might be helping the player avoid such a position by removing a nonimportant piece.

Figure 3.1 shows results from the sufficient subset experiment. We would like to see this curve drop as late as possible. Here, Shapley Value Sampling and LIME outperform the rest again. Furthermore, the gradient method performs the worst.

In Figure 3.2, we see the results from the experiment where we evaluate the piece's importance that ultimately secures the win. Here, we see that the model-specific methods (such as the gradient methods) do poorly identifying that, while LIME and Shapley Value Sampling outperform all the others. Interestingly, the occlusion method performs the worst.

In Figure 3.3, we see how explaining the saliency maps exposes how we can extract the game's tactics from the explanations. For example, we see that the row of the piece is the most determining factor of the piece's importance and if it is among the furthest of the player's pieces. However, it is also clear that other pieces are needed to support it to be a credible attack because the center of mass of the pieces is also a determining factor.

Finally, we will briefly cover the results from the concept probing methods in paper C:

Figure 3.4 shows a limited correlation between probing accuracy and importance. Further results can be seen in Table 3.3, from which we can draw two conclusions. First, when analyzing the probing results of the Regular model, $P_j(f_{reg})$, the neural network probe has the highest correlation with importance. Second, the change in probing accuracy after removing a concept from the training set, $P_j(f_{reg}) - P_j(f_{res,c_i})$, offers an even higher correlation with importance where the neural network probe also performs best.

In Figure 3.5, we show the problems linear probes face when the information is too compressed. Even though the autoencoder is increasingly learning to represent the concept and pass it onto the reconstructed layer, the neural network probe is the only probe that correctly represents that trend.

3.1.5 Summary

We presented multiple ways where game-playing can assist in evaluating explainability methods. We showed examples where we expose that some algorithms do, in fact, not align with our expectations, especially the model-specific methods. Furthermore, we presented a novel approach to evaluate concept probing results, comparing the probing results to their importance, which is problematic in most other domains where the evaluation of concept importance is based on a potentially biased dataset, reliant on extensive data collection, or expensive real-world experiments. Finally, we were able to expose shortcomings of linear probes using auto-encoders.

Table 3.2: Aggregated results from 10.000 self play games. Based on their evaluation, the positions are placed into nine bins or quantiles, i.e., positions that have similar win probability evaluated with our model are grouped in bins. We group positions with close to zero win probability in the left-most column. In the right-most column, we have a group with almost 100% win probability. The top-most row shows the proportion of games won by the player to move for each bin, and the subsequent rows show the same after removing the highest or lowest-ranked piece according to each method. The bin's average model evaluation (range between -1 and +1) is in the bottom row.

Method	**Importance**	**Proportion of games won**								
Nothing deleted	-	**0.09**	**0.26**	**0.36**	**0.44**	**0.51**	**0.55**	**0.64**	**0.74**	**0.90**
Occlusion	Highest	0.08	0.21	0.26	**0.29**	**0.31**	**0.37**	**0.40**	**0.43**	**0.50**
LIME		**0.04**	0.18	0.26	**0.29**	0.34	0.39	0.45	0.47	0.51
SVS		**0.04**	0.19	**0.24**	0.32	0.38	**0.37**	0.44	0.45	0.51
Gradient		0.12	0.23	0.35	0.41	0.49	0.50	0.57	0.63	0.66
Integ. Gradients		0.06	0.18	**0.24**	0.32	0.38	0.41	0.44	0.47	0.56
Gradient Shap		0.05	**0.15**	0.27	0.31	0.36	0.41	0.45	0.49	0.57
Deeplift		**0.04**	0.18	**0.24**	0.34	0.35	0.40	0.46	0.47	0.51
Occlusion	Lowest	0.08	0.26	0.31	0.38	0.46	0.50	0.60	0.73	**0.91**
LIME		0.11	0.28	0.42	0.47	0.55	0.61	0.64	0.77	**0.91**
SVS		0.09	**0.30**	**0.44**	**0.52**	**0.58**	**0.62**	**0.67**	**0.78**	0.90
Gradient		0.10	0.27	0.40	0.48	0.52	0.58	0.63	0.73	**0.91**
Integrated Gradient		0.11	0.22	0.32	0.37	0.43	0.51	0.57	0.73	**0.91**
Gradient Shap		**0.12**	**0.30**	0.40	0.49	0.54	0.59	0.65	0.76	**0.91**
Deeplift		0.10	0.27	0.31	0.38	0.45	0.50	0.58	0.72	**0.91**
Mean model evaluation (pre deletion)		**-0.90**	**-0.68**	**-0.45**	**-0.22**	**0.00**	**0.22**	**0.45**	**0.67**	**0.90**

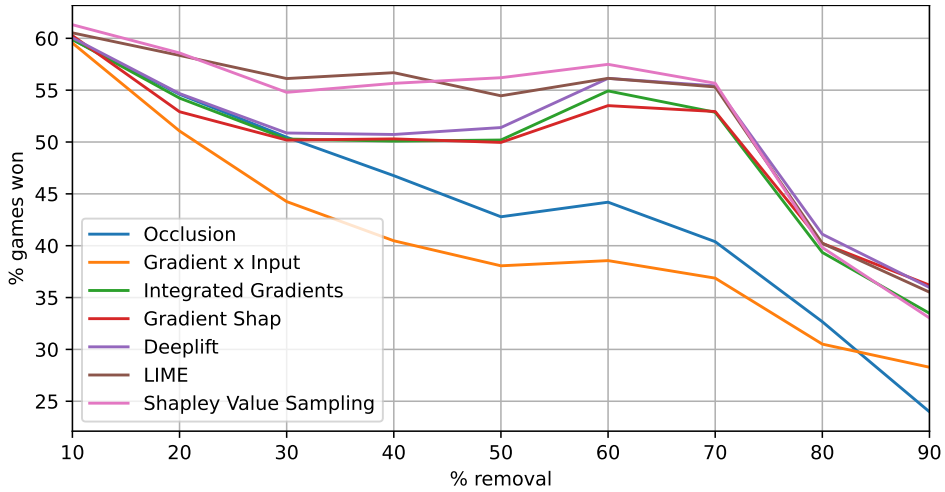


Figure 3.1: Effect of gradually removing the least-important pieces. We begin with a dataset of positions where the current player has a slight advantage. Each data point represents the proportion of removed pieces (ranked least important) and the resulting win-rate after removal.

Table 3.3: Pearson correlation between concept probing results and importance.

Probe	$P_j(f_{reg})$	$P_j(f_{reg}) - P_j(f_{res,c_i})$	$P_j(f_{reg}) - P_j(f_{unt})$
Ridge	0.69	0.83	0.64
LGBM	0.68	0.87	0.67
NN	**0.73**	**0.92**	**0.72**

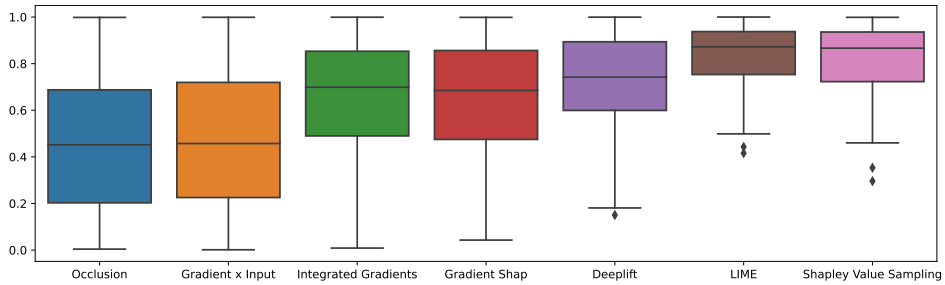


Figure 3.2: The visualization displays the distribution of saliency values assigned to the piece that secures the winning move 4-ply before the win.

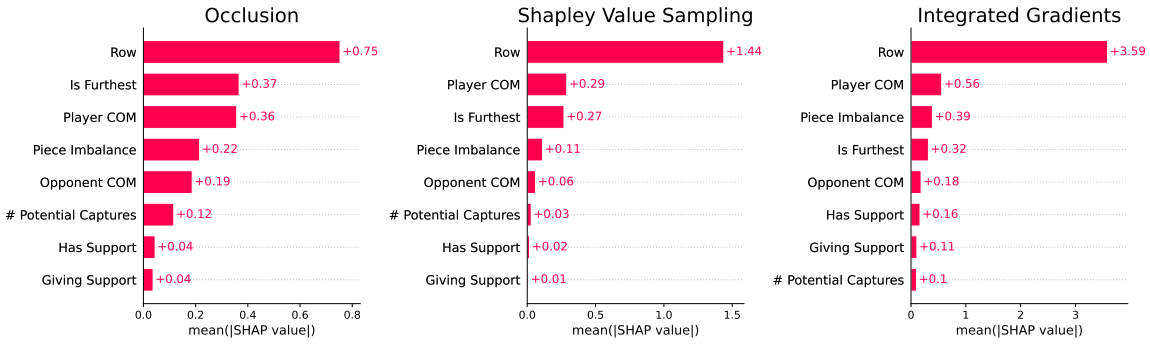


Figure 3.3: The average Shapley Values of each of the inputs to the interpretable surrogate model. Here COM stands for center of mass. Has- and Giving Support indicate if the piece is supporting or giving support diagonally to a piece of the same color. # Potential Captures indicates is the number of available capture moves for the current player.

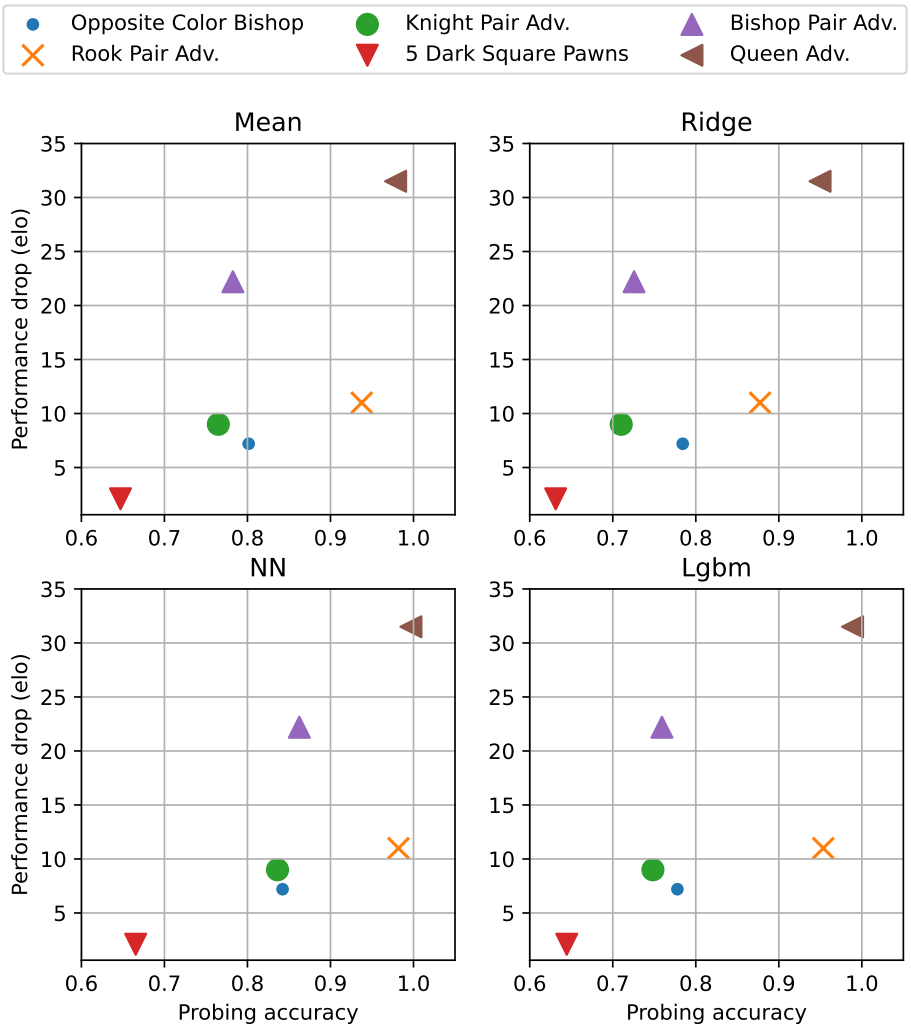


Figure 3.4: The relationship between probing accuracy and true importance measured in Elo points for all probes, as well as the mean value of all three probes.

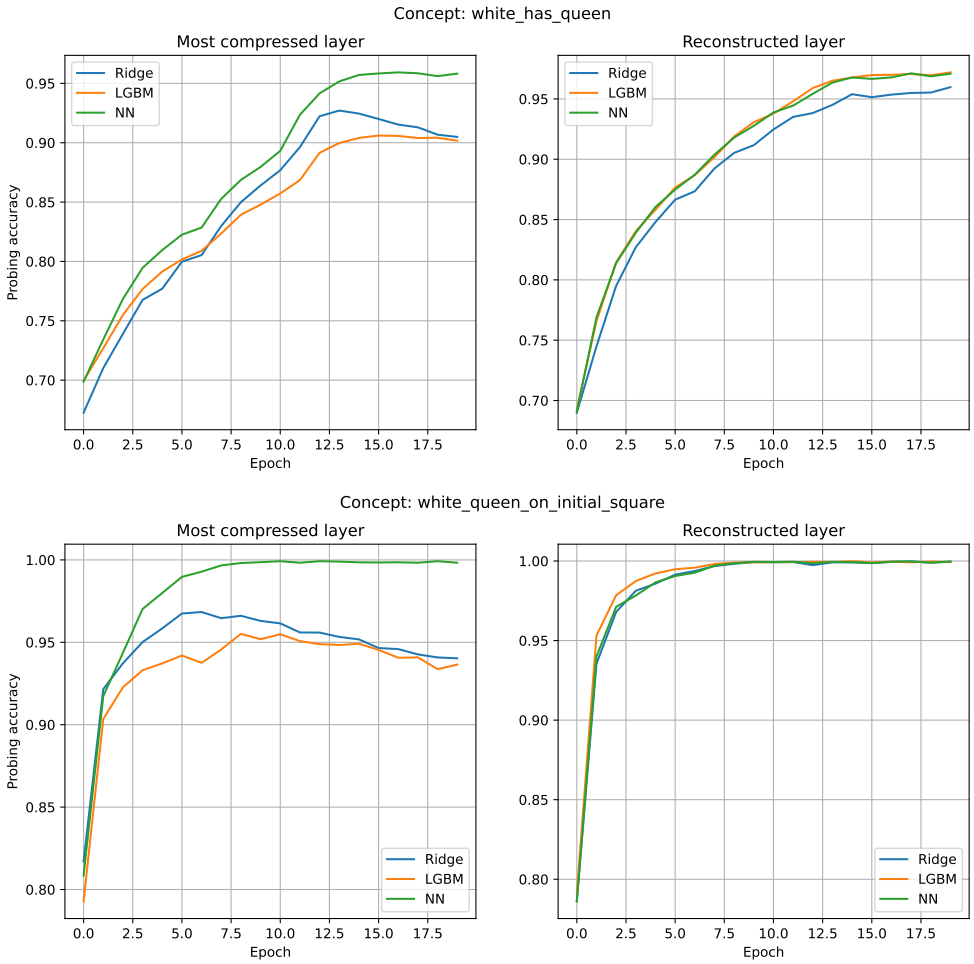


Figure 3.5: Probing accuracy results performed during training of the autoencoder. Results are for the concept white_has_queen (top) and white_queen_on_initial_square (bottom).

3.2 Unveiling What Game-Playing Agents Have Learnt

Relevant paper:

B) A. Pálsson and Y. Björnsson, "Unveiling concepts learned by a world-class chess-playing agent," in Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China, ijcai.org, 2023, pp. 4864–4872.

3.2.1 Motivation and Goal

In this section, we explore how we can use state-of-the-art explainability methods to reveal the knowledge acquired by deep learning agents, using Stockfish, the chess-playing agent, as our test bed. We addressed the following research question:

- *RQ2: How to use explainability methods to unveil the concepts learned by game-playing agents?*

In recent years, game-playing agents have moved towards using neural networks, which has eased the encapsulation of the relevant domain-specific knowledge and resulted in much-improved playing strength. This trend has come at the cost of making the resulting models ill-interpretable and challenging to understand or use for enhancing human knowledge.

In Paper B, we explore ways to reveal the knowledge acquired by these super-human agents so we can better understand and learn from them. The latest version of Stockfish, at the time of this research, is of particular interest related to this development; at that time, it was arguably the strongest chess-playing agent in existence, and its design has some interesting properties for our research. It is a hybrid of i) a classical model [1], which contains hand-crafted (but automatically tuned) evaluations that contain many concepts from human chess literature, and ii) a neural network model, which is called NNUE, or Efficiently Updatable Neural Network, which has learned its own (hidden) concepts to evaluate positions. Comparing these two served as the foundation of this part of the thesis.

The goals of this part of the research are twofold: first, to identify human-understandable knowledge embedded in Stockfish's neural network and see how they differ from the classical hand-crafted evaluation function. Second, to shed light on some actionable insight that could drive further development of the neural network or the classical model.

3.2.2 Contribution

The main contributions of this part of the research are showcasing ways the game-playing community can benefit from explainability methods. We demonstrated how to adapt and use state-of-the-art explainability methods to highlight contrast and understand some of the strengths and weaknesses of the neural network and the hand-crafted model. We can summarize the contributions in the following points:

[1] Stockfish 16.1 [55] which was released after this research has removed the hand-crafted version and relies now on two neural networks instead.

Table 3.4: Description of Δ piece value concepts.

Concept Name	Description
ΔP	Difference in number of pawns
ΔN	Difference in number of knights
ΔB	Difference in number of bishops
ΔR	Difference in number of rooks
ΔQ	Difference in number of queens

- We demonstrate how state-of-the-art interpretability methods can gain insights into concepts learned by a world-class game-playing agent.
- We use the gained insights to pinpoint the relative strengths and weaknesses of the neural network and the hand-crafted model.
- We demonstrate that the neural network contains a substantial amount of human-explainable knowledge. With a linear combination of the classical concepts we are able to explain approximately 50% of the variation of the neural network.
- We show the neural network's capability to detect specific threats statically that would require a search using the classical method.
- We discovered significant disparities in how the models implement King safety. Apart from highlighting the differences between the two approaches, this finding might serve as a basis for projects such as adjusting the classical King safety implementation to mimic the neural network better.

3.2.3 Methods

This research primarily utilized three global explainability techniques to obtain quantitative results. Furthermore, we qualitatively evaluated individual positions by filtering based on model disagreement.

First, we apply a Global Linear Surrogate Model to evaluate concepts' relative value or importance. For example, the first experiment evaluates the relative importance of different piece types. In order to do this, we construct a dataset, D, containing states s. For every s, we evaluate the model to get a model evaluation of the state, i.e., to evaluate who is better and quantify by how much. For each of these states, we evaluate delta piece count concepts. E.g., ΔB indicates the difference in the number of bishops present on the board. $\Delta B = 0$ indicates that both players have the same number of bishops, while $\Delta B = -1$ indicates that black has one more bishop than white, and so on. Furthermore, when all the concepts in Table 3.4 have been evaluated, they can be used by the surrogate model to predict the model evaluation. The surrogate model is a linear model that we dissect after training:

$$w_p * \Delta P + w_n * \Delta N + w_b * \Delta B + w_r * \Delta R + w_q * \Delta Q = y \tag{3.1}$$

In this case, the linear weights for the different concepts will serve as a proxy for importance, and we can, for example, compare the weights w_p and w_q. Thus, we interpret $\frac{w_q}{w_p}$ as the queen value evaluated in a number of pawns. We can use a similar

Table 3.5: Shapley values of concepts found in Table 2.2 estimated using Shapley value sampling with a linear model. Ratio is calculated as the value of NNUE divided by the value of Classical.

	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

approach to evaluate the relative importance of the concepts used by the classical model, which can be found in Table 2.2.

Furthermore, to extend the analysis of surrogate models, we apply Shapley Value Sampling (SVS), covered in Section 2.2.4.5, to extract more information about the concepts. We apply SVS to evaluate the Shapley Value for each of the concepts. The values in Table 3.5 reflect such an experiment.

The final explainability method used is concept probing, examined in Section 3.2 and described in Section 2.2.5.1. The aim of concept probing in this part of the research is mainly twofold: i) to get a grasp of how the agreement is between the classical concepts and the neural network, both overall and at different phases of the game, and ii) to evaluate the impact of the rich input format, it is very sparse and can be helpful to identify some concept-related relationships, especially with respect the king's position.

We conducted two experiments with a linear probing model to achieve the goals. First, we probe the model at the input and after the first linear layer. By comparing the accuracy of the probe on the input and inside the model, we get a better grasp of the relative value of the input format and the learned neural network weights. The second experiment probes the model using one phase at a time (from a total of 8 phases)—this way, we see if, in certain phases, we can observe more agreement between the neural network and the implementation of the concept by the classical method.

Finally, we perform a qualitative analysis by filtering games where there is an apparent disagreement between the methods about who has the advantage. For example, filtering games where there is a disagreement and the classical method attributes the evaluation primarily to king safety will likely expose some king safety-specific disagreement between the models.

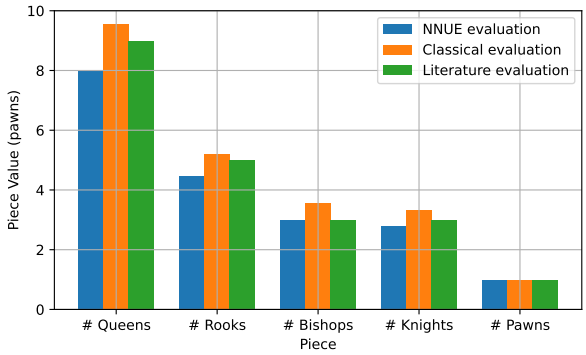


Figure 3.6: Estimated piece values using the weights of a linear surrogate model. Each #_* feature corresponds to Δ∗ feature in Table 3.4. The values of the bars represent the weights corresponding to each piece, normalized by the weight of the pawn.

3.2.4 Results

In this section, we describe the main results briefly; we refer to the paper for more details. In the paper, we explored simple concepts, such as piece values, and more complex ones, such as chess tactics, and the concepts implemented in the classical model.

First, we show how the model evaluates the different piece types. In Figure 3.6, we see that the neural network puts less weight on the pieces, from which we can infer that it values other things more, such as the position and dynamics of the pieces. Further analysis of the different phases is included in the paper, which shows how piece values change over the duration (phases) of the game.

Next, we analyze the hand-crafter concepts from the classical model; in this experiment, as seen in Table 3.5, we evaluate the Shapley values of the different concepts and compare the differences between the two models. There are two main takeaways from that experiment. First, dynamic concepts like winnable and passed pawns are relatively much more important to the neural network than the hand-crafted model. Secondly, we see that king safety is much less important for the neural network, indicating a disagreement between the models. The same holds for the space concept, although, it is considerably less important for both models.

In Figure 3.7, we see the probing accuracy of the classical concepts in the neural network. The lowest probing accuracies are for threats and king safety, and the highest is for material. We also probe the model's input to identify how much the model's rich input format contributes to making the concepts accessible to the model. For some concepts, we see a substantial increase, such as for the concept winnable, and although we see a significant increase for all concepts, the concept space has a very small increase.

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 3.8 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.

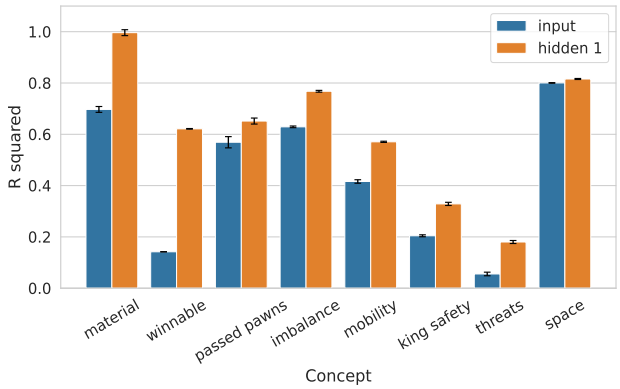


Figure 3.7: Concept probing results on i) the input features and ii) after the first hidden layer, using the official NNUE weights.

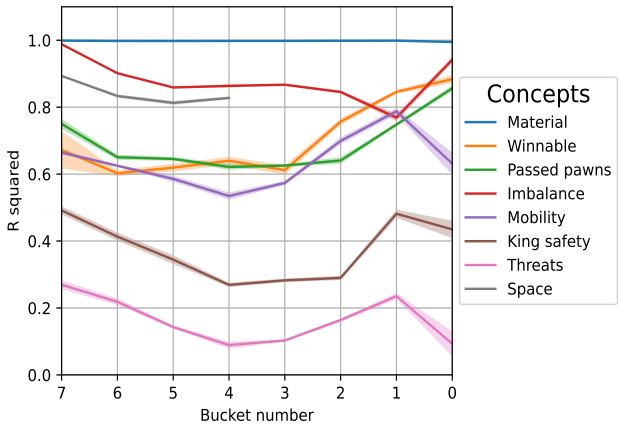


Figure 3.8: This figure shows concept probing results on subsets containing only a single bucket using ridge regression, evaluated at the first hidden layer of the model, using the official NNUE weights. The bucket number indicates which layer stack will be used; the number of pieces on the board determines the bucket number, which is calculated by $(piece\_count - 1)/4$.

Finally, we do a qualitative analysis of the difference between the methods. We filter games where the methods disagree about who has the advantage—observing the positions; we see that the NNUE has a much better ability to statically detect complex threats that would require the classical method to explore future states to detect.

3.2.5 Summary

In this section, we summarized the contributions made in Paper B. The work we explored aimed to expose what the neural network has learned using pre-defined concepts. We explored the relevant state-of-the-art explainability algorithms and exposed the contrast between the neural network and the classical model. Both model-agnostic and model-specific explainability techniques seem to, for the most part, reasonably interpret and explain the knowledge acquired by the game agent's deep neural network, each with its pros and cons. This work opened up some interesting questions about the hand-crafted implementation of king-safety; for example, can we improve the classical model by changing the king-safety concept to be more in line with the neural network?

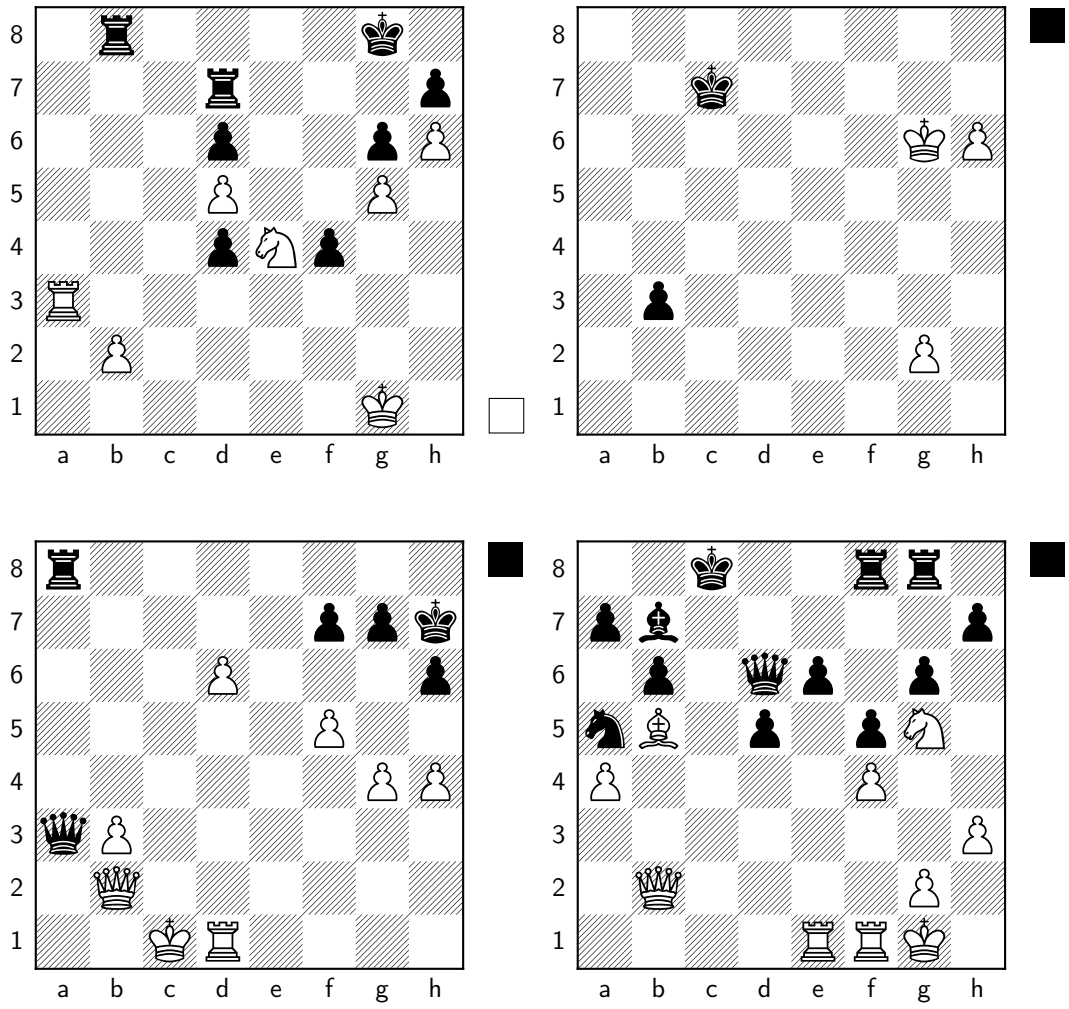


Figure 3.9: (Upper Left) The NNUE statically detects the *fork* Nf6 favouring White unlike the classical model; (Upper Right) Black (to move) wins because promoting with a check, seen (statically) by the NNUE but not the classical model. (Bottom) The classical model judges king safety the most critical feature favoring the attacking player (Black and White, respectively); the NNUE is unimpressed with the attacks and correctly (slightly) prefers the defending player.

3.3 Explaining Search

Relevant paper:

D) Y. Björnsson, S. Helgason, and A. Pálsson, "Searching for explainable solutions in sudoku," in 2021 IEEE Conference on Games (CoG), Copenhagen, Denmark, August 17-20, 2021, IEEE, 2021, pp. 1–8.

3.3.1 Motivation and Goal

In this section we address the following research question:

- *RQ4: How to bias the search process of game-playing agents to make more human-understandable decisions?*

The research focus of XAI has so far mainly been on model interpretability, in particular, providing insights into concepts learned by (deep) neural networks as covered in the previous Sections (3.1 and 3.2). However, for many tasks, heuristic search is also an integral part of the decision-making process of intelligent systems. Heuristic search is one of the fundamental problem-solving techniques in AI and computer science. It provides computer-based agents the means of reasoning, or "thinking ahead."

A computer-based system that incorporates search or thinking ahead does not make decisions based on a single state, but rather multiple states. The number and order of states that such a system explores can be so vast that it is beyond human comprehension. The solution to heuristic-search problems typically takes the form of a sequence of actions. How did each step come about? Why is that particular sequence preferable to an alternative one? Such questions are especially relevant in settings where a human is expected to verify or execute the resulting solution and in settings where a human hopes to learn from the solving process.

In this work, we focus on the game of Sudoku, mainly because of its simplicity, we do not have an adversary and there is a well established human-literature with well-defined strategies. Sudoku is formalized as a CSP and is very easily solved with computers. However, In Sudoku, we have this contrast between how we and computers solve the puzzle. We solve it with limited backtracking, while computers use it extensively.

We have two possible approaches: One is to treat the solver as a black-box and then try to explain the resulting solution post-mortem. The other is to alter the solver to produce more human-understandable solutions.

In this research, we opted for the latter method. The goal is to look for not only a solution but also the most explainable one. To achieve that, we must quantify explainability objectively based on different solution strategies and general principles to guide a solver.

3.3.2 Contribution

This paper explores how to alter a search-based reasoning process to generate solutions more explainable to humans, using the domain of Sudoku puzzles as our test bed. The decisions made by such an intelligent system must ideally be transparent, verifiable, and, where applicable, instructive for humans. The main contributions are as follows:

- We provide a hybrid of a heuristic- and constrained-based solver that biases the search towards finding solutions that are easily explainable to humans.

Table 3.6: Strategy specific cost, w_m.

	w_m
One-Dimensional Unique Candidate	1
Sole Candidate	3
Two-Dimensional Unique Candidate	5
Naked Double Candidate	10

- We model the perceived human mental effort of using different familiar Sudoku-solving techniques.
- We show that such an approach is feasible in our test domain and will apply to other domains with similar characteristics.
- We show how to find an explanation understandable to human players of varying expert levels and evaluate the algorithm empirically on a wide range of puzzles of different difficulties.

3.3.3 Methods

This research combines three methods: i) pre-defined solving strategies from the literature, ii) a utility function, and iii) a search algorithm. The strategies implemented are shown in Table 3.6, and further descriptions of the strategies can be found in the paper.

The aim is to bias the solver using a utility function that evaluates how explainable a solution is. The solver finds the solution that minimizes the utility function, resulting in the most explainable solution available. Furthermore, the utility function allows the depth-first solver to prune less explainable solutions and decrease the search time.

The utility function aims to map perceived human effort and preferences to create the most explainable and easily understandable solution - it is formalized as follows:

$$W_{total} = \sum_{i=1}^{N} \left(\left(1 + \left(1 - \frac{a_{m,i}}{U_i} \right) \right)^{\delta} \theta^{n_{m,i}} w_{m,i} + \xi_{m,i} w_{m,i} \right) \tag{3.2}$$

where m indicates the strategy used, w_m is the cost of the strategy used (see Table 3.6), N is the number of unassigned variables at the beginning of the puzzle, U_i is the number of unassigned variables at step i, $a_{m,i}$ is the number of available moves for the chosen strategy, δ is the power of the transformation function, θ is a cost decaying factor, $n_{m,i}$ indicates the count of similar moves done before our move m and finally $\xi_{m,i}$ indicates the cost of learning a strategy.

In the first term of our equation, $\left(1 + \left(1 - \frac{a_{m,i}}{U_i}\right)\right)^{\delta}$, we capture the context specific cost (see (ii)) for strategy m at step i, indicating a higher cost with lower availability. The strategy specific cost (see (i)) is lowered by repetitive use of the same strategy through the decaying factor $\theta^{n_{m,i}}$ (see (iii)). The value for the learning factor (see (iv)) $\xi_{m,i}$ was arbitrarily set to 10 if we are using strategy m for the first time at step i, otherwise it is 0.

The pseudo-code in Algorithm 1 from the paper describes the search of the solver, where each recursive call to the search function does the following: (1) for each available

Table 3.7: Least costly path found to solve the board in Figure 3.11. Cost 33.

Available Moves				Chosen Moves		
sole	1d unique	2d unique	naked	Strategy	Value	Square
3	2	5	1	1d unique	5	I1
4	3	6	1	1d unique	5	E3
4	3	7	1	1d unique	8	E1
4	3	6	1	1d unique	2	H1
4	2	6	0	1d unique	8	H3
3	1	5	0	1d unique	3	H7
3	2	4	0	1d unique	3	E8
3	2	3	0	1d unique	2	E7
2	2	2	0	1d unique	2	I8
1	1	1	0	1d unique	7	I7

strategy we apply (up to) *batch size* (we group moves into batches to decrease the search space) many moves (not transitively considering those that arise from using the strategy); (2) the cost of the (partial) solution is computed using our cost function; (3) we recurse into the search once more with the changes from applying that strategy; (4) we undo everything from applying the strategy and try the next.

The search is depth-first branch-and-bound-like, keeping track of the least costly solutions found so far and pruning where applicable. It also uses a transposition table, for avoiding reexploring the same state if reached via two different paths, and macro-actions to reduce the search space. A significant characteristic is that the actions allowed for the algorithm are always explainable in terms of Sudoku strategies. Given that the algorithm has at its disposal sufficiently advanced strategies to progress in each step, it is complete (i.e., always finds a solution).

3.3.4 Results

Here we cover the main results from this part of the research. For the experiments we gathered a dataset of 50 puzzles per difficulty level, for levels simple, easy and intermediate. Using our method we get a cost per solution and in Figure 3.10 we see how the explainability cost maps on the three difficulty levels.

To further explore the algorithm we visualize two solutions for an unsolved board with 10 empty squares shown in Figure 3.11. We show the most explainable solution as well as the least explainable solution, in Tables 3.7 and 3.8 respectively. Here we show the solvers ability to find the simplest and most explainable solution to present, as well as a solution, not preferable, which randomly changes strategies and picks strategies requiring more mental effort to understand.

3.3.5 Summary

In this work, we put the focus on the search part of the decision-making process. We use Sudoku as our test-bed and provide a solver — a hybrid of a heuristic- and constrained-based — that biases the search towards finding solutions that are easily explainable to humans with different levels of domain expertise. We provide a concrete

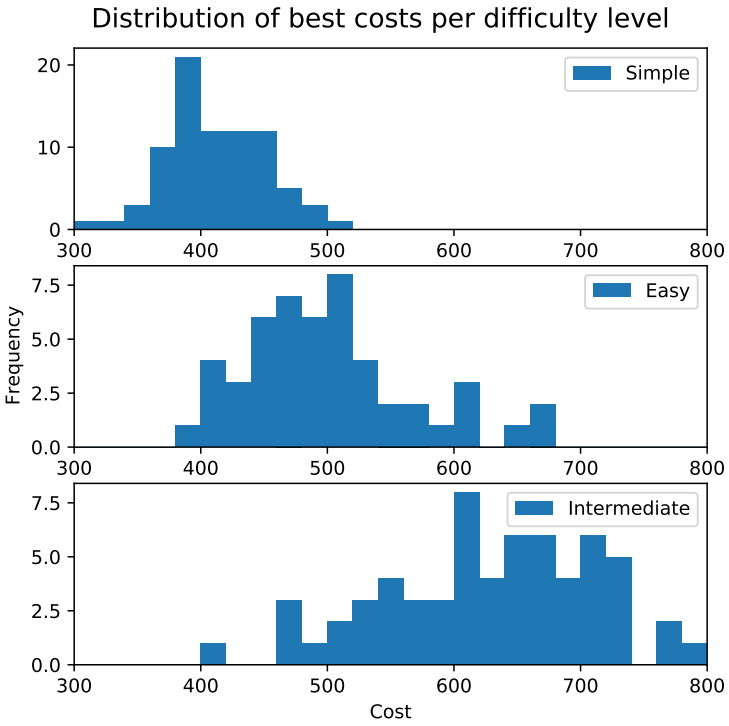


Figure 3.10: Resulting distributions of the best costs found from solving 50 tables of each difficulty levels: Simple(top), Easy(middle), and Intermediate(bottom).

3	5	7	4	2	1	9	8	6
4	2	1	8	9	6	5	7	3
6	8	9	7	3	5	4	1	2
1	9	3	2	8	7	6	4	5
	6		9	1	4			7
7	4	2	6	5	3	1	9	8
9	7	6	3	4	2	8	5	1
	1		5	7	9		6	4
	3	4	1	6	8			9

Figure 3.11: Unsolved Sudoku board with 10 remaining moves.

method for achieving this and show that such a solver is feasible in our test domain. We also provide an analysis of various aspects of the search- and solution space to better reveal the challenges faced by the solver.

Table 3.8: Most costly path found to solve the board in Figure 3.11. Cost 342.

Available Moves				Chosen Moves		
sole	1d unique	2d unique	naked	Strategy	Value	Square
3	2	5	1	naked	7	I7
2	2	4	0	sole	8	H3
3	3	5	1	naked	5	E3
3	3	4	1	naked	8	E1
2	2	3	0	2d unique	5	I1
2	2	4	0	sole	2	H1
2	2	4	0	sole	3	H7
2	2	3	0	1d unique	3	E8
2	2	2	0	2d unique	2	E7
1	1	1	0	2d unique	2	I8

Chapter 4

Related Work

In this chapter, we revisit and reflect on related work in Explainable AI and game-playing. Some work that influenced this research was already introduced in Section 2.3; however, this section will dive deeper into some work and reflect on past and concurrent related research in the game-playing domain. We emphasize the work in McGrath et al. [38], [52], which was both one of the first, and is still the most comprehensive work on concept-based explanations in the game-playing domain—following that we will highlight other related work on explainability and game-playing and reflect on how it relates to our research.

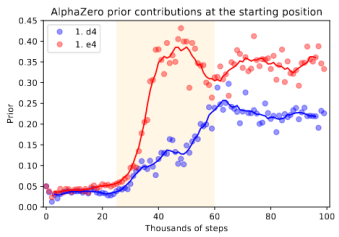
4.1 Acquisition of Chess Knowledge in AlphaZero

The paper McGrath et al. [38], [52] investigates the concepts that AlphaZero's neural network learned. It was one of the first papers to research concept probing in game-playing agents and was influential to our work. It is still one of the most comprehensive papers released on this topic. Concept probing in game-playing agents was also concurrently researched in other groups, such as the work published in Lovering et al. [53], discussed in Section 2.3.

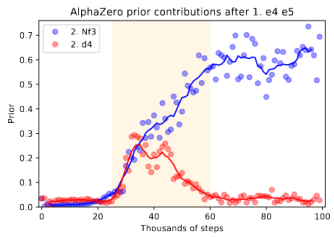
The insights gathered in McGrath et al. [38], [52] stem from two points of view; first, insights about to what degree the human-understandable concepts get represented in the final model, and second, exposing how they appear during training, which is particularly interesting because they train it using self-play - without any human knowledge. Therefore, identifying how human-understandable knowledge appears during training can shed some light on whether there is some natural course of knowledge discovery, perhaps similar to how chess theory has evolved during the past centuries. However, this second part relates less to our thesis since Stockfish, the chess agent we investigate, is not trained using self-play. Instead, it is trained using supervised learning with a predefined dataset; thus, the evolution of concepts is less relevant.

They cover multiple ways to inspect AlphaZero, such as concept probing, behavior analysis using a surrogate model, a study of the evolution of AlphaZero's openings, a qualitative analysis by chess grandmaster Vladimir Kramnik, and finally, unsupervised concept discovery using non-negative matrix factorization (NMF).

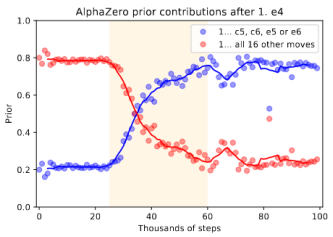
They show that human-understandable concepts are, to some degree, linearly represented by AlphaZero's activations, most of which increase during training until they saturate. They demonstrate these results using what-when-where plots. *What*, indicat-



(a) After 25k training iterations, e4 and d4 are discovered to be good opening moves, and **rapidly adopted** within a short period of around 30k training steps.



(b) Rapid discovery of options given 1. e4 e5. Within a short space of time, ♘f3 is settled on as a standard reply, whereas d4 is considered and discarded.



(c) In the window between 25k and 60k training steps, AlphaZero learns to put 80% of its mass on four replies to e4, and 20% of its mass on all other 16 moves.

Figure 4.1: Figure 7 from McGrath et al. [38].

ing what concept, *when*, indicating when in the training process, and *where*, indicating where in the network the concept is being measured. In some cases, interesting behavior is observed, such as a drop in linearity in the same layer during training or a drop in linearity in later layers compared to earlier layers. However, they do not explore the cause of it further than speculating what might have caused it. This observation is particularly interesting in light of our results in Paper C, where we showed how non-linear information, identified with a neural network probe, can increase simultaneously as its linear component decreases during training.

In their discussion about how the network identifies threats, such as whether the current player has a mate threat, they acknowledge that the network must also model the opponent's potential moves. They theorize that the network's ability to recognize these predictive concepts stems from the fact that AlphaZero is simultaneously learning the value and the policy. However, as we showed in Paper B, Stockfish's neural network can also grasp predictive concepts, even though it is trained without policy values.

Concerning concept probing, they raise multiple interesting questions; although addressing their preference for simpler linear probes, they question whether better probing architectures exist, such as some that are better at capturing spatially represented concepts. We specifically address this concern about the choice of probing architecture in Paper C, where we show that neural network probes may be more reliable than simpler linear probes. Furthermore, they question what we can infer from the probing accuracy, e.g., when can we say that a concept is really represented or used? They talk about how, sometimes, low probing accuracy can be due to the network architecture, e.g., earlier layers of convolutional neural network will not be able to identify longer-distance captures. Moreover, they acknowledge that sometimes, it is hard to identify whether we are identifying a confounder or the concept itself. Our results and discussion in Paper C about to what extent we can infer concept importance from these experiments add to the general conversation about what we can infer from the concept probing experiments, where we show the correlation between concept importance and probing accuracy using different probing methods.

Using non-negative matrix factorization, they identified 720 block/factor pairs. They successfully interpreted some of them, such as potential move computations and a count of the number of opponent's pieces that can move to a given square. However, they determined that it was not feasible to interpret many of the factors, especially those in later layers.

Finally, and less directly related to this thesis, they shed interesting light on the evolution of the agent through examples and qualitative analysis by chess grandmaster

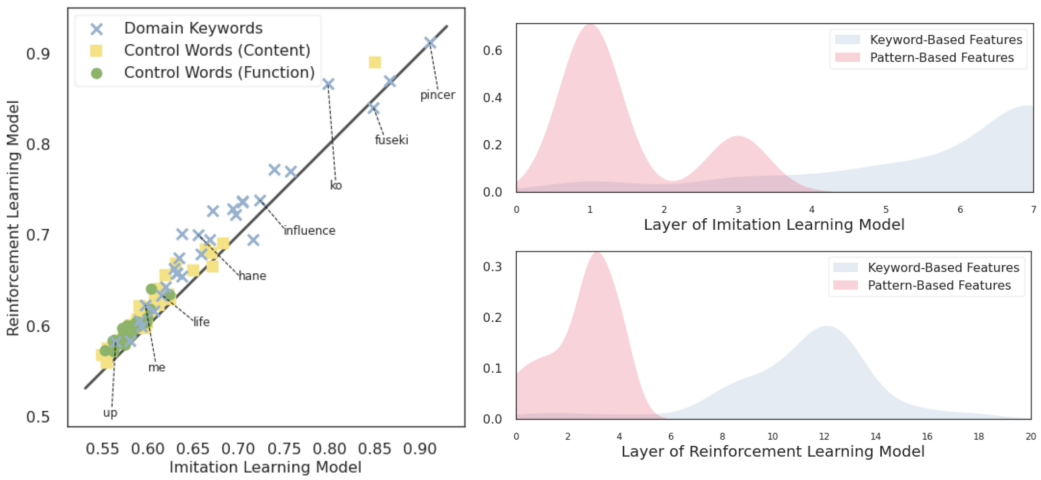


Figure 4.2: Figure 3 from Tomlin et al. [56].

Vladimir Kramnik. In Figure 4.1, we see how the prior over moves during the opening for the first 100k steps of self-play training. Interestingly, we see how AlphaZero starts considering 2. d4 as an answer to 1. e4 e5 and ultimately discards it for 2. Nf3. Kramnik shed some interesting light on how, during the first iterations, it is learning to evaluate material correctly. This is followed by an increased understanding of king safety and then correctly assessing which attacks will succeed in the corresponding order.

4.2 Other Related Work in Game-Playing

Tomlin et al. [56], **Understanding Game-Playing Agents with Natural Language Annotations**, analyzes the concepts learned by two agents, one trained using imitation learning (supervised learning, estimated to have about 1900 ELO) and one using self-play (reported to have Elo of over 5000). The concepts are split into two categories: pattern-based and keyword-based. Pattern-based concepts are extracted using a set of rules to decide if they are present. However, the keyword-based concepts approach uses the presence of keywords in commentary as a proxy for the presence of concepts within the observed state. They collected 10K games with move-by-move commentary from the Go Teaching Ladder(GTL). The commentary is in English and considered of high quality. For example, in the comment, "Bad shape. If white wants to defend it should be solid at c8, leaving no weaknesses or sente moves for black." they extract the keywords *shape* and *sente*. They use a linear concept probe, and their findings (as seen in Figure 4.2) show that the probing accuracy is significantly correlated between the models. Generally, the self-playing agent has a slightly higher accuracy than the agent trained using imitation learning. The high correlation is quite interesting considering that the self-playing agent is much stronger and supports our findings in Paper C that we may only find limited insights from concept probing accuracy alone. Furthermore, they show that the keyword-based concepts (which are used as a proxy for concepts of higher-level abstraction) are much better represented in the later layers of the model. In contrast, the pattern-based concepts are better represented in the earlier layers of the model, which aligns with the results in Lovering et al. [53].

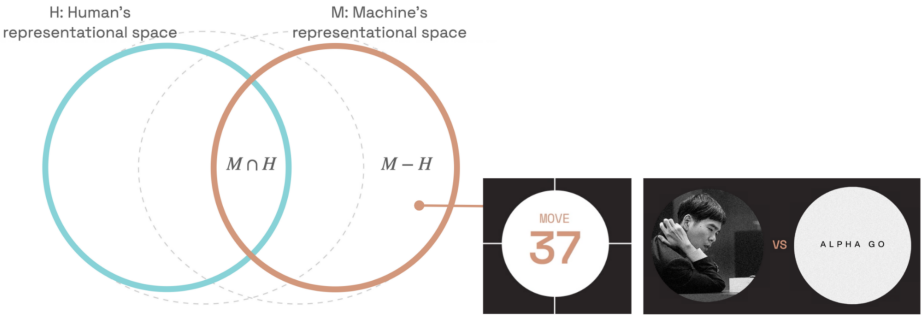


Figure 4.3: Figure 1 from Schut et al. 2023 [57].

Schut et al. [57], **Bridging the Human-AI Knowledge Gap: Concept Discovery and Transfer in AlphaZero**, uses AlphaZero and chess to develop methods to discover concepts so far unknown to humans and ultimately teaching them to humans, and therefore expanding human knowledge. We have human representational space (H), and machine representational space (M), where H represents the knowledge of humans and M represents the knowledge of machines (the knowledge of AlphaZero in this case). Their objective is to identify knowledge unknown to humans but known to machines (M-H) and teach them to some of the best chess players in the world (world top chess grandmasters). In contrast with our research, this paper does not rely on pre-defined concepts. The first step is to use unsupervised discovery of dynamic concepts using convex optimization to find concept candidates. The second step is to filter out concepts to ensure that they are novel (in the (M-H) space) and teachable. To ensure they are novel, they compare games of humans and AlphaZero using spectral analysis. To ensure the concepts are teachable (and useful), they teach another AI agent the concepts and filter them based on their informativeness. Finally, they confirm that they can expand the human representational space (H); they teach the top grandmasters the concepts through concept prototypes and quantitatively validate that the chess players increase their ability to find concept moves aligned with AlphaZero's choices. However, this is out of scope

Hammersborg and Strumke [58], **Reinforcement Learning in an Adaptable Chess Environment**, introduced an open-source computationally efficient adaptable-chess XAI research environment. It is estimated that AlphaZero played approximately 44 million self-playing games to reach its peak playing strength. This, combined with its large model size, makes it inaccessible to researchers with limited computational resources. They introduced an environment that implements Silverman 4x5 and Los Alamos 6x6 chess-variants. Furthermore, they also implement the linear concept probing methods used in McGrath et al. 2021 [38]. Examples of re-implementations of the what-when-where plots in McGrath et al. [38] can be seen in Figure 4.4.

Based on the recent success of large language models trained at scale Ruoss and Delétang, 2024 [59], **Grandmaster-Level Chess Without Search**, investigate whether it is possible to train an agent in a supervised manner that generalizes well without explicit search. They create a dataset by annotating games from Lichess using Stockfish 16 and train a transformer model to predict action-values without search (visualized in Figure 4.5). With a total of 10M games they get 530M state-value estimates to train on. Through testing they show that the transformer model reaches its strength only at sufficient dataset and model scale. To validate its performance, they tested it against humans, where it reached grandmaster level (Lichess blitz Elo 2895), and demonstrated

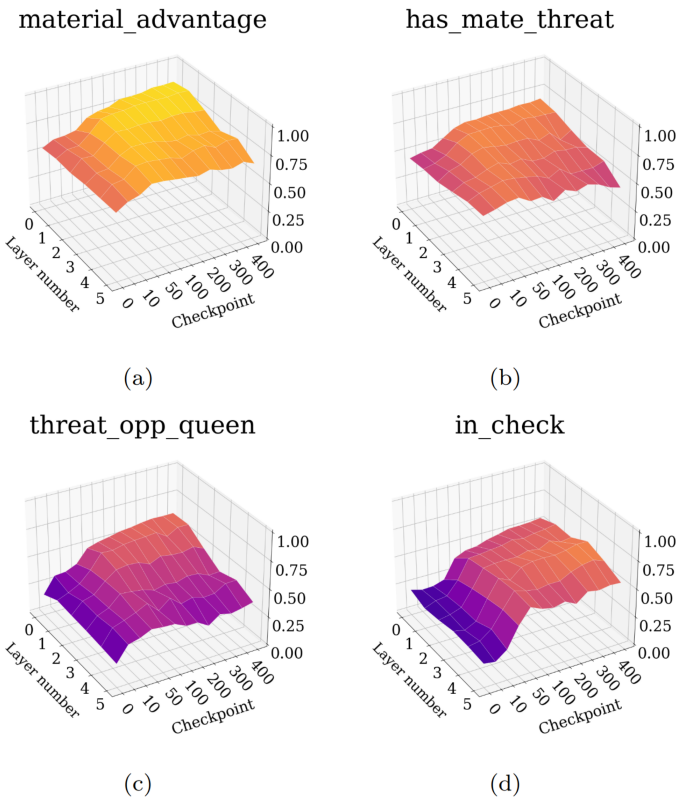


Figure 4.4: Figure 4 from Hammersborg and Strumke, 2022 [58].

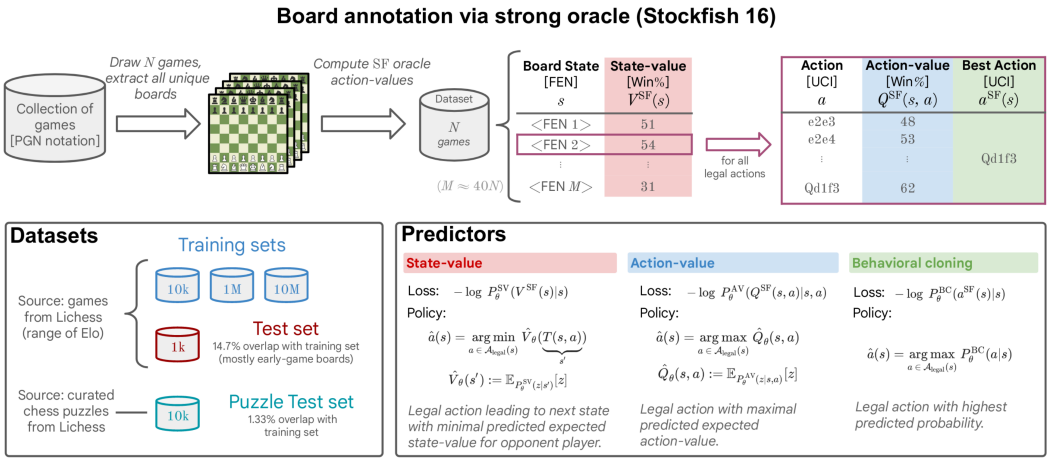


Figure 4.5: Figure 1 from Ruoss and Delétang [59].

that it could solve many challenging puzzles (up to Elo 2800). Its strength surpasses even AlphaZero (without search) and seems to generalize well on unseen board states. From our research perspective, it is interesting to see what kind of concepts the agent has learned because although the value function of Stockfish already incorporates predictive concepts, this model has an even better grasp of it. Also, it is interesting to see that given large enough models, one can learn tactical motives that replace search and allow statical evaluation. This is consistent with our finding in Paper B.

Chapter 5

Conclusion and Future Work

In the thesis, we investigated the intersection of game-playing agents and Explainable AI. We primarily focused on explaining the value estimation of game-playing agents, that is, understanding what knowledge is encoded in their evaluation models. The move decisions of game-playing agents combine both a model and a search, and understanding the model is a prerequisite to understanding the search. However, still, there is a definite interplay between the two because they complement each other. Furthermore, we briefly explored what might be done to bring explainability to search algorithms. A more comprehensive perspective, including both the model and the search, is a logical step for future work.

In this final chapter, we discuss our final conclusions, reflect on the research's answers and limitations, and conclude with suggestions for future work.

5.1 Conclusion

We defined four research questions to address in this thesis, each addressed in one of four papers. Three have been published, and the fourth and final one has been accepted for publication at ECAI-2024. This section begins by reiterating the questions and discussing how they were addressed. Then, we reflect on our research and discuss its limitations.

RQ1: *How effective/reliable are commonly used saliency map explainability methods in the game-playing domain?*

- We showed that the results from the saliency map methods differ significantly. The model agnostic methods, especially LIME and Shapley Value Sampling, provide much better intuition into the model's behavior than the model-specific ones. These methods do a better job of identifying critical pieces and putting high importance on pieces with a critical future role (securing a win). Therefore, it truly matters that these saliency maps are chosen and interpreted carefully.

RQ2: *How to use explainability methods to unveil the concepts learned by game-playing agents?*

- We demonstrated that we can unveil and compare the importance of game-playing concepts by analyzing an interpretable surrogate model. We trained the surrogate model on a special-purpose dataset where each sample was labeled with

the model's evaluation and game-playing concepts. Several interesting domain-specific insights were revealed about Stockfish's neural network model in contrast to its handcrafted evaluation model, such as the difference in evaluation of *king safety*.

RQ3: *What are the pitfalls and best practices in interpreting concept probing results in the game-playing domain?*

- We show that the widespread practice of using linear probes and interpreting their accuracy to indicate concept importance gives some, albeit limited, insights into concepts learned by the network. The probes' accuracy and feature importance is only moderately correlated. Thus, interpreting concept importance based on probe accuracy alone is somewhat unreliable. A better correlation is achievable using a more complex probe (neural network) and observing the difference in probing accuracy in the original and an impaired model.

RQ4: *How to bias the search process of game-playing agents to make more human-understandable decisions?*

- We demonstrated that we can use an explainability cost function to bias the search of a heuristic- and constrained-based solver towards finding solutions easily explainable to humans. By incorporating concepts from the literature, we can force the agent to make explainable decisions to humans of different expert levels. The explainability cost function incorporates human preference to reduce the effort required to understand or execute the decision.

Explainability in game-playing has received relatively little attention compared to fields such as image classification and natural language processing, although we feel that it is steadily increasing. Perhaps this is because researchers have focused mainly on solving games and defeating the best human players. Thus, we saw numerous opportunities to transfer established methods from these fields into the game-playing domain. Our first realization when we started was just how well-suited the game-playing domain is for answering questions related to explainability. It crystallizes in the domain's ability to develop evaluation methods grounded in the agent's behavior.

Therefore, we encourage XAI researchers to explore this domain because we are investigating general-purpose architectures, fully-connected and convolutional neural networks, as the insights gained can be widely applied. For example, our approach to unveiling the knowledge the neural network has learned using both surrogate models and concept probing may be applied in many domains. Furthermore, sticking with more complex probing methods should make the insights researchers aim to gain more robust to changes in representation.

We showed multiple examples of functionally-grounded evaluation methods using the game-playing domain. The compelling aspect of the domain is that the value estimation of the game-playing agents predicts the expected reward following the current policy, and thus, by explaining the value, we are, to some extent, also explaining what is about to happen. Meanwhile, by using self-play, we can see what is about to happen and compare the results of the self-play with the explanation. We hope this research may inspire others to consider game-playing as a setting for explainability research.

We started by applying this idea to saliency maps to evaluate their effectiveness in our domain. We approached that in two ways: first, by perturbing the state and

observing its impact using self-play. Although this provided valuable insights, the fact that it is rather obstructive inspired further development. Our second approach was perturbation-free, and instead of perturbing the state, we observed the future role of the pieces using self-play. By defining roles that we consider important, we can see how the evaluated importance from the saliency map is associated with the future role of the piece.

However, saliency maps are inherently limited to low-level features, so our next step was to move toward explanations using higher-level concepts. We demonstrated how we can unveil the knowledge learned by the agents using explainable surrogate models and a prominent method called concept-probing. The design and interpretation of concept probes are debated, and we realized that the game-playing domain is well suited for evaluating how we can interpret them, which resulted in our fourth and final paper. The key to that analysis was that we could easily compare agents' performance without relying on static and limited datasets. We evaluated the importance of information by hiding it during training and observing the decline in performance during competition. By comparing the results from different concept probing experiments to the evaluated importance of the concepts, we could make recommendations about probing architectures and experiments.

We mainly investigated four methods in this research: saliency maps, global surrogate models, concept probing, and an explainable agent. The insight we can gather from the research is partly limited by the experimental setting but also by the inherent limitations of the methods. For example, although we showed that some saliency maps correlate better with how we prefer to interpret them, we must acknowledge that simply stating that a piece is important may require considerable effort from the user to interpret. Due to the amount of interpretation delegated to the user, it might still misinterpret the explanation due to confirmation bias. However, identifying which methods better correlate with how we are likely to interpret them will reduce the likelihood of misinterpretation.

Furthermore, global surrogate models are intrinsically limited to concepts that describe an advantage in some way; perturbing the concept must affect the evaluation to be detected. Thus, important information that may either not be useful on its own or affect both players similarly will not be considered important using this method. However, this knowledge may still be important and used by the agent, and one should be mindful of that when interpreting the results from such methods.

As demonstrated, concept probing only provides limited correlation with concept importance and thus we need to be mindful when interpreting the results. Interpreting the absolute value of these probes is prone to misinterpretation. However, interpreting changes in the values of the probing experiments will provide a much better insight into the models, as demonstrated by our experiments in Paper C. However, one should still be careful when interpreting changes in the values using a linear probe, as seen in the autoencoder experiment; the information might merely be changing from a linear to a non-linear representation.

We showed how to design an explainable game-playing agent by incorporating knowledge from the literature, thus forcing them to make only explainable decisions. This is a feasible approach where a human is expected to verify, execute, or learn from the resulting solution. However, it is limited in cases where the implemented approaches are insufficient to solve the problem; in these circumstances, a hybrid approach could be feasible, with a fallback to conventional approaches when the solver gets stuck.

Finally, all of the abovementioned methods rely on pre-defined concepts. Therefore, so far, we can merely use or mirror our knowledge using these methods. With these limitations in mind and a keen awareness of the potential pitfalls of misinterpretation and confirmation bias when applying and interpreting these methods, they can provide valuable insights, as demonstrated in this thesis.

5.2 Future Work

During the research, we identified numerous future research avenues. For example, in Paper B, we identified a significant discrepancy in how the classical model and the neural network implement the *king safety* concept. Thus, a possible future work would be to figure out ways to increase the agreement between the methods by modifying the classical *king safety* concept. This could be performed by constrained optimization, adjusting the king safety concept to maximize the concept probing accuracy in the neural network. Performing this would require constraints by only defining dimensions that would fall within the concept of *king safety*; otherwise, we would start approximating the complete behavior of the neural network.

In Paper C, we introduce a method to estimate the importance of concepts to help us identify which probing algorithms to use. This approach can also be the foundation of multiple other concept probing studies. For example, it can be used to study other network architectures or to construct new probing methods. Researchers often favor either simpler or more complex probes. However, there may be a case for not choosing just one and instead developing techniques to interpret linear and non-linear concept probes simultaneously. Instead of ignoring some information, we could research ways to interpret measurements from multiple probes.

It would be advantageous to start viewing concepts from a more hierarchical point of view. This might make concept discovery more approachable. For example, most concepts implemented in the research are a combination of multiple simple rules that, when all, or specific combinations of them, are true, we consider the concept to be present. These simple rules might be meaningless on their own, but together, they form a useful concept. We can consider these simple rules to be low-level concepts, which are a part of higher-level concepts. Analyzing the concept probing accuracy of all sub-concepts might reveal new ways to interpret concepts. For example, what happens to the concept probing accuracy when introducing a new sub-concept to the higher-level concept? Or what can be said about a high-level concept from the concept probing accuracy of its sub-concepts?

Finally, our approach to delivering explainable search solutions can be further enhanced by implementing more complex concepts from the literature or adapting the solver to other games. We can further incorporate human preferences into the explainability cost function by optimizing its parameters using real games. However, we have only scratched the surface of explainability in search; the field is still maturing, and further effort could be made using other approaches, such as explaining search algorithms post-mortem. For example, when an agent changes its mind after a deeper search, what did it find out? The number of states explored is vast, and generating explanations using the contrast between states from promising and adverse paths might reveal interesting insight. Can the promising paths help us generalize the strengths and weaknesses of the current state? Furthermore, we can aggregate information from future states based on piece importance or analyze the hidden representation from a

concept-probing perspective, which could be an exciting avenue for future efforts. Do the promising paths have something in common? What are the most interesting or informative states explored by the search algorithm?

Bibliography

[1] C. Zhang and Y. Lu, "Study on artificial intelligence: The state of the art and future prospects", *Journal of Industrial Information Integration*, vol. 23, p. 100 224, 2021, ISSN: 2452-414X. DOI: https://doi.org/10.1016/j.jii.2021.100224. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2452414X21000248.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., 2012, pp. 1106–1114. [Online]. Available: https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

[3] EU, *THE AI ACT*, https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence, Accessed: 2024-05-17, 2024.

[4] X. Ferrer, T. Van Nuenen, J. M. Such, M. Coté, and N. Criado, "Bias and discrimination in ai: A cross-disciplinary perspective", *IEEE Technology and Society Magazine*, vol. 40, no. 2, pp. 72–80, 2021.

[5] T. Lombrozo, "The structure and function of explanations", *Trends in cognitive sciences*, vol. 10, no. 10, pp. 464–470, 2006.

[6] B. Kim, M. Wattenberg, J. Gilmer, *et al.*, "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV)", in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, J. G. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 2673–2682. [Online]. Available: http://proceedings.mlr.press/v80/kim18d.html.

[7] K. R. Popper, *The Logic of Scientific Discovery*. London: Hutchinson, 1934.

[8] T. R. X. d. Aguiar, "The symmetries of the hempelian model of explanation", *Kriterion: Revista de Filosofia*, vol. 2, no. SE, 2006.

[9] Wikipedia, *Chess*, https://en.wikipedia.org/wiki/Chess_theory, Accessed: 2024-05-17, 2024.

[10] Wikipedia, *Rules of chess*, https://en.wikipedia.org/wiki/Rules_of_chess, Accessed: 2024-05-17, 2024.

[11] J. Schaeffer, "2024 world computer chess championships: The end of an era 1974–2024", *ICGA Journal*, vol. 45, pp. 1–3, Apr. 2024. DOI: 10.3233/ICG-240243.

[12] Stockfish, *Stockfish: Strong open source chess engine*, https://stockfishchess.org/, Accessed: 2024-05-17, 2024.

[13] D. Silver, T. Hubert, J. Schrittwieser, *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play", *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[14] Stockfish, *NNUE readme*, https://github.com/glinscott/nnue-pytorch/blob/master/docs/nnue.md, Accessed: 2024-05-17, 2024.

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

[16] Stockfish, *Introducing NNUE evaluation*, https://blog.stockfishchess.org/post/625828091343896577/introducing-nnue-evaluation, Accessed: 2024-05-17, 2024.

[17] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of go with deep neural networks and tree search", *Nat.*, vol. 529, no. 7587, pp. 484–489, 2016. DOI: 10.1038/NATURE16961. [Online]. Available: https://doi.org/10.1038/nature16961.

[18] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search", in *International conference on computers and games*, Springer, 2006, pp. 72–83.

[19] C. D. Rosin, "Multi-armed bandits with episode context", *Annals of Mathematics and Artificial Intelligence*, vol. 61, no. 3, pp. 203–230, 2011.

[20] G. Schwalbe and B. Finzel, "A comprehensive taxonomy for explainable artificial intelligence: A systematic survey of surveys on methods and concepts", *Data Mining and Knowledge Discovery*, pp. 1–59, 2023.

[21] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Explainability in deep reinforcement learning", *Knowledge-Based Systems*, vol. 214, p. 106 685, 2021, ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2020.106685. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705120308145.

[22] Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang, "A survey on neural network interpretability", *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 5, no. 5, pp. 726–742, 2021. DOI: 10.1109/TETCI.2021.3100641. [Online]. Available: https://doi.org/10.1109/TETCI.2021.3100641.

[23] S. Lipovetsky and M. Conklin, "Analysis of regression in game theory approach", *Applied Stochastic Models in Business and Industry*, vol. 17, no. 4, pp. 319–330, 2001.

[24] C. Molnar, *Interpretable Machine Learning, A Guide for Making Black Box Models Explainable*, 2nd ed. 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book.

[25] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks", in *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., ser. Lecture Notes in Computer Science, vol. 8689, Springer, 2014, pp. 818–833. DOI: 10.1007/978-3-319-10590-1_53. [Online]. Available: https://doi.org/10.1007/978-3-319-10590-1%5C_53.

[26] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps", in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014. [Online]. Available: http://arxiv.org/abs/1312.6034.

[27] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks", in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, 2017, pp. 3319–3328. [Online]. Available: http://proceedings.mlr.press/v70/sundararajan17a.html.

[28] S. M. Lundberg and S. Lee, "A unified approach to interpreting model predictions", in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, *et al.*, Eds., 2017, pp. 4765–4774. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.

[29] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust you?": Explaining the predictions of any classifier", in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, Eds., ACM, 2016, pp. 1135–1144. DOI: 10.1145/2939672.2939778. [Online]. Available: https://doi.org/10.1145/2939672.2939778.

[30] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences", in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, 2017, pp. 3145–3153. [Online]. Available: http://proceedings.mlr.press/v70/shrikumar17a.html.

[31] J. Castro, D. Gómez, and J. Tejada, "Polynomial calculation of the shapley value based on sampling", *Computers & Operations Research*, vol. 36, no. 5, pp. 1726–1730, 2009, Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X), ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2008.04.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054808000804.

[32] S. M. Lundberg, G. G. Erion, and S.-I. Lee, "Consistent Individualized Feature Attribution for Tree Ensembles", 2018. arXiv: 1802.03888. [Online]. Available: http://arxiv.org/abs/1802.03888.

[33] G. Alain and Y. 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. [Online]. Available: https://openreview.net/forum?id=HJ4-rAVtl.

[34] E. Voita and I. Titov, "Information-theoretic probing with minimum description length", in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds., Association for Computational Linguistics, 2020, pp. 183–196. DOI: 10.18653/v1/2020.emnlp-main.14. [Online]. Available: https://doi.org/10.18653/v1/2020.emnlp-main.14.

[35] D. Hupkes, S. Veldhoen, and W. H. Zuidema, "Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure", *J. Artif. Intell. Res.*, vol. 61, pp. 907–926, 2018. DOI: 10.1613/jair.1.11196. [Online]. Available: https://doi.org/10.1613/jair.1.11196.

[36] N. F. Liu, M. Gardner, Y. Belinkov, M. E. Peters, and N. A. Smith, "Linguistic knowledge and transferability of contextual representations", in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Association for Computational Linguistics, 2019, pp. 1073–1094. DOI: 10.18653/v1/n19-1112. [Online]. Available: https://doi.org/10.18653/v1/n19-1112.

[37] R. H. Maudslay, J. Valvoda, T. Pimentel, A. Williams, and R. Cotterell, "A tale of a probe and a parser", in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds., Association for Computational Linguistics, 2020, pp. 7389–7395. DOI: 10.18653/v1/2020.acl-main.659. [Online]. Available: https://doi.org/10.18653/v1/2020.acl-main.659.

[38] T. McGrath, A. Kapishnikov, N. Tomasev, *et al.*, "Acquisition of chess knowledge in alphazero", *CoRR*, vol. abs/2111.09259, 2021. arXiv: 2111.09259. [Online]. Available: https://arxiv.org/abs/2111.09259.

[39] A. Conneau, G. Kruszewski, G. Lample, L. Barrault, and M. Baroni, "What you can cram into a single vector: Probing sentence embeddings for linguistic properties", in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, I. Gurevych and Y. Miyao, Eds., Association for Computational Linguistics, 2018, pp. 2126–2136. DOI: 10.18653/v1/P18-1198. [Online]. Available: https://aclanthology.org/P18-1198/.

[40] Y. Adi, E. Kermany, Y. Belinkov, O. Lavi, and Y. Goldberg, "Fine-grained analysis of sentence embeddings using auxiliary prediction tasks", in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=BJh6Ztuxl.

[41] T. Pimentel, J. Valvoda, R. H. Maudslay, R. Zmigrod, A. Williams, and R. Cotterell, "Information-theoretic probing for linguistic structure", in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds., Association for Computational Linguistics, 2020, pp. 4609–4622. DOI: 10.18653/v1/2020.acl-main.420. [Online]. Available: https://doi.org/10.18653/v1/2020.acl-main.420.

[42] A. Rosenfeld and J. K. Tsotsos, "Intriguing properties of randomly weighted networks: Generalizing while learning next to nothing", in *16th Conference on Computer and Robot Vision, CRV 2019, Kingston, ON, Canada, May 29-31, 2019*, IEEE, 2019, pp. 9–16. DOI: 10.1109/CRV.2019.00010. [Online]. Available: https://doi.org/10.1109/CRV.2019.00010.

[43] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Deep image prior", *Int. J. Comput. Vis.*, vol. 128, no. 7, pp. 1867–1888, 2020. DOI: 10.1007/s11263-020-01303-4. [Online]. Available: https://doi.org/10.1007/s11263-020-01303-4.

[44] A. Kumar, C. Tan, and A. Sharma, "Probing classifiers are unreliable for concept removal and detection", *Advances in Neural Information Processing Systems*, vol. 35, pp. 17 994–18 008, 2022.

[45] Y. Elazar, S. Ravfogel, A. Jacovi, and Y. Goldberg, "Amnesic probing: Behavioral explanation with amnesic counterfactuals", *Trans. Assoc. Comput. Linguistics*, vol. 9, pp. 160–175, 2021. DOI: 10.1162/tacl_a_00359. [Online]. Available: https://doi.org/10.1162/tacl%5C_a%5C_00359.

[46] A. Ravichander, Y. Belinkov, and E. H. Hovy, "Probing the probing paradigm: Does probing accuracy entail task relevance?", in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, P. Merlo, J. Tiedemann, and R. Tsarfaty, Eds., Association for Computational Linguistics, 2021, pp. 3363–3377. DOI: 10.18653/v1/2021.eacl-main.295. [Online]. Available: https://doi.org/10.18653/v1/2021.eacl-main.295.

[47] S. Ravfogel, Y. Elazar, H. Gonen, M. Twiton, and Y. Goldberg, "Null it out: Guarding protected attributes by iterative nullspace projection", in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds., Association for Computational Linguistics, 2020, pp. 7237–7256. DOI: 10.18653/V1/2020.ACL-MAIN.647. [Online]. Available: https://doi.org/10.18653/v1/2020.acl-main.647.

[48] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning", *arXiv preprint arXiv:1702.08608*, 2017.

[49] P. Dabkowski and Y. Gal, "Real time image saliency for black box classifiers", *Advances in Neural Information Processing Systems*, vol. 2017-Decem, pp. 6968–6977, 2017, ISSN: 10495258. arXiv: 1705.07857.

[50] N. Puri, S. Verma, P. Gupta, *et al.*, "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. [Online]. Available: https://openreview.net/forum?id=SJgzLkBKPB.

[51] J. Fritz and J. Fürnkranz, "Some chess-specific improvements for perturbation-based saliency maps", in *2021 IEEE Conference on Games (CoG), Copenhagen, Denmark, August 17-20, 2021*, IEEE, 2021, pp. 1–8. DOI: 10.1109/COG52621.2021.9619015. [Online]. Available: https://doi.org/10.1109/CoG52621.2021.9619015.

[52] T. McGrath, A. Kapishnikov, N. Tomašev, *et al.*, "Acquisition of chess knowledge in alphazero", *Proceedings of the National Academy of Sciences*, vol. 119, no. 47, e2206625119, 2022.

[53] C. Lovering, J. Forde, G. Konidaris, E. Pavlick, and M. L. Littman, "Evaluation beyond task performance: Analyzing concepts in alphazero in hex", in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., 2022. [Online]. Available: http://papers.nips.cc/paper%5C_files/paper/2022/hash/a705747417d32ebf1916169e1a442274-Abstract-Conference.html.

[54] A. Immer, L. T. Hennigen, V. Fortuin, and R. Cotterell, "Probing as quantifying inductive bias", in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., Association for Computational Linguistics, 2022, pp. 1839–1851. DOI: 10.18653/V1/2022.ACL-LONG.129. [Online]. Available: https://doi.org/10.18653/v1/2022.acl-long.129.

[55] Stockfish, *Stockfish*, https://stockfishchess.org/blog/2024/stockfish-16-1/, Accessed: 2024-05-17, 2024.

[56] N. Tomlin, A. He, and D. Klein, "Understanding game-playing agents with natural language annotations", in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., Association for Computational Linguistics, 2022, pp. 797–807. DOI: 10.18653/V1/2022.ACL-SHORT.90. [Online]. Available: https://doi.org/10.18653/v1/2022.acl-short.90.

[57] L. Schut, N. Tomasev, T. McGrath, D. Hassabis, U. Paquet, and B. Kim, "Bridging the human-ai knowledge gap: Concept discovery and transfer in alphazero", *CoRR*, vol. abs/2310.16410, 2023. DOI: 10.48550/ARXIV.2310.16410. arXiv: 2310.16410. [Online]. Available: https://doi.org/10.48550/arXiv.2310.16410.

[58] P. Hammersborg and I. Strümke, "Reinforcement learning in an adaptable chess environment for detecting human-understandable concepts", *CoRR*, vol. abs/2211.05500, 2022. DOI: 10.48550/ARXIV.2211.05500. arXiv: 2211.05500. [Online]. Available: https://doi.org/10.48550/arXiv.2211.05500.

[59] A. Ruoss, G. Delétang, S. Medapati, *et al.*, "Grandmaster-level chess without search", *CoRR*, vol. abs/2402.04494, 2024. DOI: 10.48550/ARXIV.2402.04494. arXiv: 2402.04494. [Online]. Available: https://doi.org/10.48550/arXiv.2402.04494.

[60] *Muzero-general*, `https://github.com/werner-duvaud/muzero-general`, Accessed: 2022-01-03.

[61] T. R. Team, *Ray provides a simple, universal API for building distributed applications.* Accessed on 2022-01-03, https://docs.ray.io, 2022. [Online]. Available: `%5Curl%7Bhttps://docs.ray.io%7D`.

[62] Stockfish, *Training Data Sets*, `thtps://github.com/glinscott/nnue-pytorch/wiki/Training-datasets`, Accessed: 2024-05-17, 2024.

[63] Wikipedia, *Elo Rating System*, `https://en.wikipedia.org/wiki/Elo_rating_system`, Accessed: 2024-05-17, 2024.

Appendix A

Evaluating Interpretability Methods for DNNs in Game-Playing Agents

Evaluating Interpretability Methods for DNNs in Game-Playing Agents

Aðalsteinn Pálsson[1] and Yngvi Björnsson[1][0000−0001−5366−2639]

Department of Computer Science, Reykjavik University

Abstract. There is a trend in game-playing agents to move towards an Alpha-Zero-style architecture, including using a deep neural network as a model for evaluating game positions. Model interpretability in such agents is problematic. We evaluate the applicability and effectiveness of several saliency-map-based methods for improving the interpretability of a deep neural network trained for evaluating game positions, using the game of Breakthrough as our testbed. We show that the more applicable methods provide insights into the importance of the different game pieces and other domain-dependent knowledge learned by the model.

Keywords: game-playing · model-interpretability · deep neural-networks

1 Introduction

Over the past few decades, research into game-playing programs for abstract strategy board games has first-and-foremost concentrated on developing new techniques and algorithms for improving gameplay. That work has resulted in super-human strength game-playing agents [9] for disparate games such as chess, checkers, Othello, Go, and many more. At the same time, other important aspects of intelligent systems have been mainly neglected, such as how to explain the rationality for one's actions in human-understandable terms. Moreover, recent advancements in the field where game-playing agents use deep neural networks (DNNs) to evaluate board positions and action selection in the think-ahead process make the decision-making process even more non-transparent.

The above-mentioned lack of transparency is not specific to game-playing agents. As computer-generated models in disparate fields such as healthcare and banking have become increasingly ubiquitous, the need for humans to understand their decisions becomes increasingly crucial for establishing trustworthiness. This has spurred research interest in *model interpretability*, that is, the development of approaches to make it less complicated for humans to understand the cause of models' decisions in terms of their inputs. Several such approaches now exist, for example, in the field of image recognition, which has hitherto been at the forefront of both deep neural network and model-interpretability research. In particular, *saliency maps* [10] have become a popular way of visualizing which regions of an image are primarily responsible for a given classification decision.

In this work, we evaluate the applicability and effectiveness of several saliency-map-based methods for improving the interpretability of a neural network trained

to evaluate game positions of a board game, using the game Breakthrough as our testbed. The paper's primary contributions are: (i) We evaluate several popular saliency-map-based methods within recently established paradigms and taxonomies for black-box interpretability methods; and (ii) show how they can be applied to interpret a deep neural network model for an abstract board-game — a domain they are not mainly intended for; and, finally; (iii) assiduously evaluate and rank the methods by their effectiveness in our domain. Furthermore, this works adds to the recently emerging literature on explaining models and actions learned by game-playing agents [6, 4].

The paper is organized as follows. Section 2 introduces the terminology and preliminaries. Section 3 explains the game-playing agent, model, and evaluation methods used. In Section 4, which constitutes the main body of the work, we introduce and analyze the finding of the empirical evaluations of the saliency-maps methods. Finally, in Section 5, we conclude and discuss future work.

2 Background

We start with a high-level overview of the model-interpretability methods we investigate, before explaining the rules of the game of Breakthrough.

2.1 Model Interpretability

The taxonomy of explanation methods of black-box models categorizes them as either *global* or *local* and *model-specific* or *model-agnostic*. Global methods create explanations valid across all input instances, while local methods' explanations are specific to individual instances. Model-agnostic methods explain any black-box models, while model-specific methods leverage the model's architecture.

The most straightforward local method is *occlusion*, where the model's output sensitivity to leaving out (zeroing) arbitrary input parameters is investigated [15]. Such an approach, where applicable, is appealing as it is both model-agnostic and straightforward to implement.

A method that is local and model-specific but still requires no ad-hoc work is to analyze the gradient of the output with respect to individual input pixels [10]. Multiplying the input with the gradient is also often preferable because it leverages the strength of the input. A further extension on the gradient method is Integrated Gradients [12], which relies on a baseline and interprets the input feature attribution as the integration of gradients on the straight-line path between the input and the baseline. GradientShap [5] is an extension of Integrated Gradients that computes the expected gradient by sampling baseline values.

DeepLIFT [8] is a model-specific explanation method that does not rely on the gradient. It overcomes the limitations of loss of information when the gradient is zero because the signal might still be meaningful. It calculates the importance in a backward fashion by distributing attributions, or blame, in terms

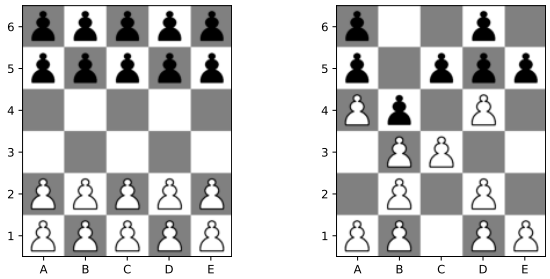


Fig. 1. Breakthrough: Initial board position (left); Example position (right)

of difference-from-reference. For all neurons, a difference-from-reference is calculated by passing through the input sample and the reference. Finally, it calculates the importance using predefined rules, such as the linear- or reveal cancel rule.[1]

LIME [7] is a local model-agnostic method, which uses an interpretable surrogate model to explain the black-box model. A local model, such as a linear model, is trained on a dataset derived from sampling noise around the input and using the model evaluation as a target. In the case of a linear model, the weights from the model serve as feature attributions.

The Shapley value [11] is a concept from cooperative game theory that can be used to calculate feature attribution. There are multiple ways to approximate the Shapley values [5], the one we use in this paper is Shapley Value Sampling. It takes random permutations of the input and adds them one by one to the baseline. This is repeated multiple times to approximate the Shapley values.

2.2 Breakthrough

The game Breakthrough is an abstract strategy board game, originally played on a 7x7 board but later popularized to an 8x8 board. The game can be played on different-sized (not necessarily squared) boards. In this work we use a smaller variant of 5x6, mainly for the ease of demonstration.

The game is a two-player turn-taking game. The players are referred to as White and Black, respectively. The board is initially set up by placing White's pieces along the first two rows and the Black pieces on the last two rows, as shown in Fig. 1 (left). White goes first and then players alternate, with each player moving one of their pieces per turn. A piece moves one square straight or diagonally forward (relative to the player). A straight forward move is allowed to an empty square only, but a diagonally forward moves may also capture an opponent's piece. For example, in Fig. 1 (right) the white pawn on $d2$ has two moves, to $d3$ or $e3$, and the piece on $d4$ also has two moves, to $c5$ or $e5$, both

[1] A baseline, or a reference, may be interpreted as a neutral state of a neural network, and is important for defining counterfactual arguments [12]. When assigning an attribution/blame to the input, it is done relative to the baseline. Most of the methods we consider in this paper rely on a baseline defined as all-zeros.

with a capture. The first player to get a piece across the board wins: White wins by moving a piece onto the last row, and likewise, Black wins by moving a piece onto the first row. If all pieces of a player are captured, that player loses. It follows from the rules that one of the players always wins (there are no draws).

3 Methods

An Alpha-Zero-like agent for playing the game of Breakthrough was developed for the paper.[2] The following subsection providing details, whereas the next subsection gives an overview of our evaluation methodology.

3.1 The Model

We trained a AlphaZero-like model [9], $(p, v) = f_\theta(s)$ with parameters θ, where p is the policy and v is the value function. The value is the output from a tanh activation function, a scalar value between -1 and 1. And the policy is a tensor with three channels, where each channel is the same size as the board and encodes the three different move directions available for all the pieces, i.e., forward, and diagonally left, or -right.

The model consists of a body of 5 residual blocks with 56 filters, followed by the policy and value heads. The input to the model is a tensor with three channels, where each channel is the same size as the board. The first channel encodes the board positions of the active player (the player to move), the second channel encodes the positions of the opponent, and the third channel is a binary encoding of the active player's color. If it is white to move, then the third channel is all ones. Otherwise, it is all zeros. In our case, we use a board with six rows and five columns. The search is performed with the Monte Carlo tree search algorithm like AlphaZero. The training procedure was via asynchronous self-play, where each move played used 200 simulations.

We deviate from AlphaZero we only feed the model the current board position. We do this mainly to make the model easier to explain, as an explanation should not depend on previous board positions.

3.2 Evaluation Measures

The goal of this paper is to compare and evaluate model explanation methods in the domain of game-playing. Our model is returning a value estimate, and the goal is to explain the estimation. When evaluating the usefulness of the explanations, we will consider if we can use the explanations to gain trust in the trained model, if the explanation satisfies our human curiosity, and if we can interpret some meaning from the model [1].

The evaluation approaches are split into three categories [3]: (i) Functionally-grounded, where the explanation is evaluated without human, using a proxy as

[2] We have no objective measure of the agent's playing strength, but as anecdote, it consistently wins all humans it plays, including an expert-level chess player.

an indication of explainability; (ii) human-grounded, requiring a human with non-expertise to evaluate a simple explanation and; (iii) application-grounded, which requires a human-expert, evaluating an explanation for a real-world task.

At first, we will inspect the saliency map from a qualitative human-grounded perspective. We will briefly debate if the saliency maps match our human objectives by visualizing statistics from the saliency methods.

Quantitative evaluation will be in the form of functionally grounded experiments. We will define three tasks where the performance will be used as a proxy for explanation quality. First, we will assess if the saliency method assigns the highest saliency to a critical piece, and conversely, if the lowest saliency is assigned to a non-critical piece. We will analyze this as an ablation study, where we separately remove the least and most important piece and measure its impact on the game's outcome. The second proxy task is to analyze the saliency of the piece that ultimately secures the win in a self-play game. The third task is to find the smallest sufficient subset of pieces required to retain a winning position. Then we iteratively remove non-important pieces according to an explainability method. The explainability method that has the highest area under the curve has the highest explainability quality.

4 Results

We ran several experiments, both for contrasting the effectiveness of different model-interpretability methods and for gaining added insights into the domain-dependent knowledge captured by the learned model.

4.1 Experimental Setup

The model is implemented in Pytorch and trained using asynchronous self-play using two GeForce RTX 3090, AMD Ryzen 9 5950 with 64 GB of RAM, and using RAY [14]. It was trained while playing a total of 630,000 self-play games. The training used stochastic gradient descent with a batch size of 512 and a decaying learning rate that was re-initialized every few thousand iterations.

LIME's perturbation of the input was binary, i.e., the features were set either to zero or one, allowing a more direct comparison with the occlusion method.

The saliency map methods in the paper used the implementations in the model interpretability library Captum [13].

4.2 Saliency Methods: Qualitative Evaluation

In image classification, a saliency map is a two-dimensional image, representing each pixel's perceived importance for the model's output. The analogy for a board game would be an image of the board's squares, representing the square's (or the piece on it) influence on the model's evaluation of the current game state.

Model-agnostic methods typically rely on modifying the input somehow and observing its effect on the model's output. That way, each input feature's attribution (or importance) to the output can be determined. One of the most

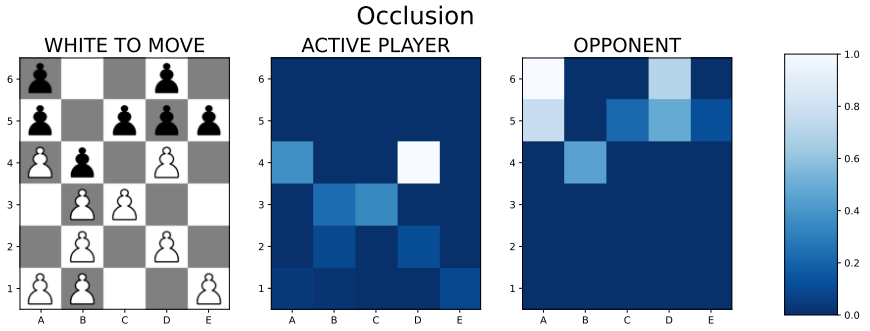


Fig. 2. The saliency map generated by Occlusion; the lighter a square is, the more important the piece occupying the square is.

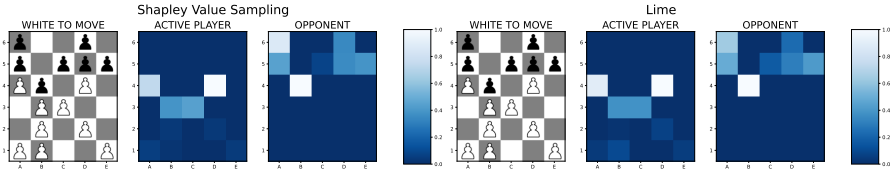


Fig. 3. Model-agnostic saliency methods: Shapley Value Sampling and Lime

straightforward of such ablation methods is that of *occlusion*, where in our domain we remove a piece from the board and observe the effect of the model's output. Fig. 2 shows the saliency map from such an experiment. [3] It shows clearly that the attacking piece on $d4$ is White's primary asset along with the supporting piece on $c3$ (and the independently potential breakthrough piece on $a4$). Unsurprisingly, for Black, the defending pieces on $a6$, $a5$, $d6$ and $d5$ play the most crucial role. This assessment is in perfect consonance with human (expert-level) assessment: the white pieces on $d4$ and $c3$, with White to move, can collectively win the game on their own, while the piece on $a4$ is a valuable long-term asset severely restricting the mobility of two of the black pieces, thus potentially placing Black later in *zugzwang*, but a well-established expert-level strategy in playing Breakthrough is to force such situations.

We ran the same type of experiment for two additional model-agnostic algorithms, *Shapley Value Sampling (SVS)* and *LIME*, which also find attribution by perturbing the model input parameters. However, they do it in a more refined way, potentially detecting non-trivial input interactions (as described in Section 2). The two methods give almost identical results, depicted in Fig. 3, with the result for the most part also consistent with the one from the occlusion method. The only significant difference is that for Black, the defending player, the piece on $b4$ gets much-added importance (without that defending piece, White will have additional ways to win by immediately playing a piece to that square).

[3] For ease of comparison, the maps are scaled to be in the range [0.0-1.0].

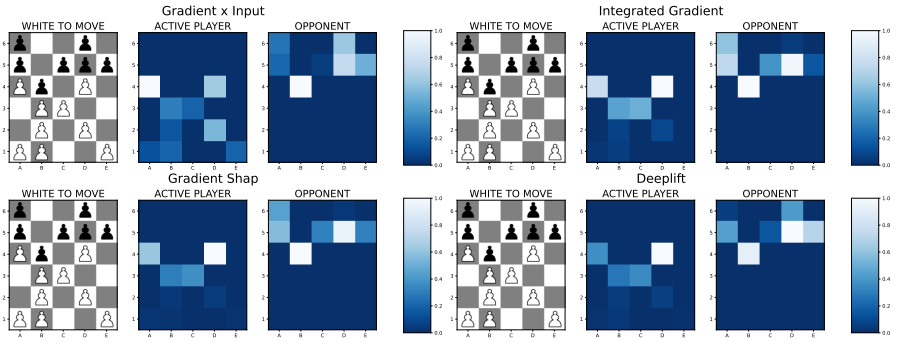


Fig. 4. Model-specific saliency methods.

We furthermore experimented with a few model-specific approaches as taking advantage of the model's internals may (in theory) yield further benefits. We looked at several methods that use the model's gradient in different ways to detect attribution, and one non-gradient-based, Deeplift. Fig. 4 shows the resulting saliency maps. Again, the more sophisticated methods, Integrated Gradient, GradientShap, and Deeplift, all give intuitively plausible results, whereas the straightforward gradient-based method is more indecisive.

4.3 Saliency Methods: Quantitative Evaluation

Qualitative evaluation as in the previous subsection, albeit able to provide valuable insights, is not sufficient for determining the relative effectiveness of the different saliency map algorithms — a quantitative approach is needed for that.

We generated 10,000 game positions from self-play by stopping play randomly 10-30 moves (plies) into the game. We played two games from each position for each saliency-based approach: one without intervention and one after removing the most important piece for the player to move as judged by the respective saliency method. The expectation is that the more reliable indicators of a most-importance piece suffer more profound drop in proportion of games won.

Table 1 summarizes the results. We see the expected effect in all cases but most profoundly for the *Occlusion* method followed closely by *LIME*, *SVS* and *DeepLift*. This gives us added confidence that these saliency methods are detecting the importance of the different pieces.

In an attempt to further discriminate the effectiveness of the different approaches in detecting valuable pieces, we looked at how important a pawn reaching the opponent's back rank was judged a few moves earlier. One can think of that information as an indicator of how quickly a particular saliency method realizes the importance of such "breakaway" pieces. Fig. 5 shows that information, and apparently, $LIME$ and SVS seem to put much-added importance on such pieces, whereas *Occlusion* and *Gradient* do not.

Table 1. The positions are placed into 9 bins based on their evaluation. The top-most rows shows proportion of games won by the player to move, for each bin, and the next rows show the same after removing the highest ranked piece. We also include the average and standard deviation of all methods after removing the lowest ranked piece.

Method	Importance	Proportion of games won								
Nothing deleted	-	**0.09**	**0.26**	**0.36**	**0.44**	**0.51**	**0.55**	**0.64**	**0.74**	**0.90**
Occlusion	Highest	0.08	0.21	0.26	**0.29**	**0.31**	**0.37**	**0.40**	**0.43**	**0.50**
LIME	Highest	**0.04**	0.18	0.26	**0.29**	0.34	0.39	0.45	0.47	0.51
SVS	Highest	**0.04**	0.19	**0.24**	0.32	0.38	**0.37**	0.44	0.45	0.51
Gradient	Highest	0.12	0.23	0.35	0.41	0.49	0.50	0.57	0.63	0.66
Integ. Gradients	Highest	0.06	0.18	**0.24**	0.32	0.38	0.41	0.44	0.47	0.56
Gradient Shap	Highest	0.05	**0.15**	0.27	0.31	0.36	0.41	0.45	0.49	0.57
Deeplift	Highest	**0.04**	0.18	**0.24**	0.34	0.35	0.40	0.46	0.47	0.51
Average	Lowest	0.10	0.27	0.37	0.44	0.50	0.56	0.62	0.75	0.91
Std Dev	Lowest	0.01	0.03	0.05	0.06	0.06	0.05	0.04	0.02	0.00
Mean bin value (before deletion)		**-0.90**	**-0.68**	**-0.45**	**-0.22**	**0.00**	**0.22**	**0.45**	**0.67**	**0.90**

It is also of interest to investigate how confidently the methods rank the less important pieces. For that, we use the (functionally grounded) method of smallest sufficient subsets [2], which in our domain translates into the set of pieces required to retain a winning position. To create a test-suite, we sampled positions from random games according to the model's value function to find positions in the current player being only a slight favorite. Then we gradually removed the pawns considered least important, one at a time, and recorded its effect on the proportion of games won. Fig. 6 depicts the result. Essentially, the later a curve drops, the more effective the respective saliency method is in ranking pieces by importance. There are two clear winners, $LIME$ and SVS, and two methods that do notably worse than the others, *Occlusion* and *Gradient*.

4.4 Explaining the Explanations

Finally, we were also interested in knowing common higher-level characteristics of pieces judged valuable. One way to unveil such characteristics is to build a sur-

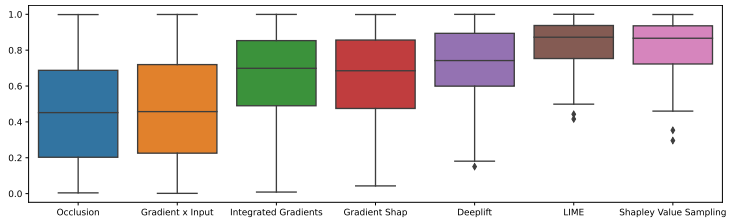


Fig. 5. Distribution of saliency values of a piece 4-ply prior to the winning move.

Method	Importance	Proportion of games won								
Nothing deleted	-	**0.09**	**0.26**	**0.36**	**0.44**	**0.51**	**0.55**	**0.64**	**0.74**	**0.90**
Occlusion	Highest	0.08	0.21	0.26	**0.29**	**0.31**	**0.37**	**0.40**	**0.43**	**0.50**
LIME	Highest	**0.04**	0.18	0.26	**0.29**	0.34	0.39	0.45	0.47	0.51
SVS	Highest	**0.04**	0.19	**0.24**	0.32	0.38	**0.37**	0.44	0.45	0.51
Gradient	Highest	0.12	0.23	0.35	0.41	0.49	0.50	0.57	0.63	0.66
Integ. Gradients	Highest	0.06	0.18	**0.24**	0.32	0.38	0.41	0.44	0.47	0.56
Gradient Shap	Highest	0.05	**0.15**	0.27	0.31	0.36	0.41	0.45	0.49	0.57
Deeplift	Highest	**0.04**	0.18	**0.24**	0.34	0.35	0.40	0.46	0.47	0.51
Occlusion	Lowest	0.08	0.26	0.31	0.38	0.46	0.50	0.60	0.73	**0.91**
LIME	Lowest	0.11	0.28	0.42	0.47	0.55	0.61	0.64	0.77	**0.91**
SVS	Lowest	0.09	**0.30**	**0.44**	**0.52**	**0.58**	**0.62**	**0.67**	**0.78**	0.90
Gradient	Lowest	0.10	0.27	0.40	0.48	0.52	0.58	0.63	0.73	**0.91**
Integrated Gradient	Lowest	0.11	0.22	0.32	0.37	0.43	0.51	0.57	0.73	**0.91**
Gradient Shap	Lowest	**0.12**	**0.30**	0.40	0.49	0.54	0.59	0.65	0.76	**0.91**
Deeplift	Lowest	0.10	0.27	0.31	0.38	0.45	0.50	0.58	0.72	**0.91**
Mean bin value (before deletion)		**-0.90**	**-0.68**	**-0.45**	**-0.22**	**0.00**	**0.22**	**0.45**	**0.67**	**0.90**

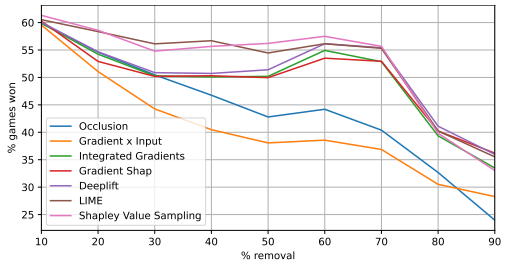


Fig. 6. Effect of gradually removing the least-important pieces.

rogate model from hand-made higher-level features and then train the surrogate model to predict the saliency values.

We build such a surrogate model using a LightGBM regression tree. Fig. 7 shows the relative importance of higher-level features we defined for the model; it clearly shows how important it is in Breakthrough to have advanced pieces, but features such as a center-of-mass are also important.

5 Conclusion and Future Work

This paper evaluated several popular saliency-based model interpretability methods on a DNN based game-playing agent, demonstrating their usefulness for identifying the most and least essential game pieces. The more sophisticated attribution methods, like Shapley Value Sampling and LIME, performed overall the best. One of the strengths of those methods is that they can capture non-trivial interactions between the inputs, which seems well suited to identify vari-

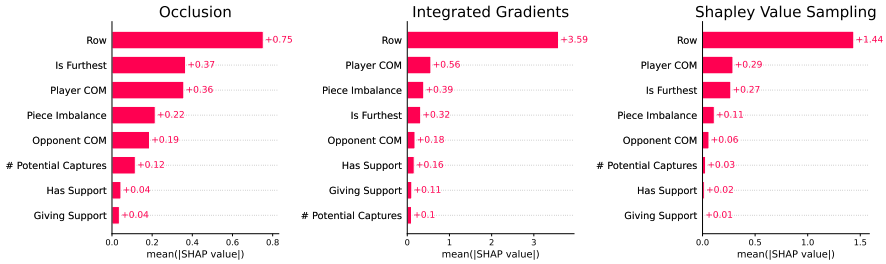


Fig. 7. The average Shapley Values of each of the inputs to the interpretable surrogate model. Here COM stands for center of mass. Has- and Giving Support indicate if the piece is supporting or giving support diagonally to a piece of the same color. # Potential Captures indicates is the number of available capture moves for the current player.

ous in-game piece dynamics. Moreover, those methods are both model-agnostic, making them well-suited for a wide range of models.

As future work, we plan to evaluate the methods' applicability in other games to better establish their usefulness in the domain of abstract board games. Also, we only scratched the surface of looking at higher-level domain concepts (beyond piece importance), and further research in that direction holds promise.

References

1. 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.
2. Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. *Advances in Neural Information Processing Systems*, 2017-Decem:6968–6977, 2017.
3. Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv: Machine Learning*, 2017.
4. Jessica Fritz and Johannes Fürnkranz. Some chess-specific improvements for perturbation-based saliency maps. In *IEEE Conference on Games (CIG) 2021, Copenhagen, Denmark*. IEEE, 2021.
5. 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.
6. 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 *ICLR*, 2020.
7. Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?". In *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pages 1135–1144. ACM Press, 2016.
8. Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *34th International Conference on Machine Learning, ICML 2017*, 7:4844–4866, 2017.

9. David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815, 2017.
10. Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2014.
11. Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41(3):647–665, 2014.
12. Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *34th International Conference on Machine Learning, ICML 2017*, volume 7, pages 5109–5118, 2017.
13. The PyTorch Team. Model interpretability and understanding for PyTorch., 2021. Accessed on 20-09-2021, https://captum.ai.
14. The Ray Team. Ray provides a simple, universal API for building distributed applications., 2021. Accessed on 20-09-2021, https://docs.ray.io.
15. Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science*, 8689 LNCS(PART 1):818–833, 2014.

Appendix B

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

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.

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

[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.

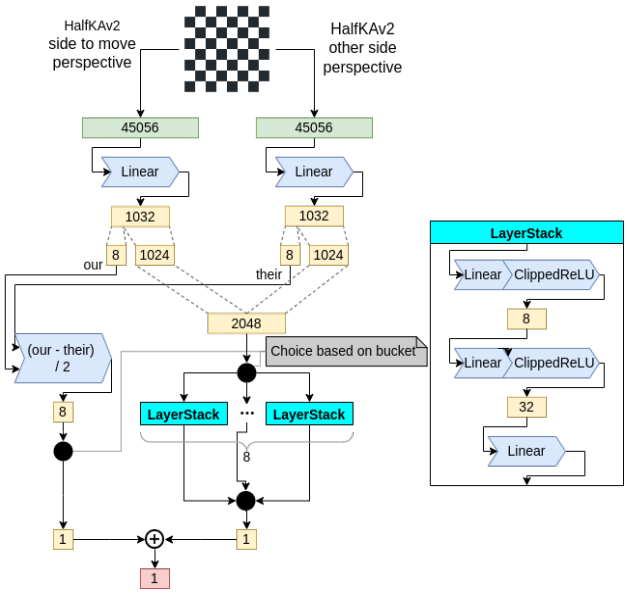


Figure 1: Stockfish' NNUE architecture [Stockfish, 2022b]. It processes the values depending on the game phase (8 buckets, depending on the number of pieces), i.e., it learns different PSQT values and uses a different layer stack (sub-network) for each bucket.

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.

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

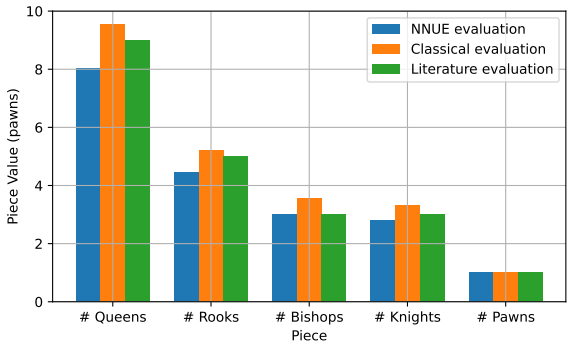


Figure 2: Piece values using Concept Regression. Each feature corresponds to #_* features in Table 2, i.e. its value is the difference in number of each piece type on the board.

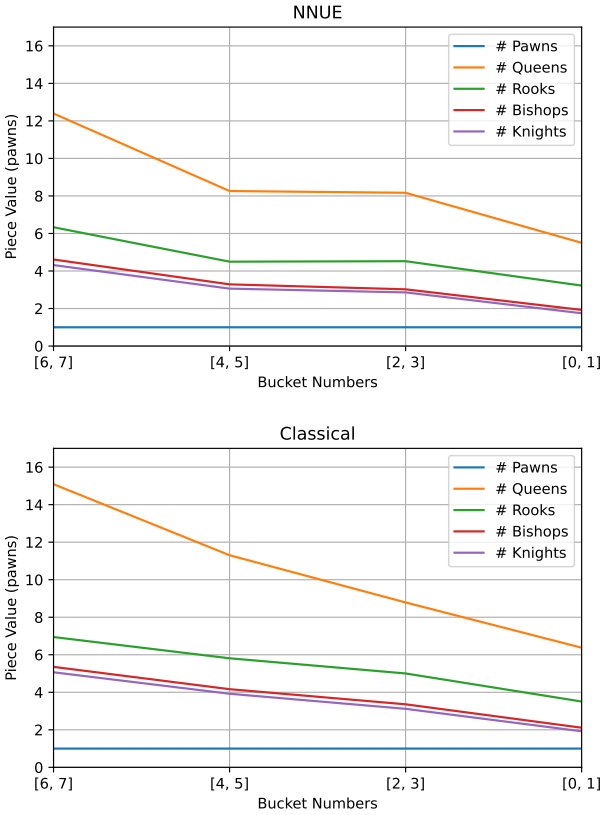


Figure 3: Comparison between the piece values depending on the phase of the game. The bucket number indicates which layer stack will be used; the number of pieces on the board determines the bucket number, which is calculated by $(piece\_count - 1)/4$.

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

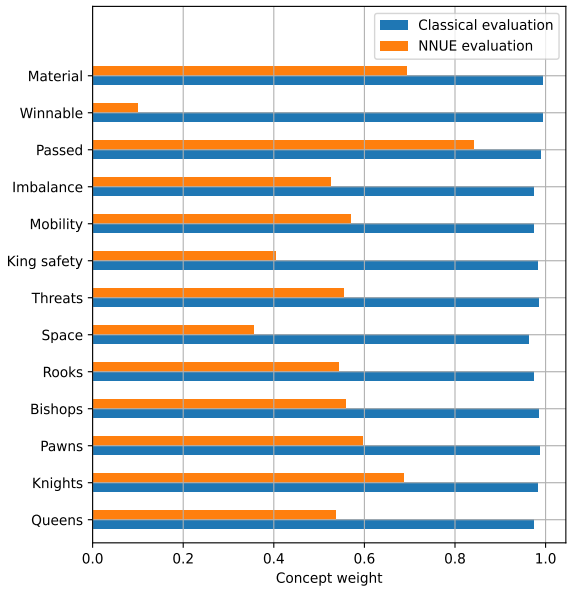


Figure 4: This figure shows the weights of two linear surrogate models used to explain the Classical and NNUE evaluations, respectively. The fitted models give a r^2-score of 0.999 for the classical evaluation and 0.497 for the NNUE evaluation, respectively.

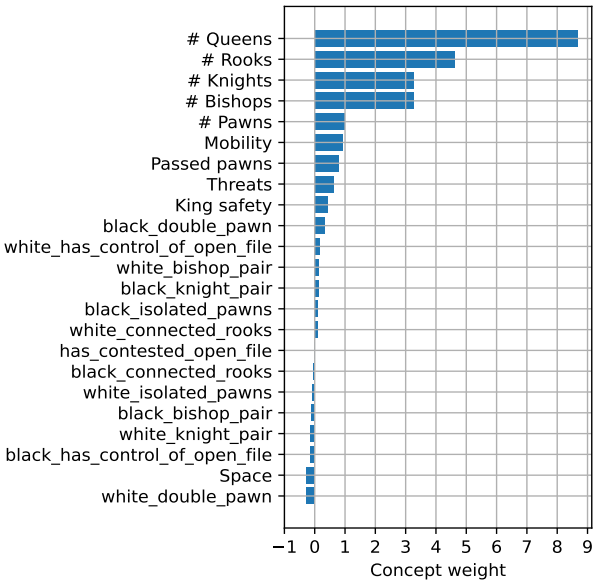


Figure 5: Surrogate model predicting NNUE's evaluation weights. We exclude *material* and *imbalance* to avoid concept interaction. It is advantageous to have a *bishop pair*, to have *control of an open file* and *connected rooks*, and disadvantageous to have *doubled pawns*.

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

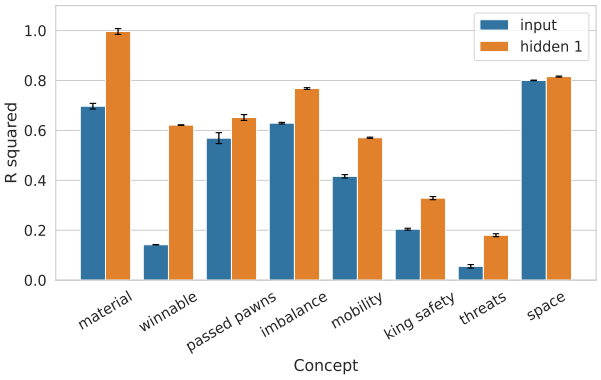


Figure 6: Concept probing results on i) the input features and ii) after the first hidden layer, using the official NNUE weights.

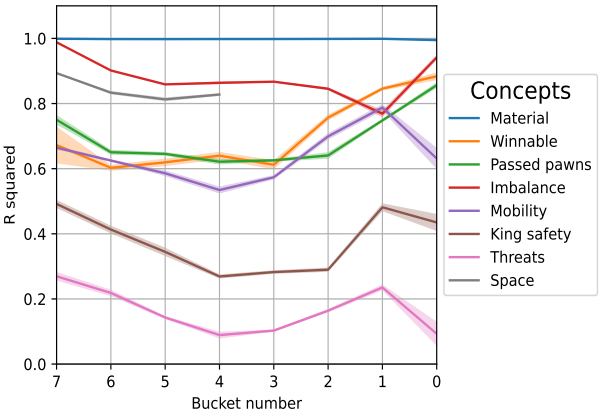


Figure 7: This figure shows concept probing results on subsets containing only a single bucket using ridge regression, evaluated at the first hidden layer of the model, using the official NNUE weights. The bucket number indicates which layer stack will be used; the number of pieces on the board determines 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.

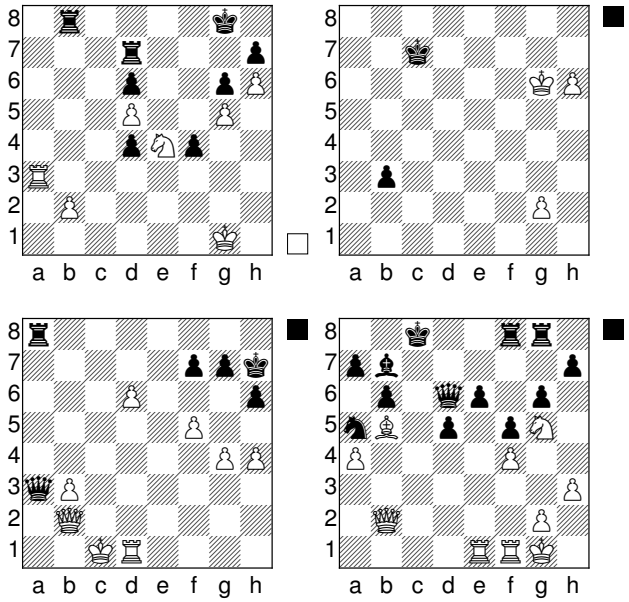


Figure 8: (Upper Left) The NNUE statically detects the *fork* Nf6 favouring White unlike the classical model; (Upper Right) Black (to move) wins because promoting with a check, seen (statically) by the NNUE but not the classical model. (Bottom) The classical model judges king safety the most critical 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.

Appendix C

Empirical Evaluation of Concept Probing using a World-Class Game-Playing Agent

Empirical Evaluation of Concept Probing for Game-Playing Agents

Aðalsteinn Pálsson and Yngvi Björnsson

Department of Computer Science, Reykjavik University

Abstract. Concept probing is one prominent methodology for interpreting and analyzing (deep) neural network models. It has, for example, formed the backbone of several recent works to understand better the high-level knowledge learned and employed by game-playing agents, particularly in chess. However, some recent theoretical and empirical studies have questioned the methodology's reliability and highlighted some limitations. Here, in the game-playing domain of chess, we investigate the effectiveness of several different probing architectures and look into the reliability of methods for interpreting their results. We use a world-class chess-playing agent as our test domain, which allows us, via self-play, to quantify the importance of the concepts identified in the agent's neural network by the concept probes. Our results demonstrate that the widespread practice of using linear probes and interpreting their accuracy to indicate concept importance is somewhat unreliable and needs to be revised. We demonstrate several ways of doing that in our domain, particularly by using more complex probes and amnesic-like probing.

1 Introduction

Artificial intelligence systems employing machine-learning models are increasingly deployed in real-life settings, partly because of the recent successes of deep-learning neural-network-based approaches. Unfortunately, one drawback of that approach is the black-box nature of the neural networks and the need for more interpretability and explainability of the decisions they make. Explainable AI (XAI) seeks to rectify this, offering several methodologies to help improve the models' interpretability.

Concept probing is one prominent methodology for interpreting and analyzing (deep) neural network models [2]. Given a neural network trained on some task, concept probing aims to gain insights into the extent to which the network's internal layers have learned to represent various (high-level) concepts. This is done by training a separate classifier/regression model predicting a concept of interest using a layer's activations as input. If the resulting classifier performs well, the presumption is that the layer has learned to represent the given concept. This methodology has gained momentum and is now widely used to gain insights into neural network models in diverse domains such as natural language processing [5], image recognition [9], and game-playing [18]. However, some recent theoretical and empirical studies have cast doubt on how reliable this methodology is and highlighted some limitations [3, 6, 26].

Here, we investigate the effectiveness and reliability of concept probing in game playing using a world-class chess agent as our testbed, comparing different probing architectures and interpretation methods. The reasons for choosing this domain are threefold: first, there have been several recent high-profile works in XAI in game playing [18, 14, 21]; second, to address concerns raised in that work as to how reliable the probing results are; and three, because the domain allows us to quantify (via self-play) the importance of the identified concepts for the agent's playing strength (we contrast that information to the concepts' importance as judged by the probes).

The primary contributions of the work for our domain are: *i)* We empirically demonstrate that the widespread practice of using linear probes and interpreting their accuracy to indicate concept importance could be more reliable and should be revisited; *ii)* We propose ways of doing that, including by using more complex probes and amnesic-like probing techniques, where applicable; *iii)* We offer additional insights into the pitfalls and best practices of concept probing, including concrete and relevant examples of potential failures.

The remainder of the paper is organized as follows. The next section introduces the terminology and background, followed by our methods and empirical result sections, respectively. Finally, we conclude and discuss future work.

2 Background

This section gives a brief background of model interpretability, generally and in games, and discusses some of the criticisms raised.

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 real-life situations (see e.g. [4, 19] 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).

Most prior work on local methods for interpreting models use feature-based explanations, which alter the input features (e.g., occlude or perturb them) [15, 27]. Such approaches are especially helpful in image-based domains and can reveal how different pixels or regions in an image contribute to the network's output classification. Similar approaches have been used to explain chess positions [24].

However, the methods are not readily applicable to intricate concepts. Furthermore, the feature-based methods are known to be unreliable [10] and susceptible to confirmation bias [9]. Thus, to overcome this shortcoming, more recent explanation techniques, often

referred to as *concept-based*, use high-level human concepts as their interpretable units.

2.2 Concept Probing

A recent concept-based model-specific interpretability methodology for "peeking" into DNNs is *probing* [2]. 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. By training classification or regression surrogate models (on a dataset not used for training the network itself) — called probes — to predict a given concept from a layer's activations, one can measure how much information that layer carries regarding the given concept. The higher the prediction accuracy of the probe, the more information that layer carries for representing the concept.

When concept probing we train a model, or a probe, g to map representations $f_l(x)$ at layer l to concepts z. As a proxy for how well f represents the concepts, one evaluates

$$performance\_metric(g(f_l(x)), z) \tag{1}$$

on unseen test data, and reports the results using a relevant performance metric, such as accuracy, as we do in this paper. This means that an accuracy value of 0.5 infers layer l does not contain information regarding concept z. In contrast, an accuracy value of one indicates that concept z can be fully decoded from layer l in model f. The choice of probing architecture, g, will depend on the experiment and whether the concept is binary or continuous. Researchers still debate the specific choices of probing architecture: some advocate for simpler linear probes, while others advocate for more complex ones, such as neural networks. The detailed nuances related to the specific choice of probing architecture will be discussed further in Section 2.3.

The model training and probing process requires two separate datasets: the training dataset $D_t = \{x^i, y^i\}$ used to train the model, f, and a separate probing dataset $D_p = \{f_l(x^i), z^i\}$, used to train the probe g. Where $f_l(x^i)$ is the internal represetions in layer l for sample x^i and z^i is the concept's value for the corresponding sample. The probe is usually trained on a balanced dataset for binary concepts, meaning that the majority class is undersampled.

When designing the probing dataset D_p, a few things need to be considered, including that the more complex probe architectures (e.g., neural networks) typically call for larger training datasets than the simpler ones (e.g., linear). Thus, we opt for an extensive training dataset D_p of one million positions to ensure that all the probing architectures we compare have had sufficient training data.

2.3 Challenges Related to Concept Probing

What kind of probes are most useful for concept probing is still actively debated. While some researchers have advocated for simpler probes [2, 7, 13, 16, 17], others have advocated for more complex probes [5, 1, 23]. The debate is mainly focused on what kind of results we can safely interpret from the probes; for example, it can be argued that too powerful of a probe could learn its own mappings between the representations and the concept (not present in the original model), and conversely, for too simple probes, a probe might not accurately reveal complex (non-linear) representation that the model has learned.

To gain further insight into the probed model, some additional experiments have been proposed to identify to what extent the model is learning a concept. For example, evaluating a lower- and upper bound for the value has been proposed to put the probing accuracy's numerical value into perspective. A lower bound can be found by probing an untrained network; this can help identify a few concerns, for example, to help attribute to which extent the probing accuracy is because of model learning [5] rather than other factors.

An upper bound can be seen as a way to show how well, theoretically, a mapping from the input to the concept could perform; this can be achieved by training a dedicated model $h(x)$ to predict the concept z given the input, x. Although, in theory, this might seem like a viable option, one drawback is that it is likely very resource-intensive because the models required for such an experiment are likely larger than needed for regular probing and are more data-intensive. After all, they might require similar amount of data to the one used to train the original model f.

If concept probing is a reliable tool for identifying the presence of concepts, in theory, it should be able to guide the removal of information from the network. However, the work in [11] demonstrated that it fails to remove the concepts entirely and, in some cases, destroys other task-relevant features because the probing classifier is likely to use non-concept features to predict the presence of the concept.

Although concept probing only measures how well model representations can be mapped to a concept, it is tempting to infer that the model is using the concept. However, researchers have found some evidence of false positives, meaning that high-accuracy probing does not necessarily mean that the concept benefits the task of model f [6]. In that respect, one attractive property of the game-playing domain we use is that we can quantify via self-play how useful a concept really is for a model.

To further investigate how to interpret these experiments, researchers have proposed various interventions that provide further contrast to the results; one such experiment is to erase the concept from the model training data (as we also do in this paper) $D_o = \{x^i, y^i\}$ [26]. By erasing the concept from the training data, we can monitor performance degradation on the original task, which puts the importance of the concept into perspective, as well as changes in probing accuracy. Interestingly, Ravichander et al. [26] showed that by removing a concept from the training data, the probe may still be able to predict some properties of the concept.

Elazar et al. [6] coined the term *amnesic probing*, an extension to conventional probing, to describe an approach for studying behavioral changes of a model after removing concept information from its representations using Iterative Nullspace Projection (INLP) [25]. They show that even after removal, subsequent layers can recover some of the removed properties which, to some extent, speaks to the importance of non-linear information within the model because INLP only removes its linear components. In this paper we will refer to probing for concepts removed from the training set as *amnesic-like probing* due to its similarity with amnesic probing [25] – both method probe for removed information.

Although this is still active research, a few things are still uncertain, e.g., to what extent can we infer that the model uses the concept while predicting, and second, perhaps more importantly, what are some actionable insights that may be gained through concept probing?

2.4 Interpretability in Games

Interpretability research in game-playing has received added attention in recent years. Game-playing is a domain with diverse challenges; for example, chess, which we use as a testbed in this research,

has pieces with different roles and relatively simple rules yet sophisticated playing strategies, which makes for exciting challenges related to explainability.

The saliency method SARFA [24] is a perturbation approach to evaluating action-focused saliency that tackles specifically the problem that pieces affect value function differently. E.g., if removing a queen affects all future states similarly, including the proposed action, it is irrelevant to the proposed action.

The concept-probing work reported in McGrath et al. [17, 18] investigates the concepts that AlphaZero's neural network learned. They use Stockfish's hand-crafted concepts (among others) as a proxy for human-understandable knowledge. Using concept probing, they show that the network represents many human-understandable concepts and how and when they emerge during training. They demonstrate that material concepts get represented early in training, while more complex and subtle ones appear later. Furthermore, the paper briefly discusses alternative probing architectures; they advocate for simpler linear classifiers due to the danger of the concept probe learning its own complex relationships rather than capturing the structure in AlphaZero's representations.

The work in Pálsson and Björnsson [21] showcases how it is possible to identify what Stockfish's neural network has learned using various interpretability methods, including concept probing. It exposed Stockfish's neural network emphasis on dynamic concepts and how it evaluates king safety differently compared to its hand-crafted counterpart.

In Lovering et al. [14], researchers analyze high-level concepts that an AlphaZero Hex playing agent learned. They demonstrate that the search discovers concepts during self-play before the neural network learns to encode them. Furthermore, they show that short-term end-game planning is best encoded in the final layers of the model, while long-term planning concepts are encoded in the middle layers of the model.

Tomlin et al. [32] extracted keywords from move-by-move Go commentary and compared them with simpler pattern-based concepts. Using a linear probe, they show that the keyword-based concepts (which are used as a proxy for concepts of higher-level abstraction) are better represented in the later layers of the model. In contrast, the pattern-based concepts are better represented in the earlier layers of the model.

Most concept-based methods now rely on predefined concepts; however, in Schut et al. [28], researchers explore ways to extract and successfully teach top chess grandmasters concepts beyond human knowledge.

3 Methods

In this section, we first describe Stockfish's neural-network evaluation function, followed by a description of the probing methods used in the paper. Subsequently, we describe the Autoencoder used in the first experiment and the calculation of piece-specific importance, as well as discuss the chess-specific concepts implemented for the research.

3.1 Stockfish' Neural Network

Stockfish (since version 12) uses a neural network called NNUE [20] (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) [30, 31].

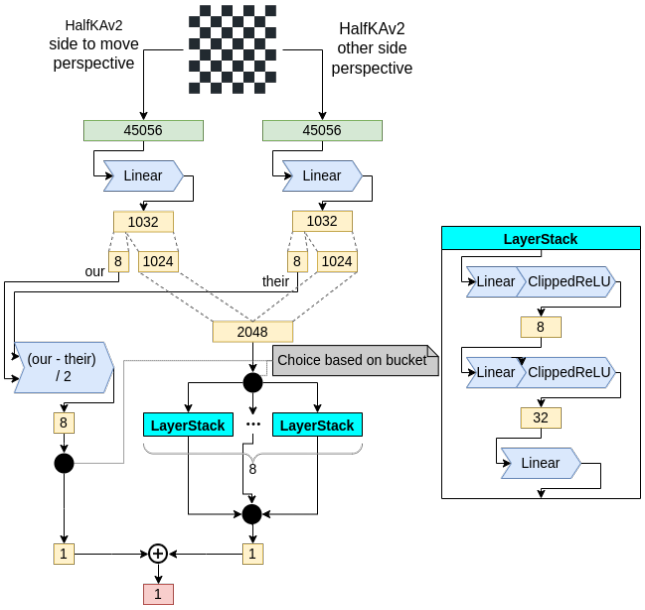


Figure 1. Stockfish' NNUE architecture [31]. It processes the values differently depending on the game phase. Depending on the total number of pieces left on the board, it will assign a bucket value and choose the corresponding path depending on the value. The bucket value is calculated by $bucket = (piece\_count - 1)/4)$.

Table 1. Autoencoder design, output sizes of each layer of the autoencoder.

Layer	Size
Input	768
Layer 1	1024
Layer 2	64
Layer 3	32
Layer 4	64
Layer 5	1024
Layer 6 (reconstruction)	768

The NNUE architecture uses a (shallow) design with linear and clipped ReLU layers, as shown in Figure 1. Notable design choice is routing the inference through different LayerStacks, or sub-networks, depending on the game's phase (number of pieces on the board). Thus at each time, only one of eight LayerStack is used to calculate the evaluation. The LayerStacks follow only a single linear layer shared between all LayerStacks. This design choice is interesting from a concept-probing perspective because one would expect each LayerStack to re-implement much of the same concepts.

The NNUE model is relatively shallow, and in our experiments, we probe the model solely after two layers (Layer 2). After a single linear layer, static information (invariant of the learning process) from the input is still relatively accessible. The probe only needs to decode the weights of a single linear layer; therefore, we want to look deeper. However, going even deeper into Stockfish's neural network will introduce new challenges; the computation splits into multiple pathways depending on the game phase, making the experimental evaluation more intricate. Thus, we chose layer two as the best practical compromise.

3.2 Probing Architectures

We experiment with three types of probing architectures in this paper; a linear probe using ridge [22] classification (Ridge), a neural network probe (NN), and a LightGBM [8] decision tree probe

Table 2. The name and description of concepts used in the probing experiments.

Concept	Description
white_has_queen	White has a queen
white_queen_on_ initial_square	White has a queen on D1
Queen Advantage	Only one has a queen(s)
Rook Pair Advantage	Only one has a rook pair
Bishop Pair Advantage	Only one has a bishop pair
Knight Pair Advantage	Only one has a knight pair
Opposite Color Bishop	Both have only one bishop of opposite color
5 Dark Square Pawns	5 pawns are on dark squares

Table 3. Description of Δ piece value concepts.

Concept Name	Description
ΔP	Difference in number of pawns
ΔN	Difference in number of knights
ΔB	Difference in number of bishops
ΔR	Difference in number of rooks
ΔQ	Difference in number of queens

(LGBM) . The neural network probe uses feedforward architecture with ReLu nonlinearity and two hidden layers of sizes 100 and 10. For the linear 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 decision tree probe is used with default parameters.

3.3 Autoencoder

We designed an autoencoder experiment to contrast the different probing architectures. Within this controlled environment, we can investigate whether the different probes are a good proxy for the information stored within the representations. When we encode and decode information, we can observe the reconstructed state to know how much information passes through the network - for us to decode information successfully, it must also be available in the encoded state. This method monitors the probing accuracy during training in the most compressed and reconstructed states. It allows us to observe inconsistencies in probing accuracy values and trends.

Inside the autoencoder, we expect considerable compression of information, which is relevant when probing a small neural network, such as Stockfish's NNUE, where the concept representations are likely more compressed than in a larger neural network.

The autoencoder consists of two parts, an encoder and a decoder, which are trained to compress and reconstruct the input space. In this case, the encoder and decoder mirror each other in terms of the number of layers and neurons in each layer. The design uses a sequence of fully connected linear layers with ReLU activations.

The input into the autoencoder is a tensor of size 768, or 64 (squares) $\times$ 6 (piece types) $\times$ 2 (color), and the output size of each layer is listed in the Table 1.

3.4 Piece Importance

By training and interpreting a linear surrogate model, we can evaluate how different piece-specific concepts impact the evaluation relative to each other. We perform this analysis to illustrate the model's behavior and alleviate concerns that we might be causing more harm to the model than intended.

For example, to identify the relative importance of different pieces, we can train the linear surrogate model:

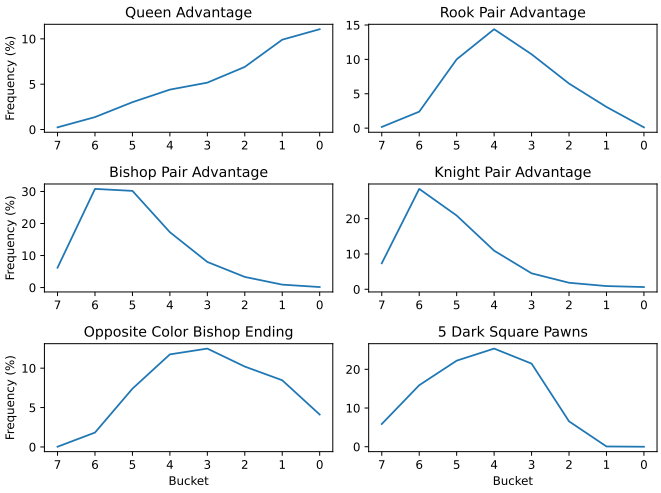


Figure 2. The frequency of the concepts in relation to the phase of the game, or bucket. Each value on the graph represent the proportion of each bucket that the concepts' value was true(and thus removed during training). The bucket indicates the number of pieces left on the board, where bucket 7 has the most and 0 has the fewest.

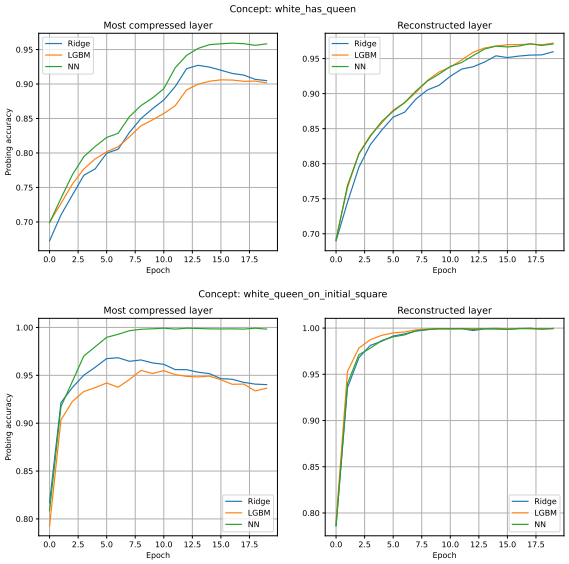


Figure 3. Probing accuracy results performed during training of the autoencoder. Results are for the concept white_has_queen (top) and white_queen_on_initial_square (bottom).

$$w_p * \Delta P + w_n * \Delta N + w_b * \Delta B + w_r * \Delta R + w_q * \Delta Q = f(x) \quad (2)$$

where $f(x)$ is the model evaluation, the ΔP ... ΔQ concepts (seen in Table 3) indicate the difference in a number of pieces, e.g., ΔB indicates the difference in the number of bishops present on the board.

In this case, the linear weights for the different concepts will serve as a proxy for importance, and we can, for example, compare the weights w_p and w_q. Thus, we interpret $\frac{w_q}{w_p}$ as the queen value evaluated in a number of pawns as we do in Figure 4.

3.5 *Custom Concepts*

We implemented several chess-specific concepts for this research, all listed in Table 2.

First, we implement two straightforward concepts for the autoencoder experiment, *white_queen_on_initial_square* and *white_has_queen*. The concept *white_queen_on_initial_square* is represented by a single bit in the autoencoder's input, while *white_has_queen* can be represented by any of a total of 64 bits, assigned to represent the queen(s) position(s) on the board.

The latter concepts, also listed in Table 2, are used for training handicapped agents where we hide each one of these concepts, in turn, from the agents. This approach allows us to objectively evaluate the importance of each of these concepts by comparing the resulting agents' playing strengths. We chose concepts known from the chess literature either as essential or not. For example, having a pair of bishops or knowing how to dynamically evaluate the queen's strength compared to the other pieces (e.g., vs. two rooks or three minor pieces) is critical, whereas having a pair of knights or rooks is less so. The concept of having exactly five pawns on a dark square was explicitly contrived with the expectation of being irrelevant to evaluating a chess position. Although the choice of the above concepts was inspired by established human-based chess knowledge of what is important and what is less so, we also ran experiments to quantify how important these concepts were for the neural network model and that it was in agreement with established chess theory (see Section 4).

When defining the concepts for the restricted agents, we also had a few considerations in mind. First, the concept can not be too frequently active, where the frequency alone can be destructive to the dataset. Removing positions from the training dataset with a given concept will inevitably affect the distribution of seen training positions over the different game phases; however, we verified that all game phases are still well represented, as seen in Figure 2. Second, it is important that when the concepts become active, they can also become deactive, so we do not remove all endgames associated with the concept and other concepts that may arise later in the game.

4 Results

Here, we present the results of our experiments. First, we describe the experimental setup, followed by the results of the autoencoder, concept importance, and probing accuracy with respect to concept importance experiments, respectively.

4.1 *Experimental Setup*

We use version 15.1 of Stockfish, which was the latest release at the start of the research.

For training, we use a dataset generated by *Leela Chess Zero* that is listed as a quality dataset (*training_data* at [29]). For concept probing, we sample one million chess positions from a Lichess dataset generated from standard rated chess games [12]. When training a modified agent, f_{res,c_i}, where concept c_i is hidden, we check each sample for concept c_i. If it is present, we discard the sample and continue sampling until the desired batch size is reached. Each agent is trained for 500 epochs using Stockfish's official training script (with slight modifications to remove chosen concepts during training) with recommended hyperparameters: batch size was set to 16384, initial learning rate as 8.75e-4, and gamma (learning rate decay rate) as 0.992.

Table 4. Results in a gauntlet style tournament, where all agents agents only play against the Regular agent. Each modified agent plays 3000 matches at search depths 14, 15 and 16 each, thus a total of 9000 matches.

Description	Win-rate %	Elo drop
Queen	$45.51(\pm 0.54)$	$31.6(\pm 3.8)$
Bishop Pair	$46.91(\pm 0.51)$	$21.7(\pm 3.6)$
Rook Pair	$48.33(\pm 0.49)$	$11.7(\pm 3.4)$
Knight Pair	$48.57(\pm 0.49)$	$10.0(\pm 3.8)$
Opposite Color Bishop	$49.04(\pm 0.48)$	$6.7(\pm 3.3)$
5 Dark Square Pawns	$49.74(\pm 0.48)$	$1.8(\pm 3.3)$

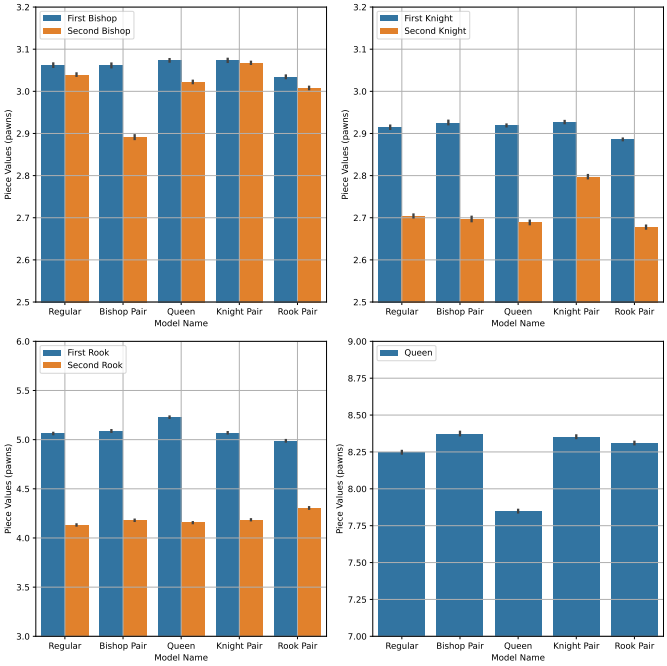


Figure 4. (upper left) When we remove a bishop pair advantage from the training data, the model associates a lower importance to the second bishop. (upper right) Removing a knight pair advantage from the training data the model puts greater importance on the second knight. (lower left) Removing a rook pair advantage from the training data has little impact on the value it puts on having a second rook. (lower right) Never seeing examples where one player has a queen and the other does not, the model associates a lower value to the queen.

The autoencoder was trained using Adam with the learning rate set as 5e-4. It finished with an accuracy of 0.9973 and a Binary Cross Entropy loss of 7.27e-3, meaning that it was able to reconstruct its input almost perfectly.

All models were trained on a GeForce RTX 3080 graphics card. The error bars provided in the figures show one standard deviation above and below the mean from cross-validation over five splits.

4.2 *Autoencoder Experiment*

In this section, we present the results from the concept probing experiment where we monitor what happens during autoencoder training, shown in Figure 3. By observing the concept probing accuracy at the reconstructed layer, we can form an expectation of a lower bound for the amount of information present in the most compressed layer; it should be at least as much as in the reconstructed layer. Because if it is present at the reconstructed layer, it should also be present at the most compressed layer. Therefore, we base our interpretation of the

results on observing how the concept probing accuracy in the reconstructed state evolves during training and comparing it to the concept probing accuracy in the most compressed layer.

At each training epoch, we probe for two simple concepts, *white_has_queen* and *white_queen_on_initial_square*. The concept's value is true if the statement is true and false otherwise. The probing is performed at the most compressed layer as well as the reconstructed layer. The encoded size is 32, while the reconstructed size is 768 – like the input.

The most important takeaway is that interpreting the probing accuracy at the most compressed state using the linear probe will lead to misleading results. For both concepts, the linear probe's (Ridge) probing accuracy decreases after saturating in the most compressed layer. At the same time, we see that at the reconstructed layer, the probing accuracy is either increasing or has reached saturation at 100% accuracy. Meanwhile, the neural network probe more accurately represents the presence and trend of information passing through the autoencoder as reflected by the reconstructed layer.

Albeit this is a somewhat contrived experimental scenario, this result nonetheless shows that linear probes may be insufficient to detect information present in the internal layers of a neural network, especially when one expects the information to be highly compressed.

4.3 Estimating Concept Importance

The attractive characteristic of the game-playing domain is that we can easily compare agents by making them compete, thus assessing their strength without relying on a potentially biased datasets. For this experiment, we choose six concepts well-suited for removal during training.

In order to identify the importance of a concept, we will: i) train an agent with and without the knowledge of a concept, ii) make them compete to assess the performance decline, and iii) expose how they evaluate positions differently. The tournament will be in a gauntlet style, where the restricted agents play against the regular agent (trained on the entire dataset). From the results, we then calculate the relative Elo rating of all agents, where the difference in Elo points serves as (a proxy for) the importance of the concept.

In Table 4, we see the results from the tournament. The *Queen Advantage* concept is the most important, resulting in a drop in performance of 31.6 Elo points, while the concept *5 Dark Square Pawns* shows a non-significant drop in performance. The difference between *Knight Pair Advantage* and *Rook Pair Advantage* is also non-significant, and it is approximately half of the importance of *Bishop Pair Advantage*.

To demonstrate the behavioral changes that result from the interventions and alleviate some concerns that we might be causing unnecessary harm to the agents, we apply an explainable surrogate model to evaluate the piece-specific values. In Figure 4, we can see how not learning about these different piece-specific advantages affects the value it places on the pieces. For example, the agent that did not learn how valuable a bishop pair advantage is will underestimate the value of the second bishop (compared to the other agents). Conversely, the agent that did not learn the value of the knight pair advantage overestimates the value of the second knight. Furthermore, not learning about having a queen advantage results in the queen's value being approximately half a pawn less.

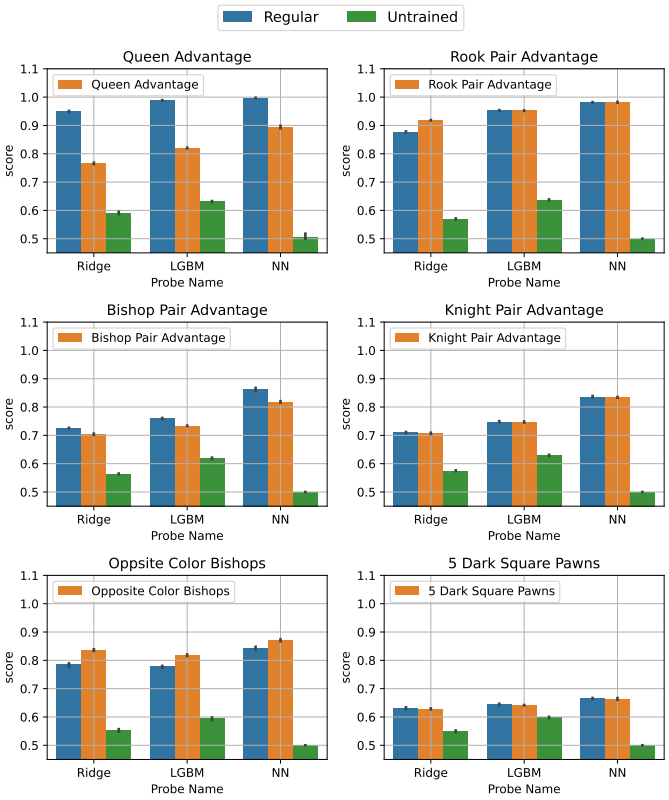


Figure 5. The probing accuracy of the regular model as well as models trained without seeing a concept.

4.4 Concept Importance and Probing Accuracy

In this section, we will interpret the concept importance, how it correlates to probing accuracy, and what kind of recommendation we can give about interpreting the results from such experiments.

Figure 5 shows different probing results. We probe i) the Regular model, f_{reg}, trained on the whole dataset, ii) the restricted models, f_{res,c_i}, with concept c_i removed from the training set, and iii) an untrained model, f_{unt} (with weights uniformly initialized). For all models, we probe with the three probing architectures. Furthermore, in Figure 6, we observe the relationship between the probing accuracy (in Layer 2) and the importance of the concept, evaluated in Elo points.

We observe a moderate correlation between the probing accuracy and the importance of the concepts, as also emphasized in Table 5, which we will analyze later in this section. The most important concept, *Queen Advantage*, has the highest probing accuracy, and the unimportant one, *5 Dark Squared Pawns*, has the lowest, with the other concepts falling in between, which is reassuring. Nonetheless, we also see that concepts that receive near identical probing accuracy may differ significantly in importance, such as the concepts *Bishop Pair Advantage* and *Knight Pair Advantage*, which indicates one has to take all probing results with some grain of salt. According to the literature, the concepts operate very differently, and actually, the bishop pair is considered more important than the knight pair.

In Figure 5, we compare the probing accuracy of the regular model versus a model trained on data with the concept in question removed from the training data. For the most important concepts, *Queen Advantage* and *Bishop Pair Advantage*, there is a notable drop in prob-

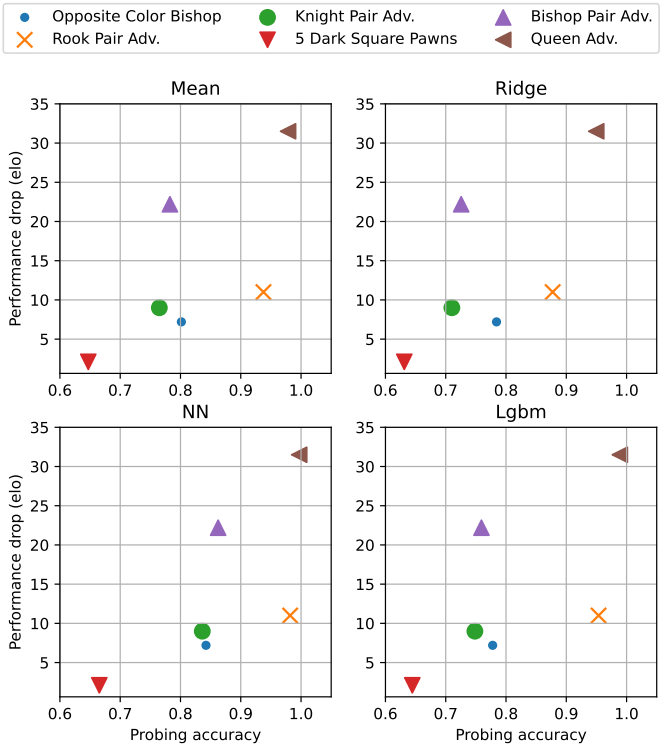


Figure 6. The relationship between probing accuracy and the importance measured in Elo points for all probes, as well as the mean value of all three probes.

Table 5. Pearson correlation between concept probing results and importance.

Probe, j	$P_j(f_{reg})$	$P_j(f_{reg}) - P_j(f_{res,c_i})$	$P_j(f_{reg}) - P_j(f_{unt})$
Ridge	0.69	0.83	0.64
LGBM	0.68	0.87	0.67
NN	**0.73**	**0.92**	**0.72**

ing accuracy; in contrast, the accuracy remains similar for the less important concepts.

To see what concept probing results mostly correlate with the importance. We analyze the Pearson correlation between the importance and i) $P_j(f_{reg})$, ii) $P_j(f_{reg}) - P_j(f_{res,c_i})$ and iii) $P_j(f_{reg}) - P_j(f_{unt})$. Where $P_j(f)$ is the probing accuracy of probe P_j applied to model f. The results can be seen in Table 5, from which we can draw two conclusions. First, when analyzing the probing results of the Regular model, $P_j(f_{reg})$, the neural network probe has the highest correlation with importance. Second, the change in probing accuracy after removing a concept from the training set, $P_j(f_{reg}) - P_j(f_{res,c_i})$, offers an even higher correlation with importance where the neural network probe also performs best.

To check to what extent we can predict the importance using the probing results, and provide a bit more context to previous results, we train a linear model to predict the importance and report the mean absolute error using leave-one-out cross-validation. The results are shown in Table 6 where we see again that the neural network probe is performing the best, and using the difference in probing accuracy between the Regular and Restricted models results in the highest accuracy, with a mean absolute error of 3.68 Elo points.

Table 6. The mean absolute error in Elo points, using a leave-one-out cross validation, of a linear model predicting the importance, given the concept probing results.

Probe, j	$P_j(f_{reg})$	$P_j(f_{reg}) - P_j(f_{res,c_i})$	$P_j(f_{reg}) - P_j(f_{unt})$
Ridge	9.61	5.68	10.48
LGBM	9.77	5.83	10.15
NN	**8.84**	**3.68**	**8.90**

5 Conclusion

In this work, we cast some light on how effective and reliable concept probing is, using a contrived autoencoder and a state-of-the-art game-playing agent as our test domain. The main takeaways are:

- The widespread practice of using linear probes and interpreting their accuracy to indicate concept importance gives some, albeit limited, insights into concepts learned by the network. The correlation between probes' accuracy and feature importance is only moderately correlated. Additionally, interpreting concept importance based on probe accuracy alone is unreliable. For example, we saw a concrete example of concepts indistinguishable that way — possessing the Bishop or Knight pair, the former highly beneficial and the latter not (as judged by both our self-play experiments and established chess-domain knowledge).
- However, a far better correlation is achievable using a more complex probe (neural network) and reading into the results using amnesic-like probing techniques. Doing this, where applicable, is recommended; however, we acknowledge that this may not always be possible (e.g., when unable to retrain the model).
- Using an untrained model as a baseline for the probing (e.g., as suggested in [5, 33]) did not prove helpful in our domain.

We see this work as valuable input into the ongoing discussion of the pros and cons of different probing architectures, especially in the game-playing domain. Furthermore, it provides some guidance for best practices in interpreting probing results.

As for future work, we plan to address some of the limitations of this study. First, we use only a handful of chess concepts, and adding more would improve confidence in the correlation metric. Second, we use the effects on playing strength by omitting a concept from model training as an importance metric. Albeit useful, as demonstrated, this is not necessarily the entire truth; we might be causing unintended harm to the models by removing other valuable correlated concepts. However, we took care when defining the concepts to avoid that and examined the models for unexpected behavior, such as the experiment in Figure 4.

Finally, retraining a model from scratch while holding out some concepts is not always feasible cost-wise or even impossible; thus, developing more practical ways to restrict a model without having to retrain from scratch might be worthwhile.

References

[1] Y. Adi, E. Kermany, Y. Belinkov, O. Lavi, and Y. Goldberg. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=BJh6Ztuxl.

[2] G. Alain and Y. 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. URL https://openreview.net/forum?id=HJ4-rAVtl.

[3] Y. Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

[4] F. Bodria, F. Giannotti, R. Guidotti, F. Naretto, D. Pedreschi, and S. Rinzivillo. Benchmarking and survey of explanation methods for black box models. *CoRR*, abs/2102.13076, 2021. URL https://arxiv.org/abs/2102.13076.

[5] A. Conneau, G. Kruszewski, G. Lample, L. Barrault, and M. Baroni. What you can cram into a single \$&!#* vector: Probing sentence embeddings for linguistic properties. In I. Gurevych and Y. Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2126–2136. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-1198. URL https://aclanthology.org/P18-1198/.

[6] Y. Elazar, S. Ravfogel, A. Jacovi, and Y. Goldberg. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Trans. Assoc. Comput. Linguistics*, 9:160–175, 2021. doi: 10.1162/tacl_a_00359. URL https://doi.org/10.1162/tacl_a_00359.

[7] D. Hupkes, S. Veldhoen, and W. H. Zuidema. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *J. Artif. Intell. Res.*, 61:907–926, 2018. doi: 10.1613/jair.1.11196. URL https://doi.org/10.1613/jair.1.11196.

[8] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3146–3154, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html.

[9] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, and R. Sayres. Interpretability beyond feature attribution: Quantitative Testing with Concept Activation Vectors (TCAV). In *35th International Conference on Machine Learning, ICML 2018*, volume 6, pages 4186–4195, 2018. ISBN 9781510867963.

[10] P. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim. The (un)reliability of saliency methods. In W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K. Müller, editors, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *Lecture Notes in Computer Science*, pages 267–280. Springer, 2019. doi: 10.1007/978-3-030-28954-6_14. URL https://doi.org/10.1007/978-3-030-28954-6_14.

[11] A. Kumar, C. Tan, and A. Sharma. Probing classifiers are unreliable for concept removal and detection. *Advances in Neural Information Processing Systems*, 35:17994–18008, 2022.

[12] Lichess. Lichess data set. https://database.lichess.org/, 2023. Accessed: 2023-08-13.

[13] N. F. Liu, M. Gardner, Y. Belinkov, M. E. Peters, and N. A. Smith. Linguistic knowledge and transferability of contextual representations. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 1073–1094. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1112. URL https://doi.org/10.18653/v1/n19-1112.

[14] C. Lovering, J. Forde, G. Konidaris, E. Pavlick, and M. 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. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/a705747417d32ebf1916169e1a442274-Paper-Conference.pdf.

[15] S. M. Lundberg and S. I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 4766–4775, 2017. URL https://github.com/slundberg/shap.

[16] R. H. Maudslay, J. Valvoda, T. Pimentel, A. Williams, and R. Cotterell. A tale of a probe and a parser. In D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7389–7395. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.659. URL https://doi.org/10.18653/v1/2020.acl-main.659.

[17] T. McGrath, A. Kapishnikov, N. Tomasev, A. Pearce, D. Hassabis, B. Kim, U. Paquet, and V. Kramnik. Acquisition of chess knowledge in alphazero. *CoRR*, abs/2111.09259, 2021. URL https://arxiv.org/abs/2111.09259.

[18] T. McGrath, A. Kapishnikov, N. Tomašev, A. Pearce, M. Wattenberg, D. Hassabis, B. Kim, U. Paquet, and V. Kramnik. Acquisition of chess knowledge in alphazero. *Proceedings of the National Academy of Sciences*, 119(47):e2206625119, 2022.

[19] J. X. Mi, A. D. Li, and L. F. Zhou. Review study of interpretation methods for future interpretable machine learning. *IEEE Access*, 8: 191969–191985, 2020. ISSN 21693536. doi: 10.1109/ACCESS.2020.3032756.

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

[21] A. Pálsson and Y. Björnsson. Unveiling concepts learned by a world-class chess-playing agent. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023*. ijcai.org, 2023.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[23] T. Pimentel, J. Valvoda, R. H. Maudslay, R. Zmigrod, A. Williams, and R. Cotterell. Information-theoretic probing for linguistic structure. In D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4609–4622. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.420. URL https://doi.org/10.18653/v1/2020.acl-main.420.

[24] N. Puri, S. Verma, P. Gupta, D. Kayastha, S. Deshmukh, B. Krishnamurthy, and S. 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. URL https://openreview.net/forum?id=SJgzLkBKPB.

[25] S. Ravfogel, Y. Elazar, H. Gonen, M. Twiton, and Y. Goldberg. Null it out: Guarding protected attributes by iterative nullspace projection. In D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7237–7256. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.ACL-MAIN.647. URL https://doi.org/10.18653/v1/2020.acl-main.647.

[26] A. Ravichander, Y. Belinkov, and E. H. Hovy. Probing the probing paradigm: Does probing accuracy entail task relevance? In P. Merlo, J. Tiedemann, and R. Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 3363–3377. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.eacl-main.295. URL https://doi.org/10.18653/v1/2021.eacl-main.295.

[27] M. T. Ribeiro, S. Singh, and C. 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. doi: 10.1145/2939672.2939778.

[28] L. Schut, N. Tomasev, T. McGrath, D. Hassabis, U. Paquet, and B. Kim. Bridging the human-ai knowledge gap: Concept discovery and transfer in alphazero. *CoRR*, abs/2310.16410, 2023. doi: 10.48550/ARXIV.2310.16410. URL https://doi.org/10.48550/arXiv.2310.16410.

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

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

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

[32] N. Tomlin, A. He, and D. Klein. Understanding game-playing agents with natural language annotations. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 797–807. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.ACL-SHORT.90. URL https://doi.org/10.18653/v1/2022.acl-short.90.

[33] K. W. Zhang and S. R. Bowman. Language modeling teaches you more syntax than translation does: Lessons learned through auxiliary task analysis. *CoRR*, abs/1809.10040, 2018. URL http://arxiv.org/abs/1809.10040.

Appendix D

Searching For Explainable Solutions in Sudoku

Searching for Explainable Solutions in Sudoku

Yngvi Björnsson
Department of Computer Science
Reykjavik University
Reykjavik, Iceland
yngvi@ru.is

Sigurður Helgason
Department of Computer Science
Reykjavik University
Reykjavik, Iceland
sigurdurhel15@ru.is

Aðalsteinn Pálsson
Department of Computer Science
Reykjavik University
Reykjavik, Iceland
adalsteinn19@ru.is

***Abstract*—Explainable AI is an emerging field that studies how to explain the rationality behind the decisions of intelligent computer-based systems in human-understandable terms. The research-focus so far has though almost exclusively been on model interpretability, in particular, on trying to explain the learned concepts of (deep) neural networks. However, for many tasks, constraint- or heuristic-based search is also an integral part of the decision-making process of intelligent systems, for example, in planning and game-playing agents. This paper explores how to alter the search-based reasoning process used in such agents to generate more easily human-explainable solutions, using the domain of Sudoku puzzles as our test-bed. We model the perceived human mental effort of using different familiar Sudoku solving techniques. Based on that, we show how to find an explanation understandable to human players of varying expert levels, and evaluate the algorithm empirically on a wide range of puzzles of different difficulty.**

***Index Terms*—XAI, heuristic search, Sudoku**

I. Introduction

The field of explainable AI (XAI) is concerned with developing techniques that are useful for explaining the reasons behind the decisions of intelligent computer systems, preferably in a way easily understandable to humans. The research focus of XAI has so far mostly been on model interpretability, in particular, providing insights into concepts learned by (deep) neural networks. Given how prevailing such machine-learning techniques have become, this is unsurprising. However, for many tasks, heuristic search is also an integral part of the decision-making process of intelligent systems.

Heuristic search is one of the fundamental problem-solving techniques in AI and computer science. It provides computer-based agents the means of reasoning, or "thinking ahead." One of the main appeals of the approach is its general applicability to decision making in a wide range of disparate problem domains, including manufacturing optimization, automated scheduling and planning, and game-playing. Such informed search techniques are often capable of producing high-quality (real-time) answers to complex decision problems when provided with proper guidance.

Although current research into model interpretability may be applicable for explaining learned heuristic models used by the search, the better part of the heuristic-search reasoning process remains unexplained. The solution to heuristic-search problems typically takes the form of a sequence of actions. For example, a human to fully appreciate the solution might need to know not only how each step came about, but also why that particular sequence is preferable to an alternative one.

In this paper, we take initial steps into making the reasoning process of heuristic search more transparent to humans, using Sudoku puzzles as our test-bed. There are fundamentally two different approaches to achieve this. The former is to treat the solver as a black-box and then try to explain the resulting solution post-mortem; in contrast, the latter alters the solver to produce more human-understandable solutions. Both approaches have their pros and cons, but the latter typically offers greater flexibility in delivering human-explainable solutions, although possibly at the cost of solution quality or run-time performance. We opt for the second approach here as the goal is first and foremost to generate highly informative solutions for humans instead of, e.g., run-time performance. The task of the search process has thus become more intricate: it is looking for not only an acceptable solution to the problem (quality-wise) but also an easily explainable one. Such a solving approach is relevant in settings where a human is expected to verify or execute the resulting solution as well as in settings where a human is hoping to learn from the solving process.

The primary contributions of the paper are: *(i)* we present a heuristic-search-based Sudoku solver that returns solutions understandable to humans with different levels of expertise; *(ii)* a study of numerous problem characteristics that relate to the solution difficulty in our test-bed domain is provided; *(iii)* although the test-bed here is Sudoku, many of the underlying ideas are transferable to a wider-range of problem domains with similar characteristics; *(iv)* and finally, this is one of few recent works emphasizing explainability in heuristic search, and issue that hopefully will attract added research attention.

The paper is organized as follows. Section II introduces both the necessary terminology and preliminaries. Section III explains Sudoku and the basic strategies humans use for solving such puzzles. Related to that, in Section IV, we develop a metric for (objectively) measuring the perceived difficulty humans have in understanding solutions using the human-like strategies. In Section V, we introduce our algorithm for finding the most instructive solution paths, followed by an empirical evaluation of the algorithm in Section VI. Finally, we discuss related work in Section VII and conclude and discuss future work in Section VIII, respectively.

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

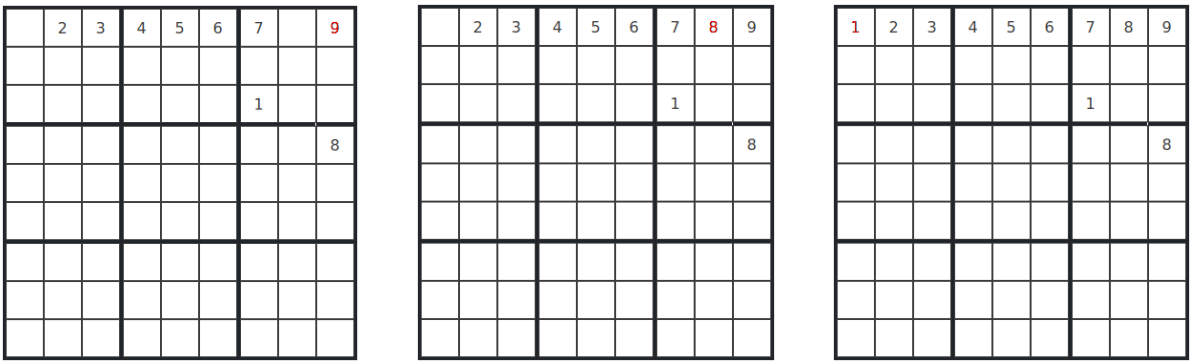


Figure 1. Sole candidates: In the topmost row: 9 is a sole candidate (left), 8 becomes a sole candidate (middle), and finally the 1 becomes one too (right).

II. Background

Heuristic (or *informed*) search methods encompass a wide range of search algorithms. The term is commonly used to refer to state-space-based search methods that rely on heuristics for guiding a search process, such as *A**, *Minimax*, *MCTS*, and backtracking-based *CSP* search. The state-space is typically explored by growing a search tree representing the different possible continuations to take. The problem domains can be single- or multi-agent based, deterministic or non-deterministic, fully observable or not, and the heuristics largely domain-specific (hand-crafted or learned) or largely domain-independent (like variable and action selections in CSP). For example, classic heuristic-search-based planning typically employs single-agent search in deterministic and fully observable domains.

Different search domains face dissimilar explainability challenges. In an adversary-search domain like chess, a human observer might need to realize why a seemingly promising continuation is not taken, for example, because of an unanticipated refutation, whereas in other domains, like Sudoku, the human must first and foremost understand each step in the proposed solving sequence.

Constraint satisfaction problems (CSPs) are a class of problems that can be described by a set of variables, $V = \{v_1, v_2, ...v_i\}$, a set of domains $D = \{d_1, d_2, ..., d_i\}$, where d_j is the set of values that variable v_j can take, and lastly a set of binary constraints $C = \{c_1, c_2, ..., c_n\}$, where $c_j = (v_a, v_b, relationaloperator)$.

A variable v_i is *arc-consistent* with another variable v_j if (and only if), for every value a in d_i there exists a value b in d_j such that (a, b) satisfies the binary constraint between the two variables. A problem is arc-consistent if (and only if) every variable is arc-consistent with every other variable. Although there exist several related algorithms for making a problem arc-consistent, AC-3 [11] is the most widely used.

Sudoku, our test-bed puzzle game, is naturally formalized as a CSP, and Sudoku puzzles are generally effectively solvable using a CSP solver. However, when searching not only for a solution but the most explainable one, then the problem is more naturally formalized as a heuristic-search-based single-agent problem. One distinguishing feature between CSP and heuristic search is that, in the former, the goal state itself represents the solution, whereas in the latter the exact sequence of actions taken that lead to the goal state is the solution.

III. Sudoku

The game of Sudoku is played on a rectangular grid, usually at order 3, meaning that the grid is of size $3^2 \times 3^2 = 9 \times 9$, and overlaid by mutually exclusive boxes (also called blocks or regions) of size 3×3 (see Figure 1). We interchangeably refer to grid cells as variables, as this is how they would be represented in the CSP formalism. The domain of each variable (grid-cell) is 1 to 9 (3^2).

Initially, a partially filled puzzle is provided and the task of the player is to fill out the remaining grid cells such that each number occurs exactly once in each row, column, and box. Sudoku puzzles are traditionally generated such that they have exactly one solution. The puzzles can be of a varied level of difficulty, ranging from being easily solved by humans using only trivial deduction techniques to requiring quite sophisticated levels of reasoning.

A. Human Solving Strategies

Humans do apply various deduction schemes to solve Sudoku puzzles and a rich literature exists (e.g., see [4]) describing a wide array of strategies of different complexity, including with names such as "X-wing", "Swordfish" and "Unique Rectangle". Novice players do typically rely only on a couple of simple deduction strategies, however, as they become more proficient the more elaborate strategies they incorporate into their arsenal of solving techniques. Consequently, when presenting a puzzle solution to a novice player, the solution must be at an appropriate sophistication level for the player to fully comprehend it, for example, by using only familiar deduction strategies. As the players develop their problem-solving skills, more advanced strategies can be incorporated.

We will borrow from the Sudoku literature some of the deduction strategies that are typical for beginner (to intermediate) level players, as presented in the following subsections. We refer to the (other) cells in the same row, column, or box as a cell c as *row-*, *column-*, and *box-neighbours* of cell c,

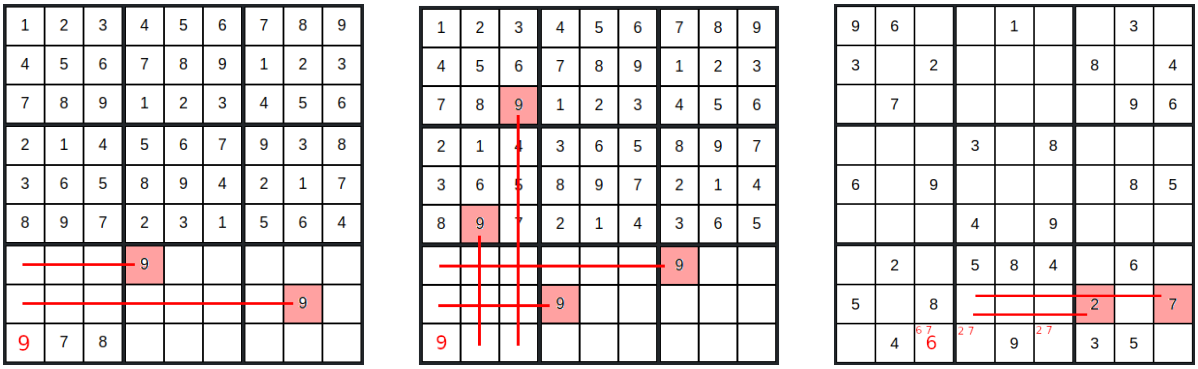


Figure 2. One-dimensional (left) and two-dimensional (middle) unique candidate, and naked-double candidate (right)

respectively. The *neighbours* of a cell c is the union of the row-, column- and box neighbours.

B. Sole Candidate

The *sole candidate strategy* detects a grid cell whose domain has been reduced to a single candidate, for example, as seen in Figure 1. More generally, in detecting a sole candidate all neighbours of a cell c may play a role in restricting the cell's domain. Note, that in the CSP framework sole-candidate domain reduction would be performed via arc-consistency.

C. Unique Candidates

If a value, say v, has been eliminated from the domains of all row-, column- or box-neighbours of a cell c, then we know that cell c must be assigned the value v. The *unique candidate strategy* tries to do such an elimination, for example, as depicted in Figure 2.

We remove values from the box cell's domain, as in Figure 2, by scanning its neighbours. The unique candidate can be found with two levels of difficulty, they can be found by scanning ether one or two dimensions. We refer to candidates found by scanning two dimensions as *two-dimensional unique candidates*, and those found by scanning only one of the two dimensions, we refer to them as *one-dimensional unique candidates*. The reason for this distinction is that it typically requires somewhat less mental effort to scan only one type of neighbours. Analogously, when finding a unique candidate in a row (or a column), we use the box-neighbours and the col-neighbours (or row-neighbours) of the remaining row (or col) cells. All one-dimensional unique candidates are also two-dimensional unique candidates, but not the reverse.

D. Naked Doubles Candidate

Naked doubles is a pair of neighbour cells, say c_1 and c_2, having an identical domain D of size two. This means that each of two values remaining in D must be assigned to one of the two cells. Although, we do not know (yet) which cell should be assigned which value, we know that the two domain values must be reserved for those two cells only. This implies that if c_1 and c_2 are, for example, row-neighbours, then no other cell in that row can take on the two values in D and we can remove the values from their respective domains if present. Analogous arguments apply if c_1 and c_2 are col- or box-neighbours, as seen in Figure 2.

IV. Estimating the Cost of an Explanation

How understandable is a solution to a human player? Ultimately, the answer to this is subjective and depends on the person. Here we try to quantify explainability in somewhat objective terms based on different solution strategies and general principles. Assuming different strategies exist for achieving individual steps of a multi-step plan, for example, the different strategies for deducting an assignment of a Sudoku grid-cell we saw in the previous section.

We define a cost function for measuring a plan's (solution's) explainability — the higher the cost the less understandable the plan is — where we assume the following general principles:

(i) Different strategies may impose different non-negative step costs.
(ii) The same strategy may impose a different step cost depending on the context it is used in.
(iii) Repeatedly applying the same strategy yields a lowered step cost at each successive step (other things being equal). Switching between strategies may involve additional costs.
(iv) Using a strategy not previously used in the solving process may impose an additional (one-time) cost. Thus, using a smaller set of disparate strategies is always preferable (other things being equal).
(v) The total cost of a plan (solution) is the summed cost of its steps.

More formally, taking the above principles into account, we specify our Sudoku solution cost metric as:

$$W_{total} = \sum_{i=1}^{N} \left(\left(1 + \left(1 - \frac{a_{m,i}}{U_i} \right) \right)^{\delta} \theta^{n_{m,i}} w_{m,i} + \xi_{m,i} w_{m,i} \right) \tag{1}$$

where m indicates the strategy used, w_m is the cost of the strategy used (see Table I), N is the number of unassigned variables at the beginning of the puzzle, U_i is the number

Table I
STRATEGY SPECIFIC COST, w_m

	w_m
One-Dimensional Unique Candidate	1
Sole Candidate	3
Two-Dimensional Unique Candidate	5
Naked Double Candidate	10

of unassigned variables at step i, $a_{m,i}$ is the number of available moves for the chosen strategy, δ is the power of the transformation function, θ is a cost decaying factor, $n_{m,i}$ indicates the count of similar moves done before our move m and finally $\xi_{m,i}$ indicates the cost of learning a strategy.

In the first term of our equation, $\left(1+\left(1-\frac{a_{m,i}}{U_i}\right)\right)^{\delta}$, we capture the context specific cost (see (ii)) for strategy m at step i, indicating a higher cost with lower availability. The strategy specific cost (see (i)) is lowered by repetitive use of the same strategy through the decaying factor $\theta^{n_{m,i}}$ (see (iii)). The value for the learning factor (see (iv)) $\xi_{m,i}$ was arbitrarily set to 10 if we are using strategy m for the first time at step i, otherwise it is 0.

One can think of the strategy-specific cost in Table I as approximate of the different techniques' sophistication level: novice players easily apply the less costly ones, whereas the more costly require added skill. Solutions, i.e., action sequences, comprised chiefly of simple strategies, are understandable even to less skilled players. In contrast, more advanced players would also comprehend a solution that uses more sophisticated steps. The exact costs listed in the table are per se inconsequential; instead, it is worth noting that different strategies have different associate costs, which may vary depending on the targeted user. For example, given knowledge of the expert level of a user observing the solution, unfamiliar strategies are made costly (even infinite) whereas ones familiar to the user are cheap, thus ensuring that the generated solution uses only strategies familiar to the user.

In this framework, when given a solution, it is naturally explained as the sequence of actions taken annotated with the strategy used and the cells involved. For example, in Figure 1, the explanations would be along the lines of: *9 added as a sole-candidate (using all neighbor-constraints), 8 added as a sole-candidate (using row- and box-constraints), and 1 added as a sole neighbor (using a row-constraint)*.

It is important to note that our solver is not dependent on using this exact cost-function formulation for evaluating its solution paths. In theory, any metric will do that is monotonically non-decreasing with an added number of actions. In practice, however, we would like to capture the idea of varying mental-efforts required for the human to discover and/or understand different solutions, and thus we will model the cost on the above-mentioned principles.

V. METHODS

A backtracking-based CSP solver can be used to solve Sudoku puzzles efficiently. However, here the task is not only to solve the puzzles but to do so in a manner explainable to humans with different levels of expertise. One approach would be to use a standard solver and then try to explain the outcome post-mortem. Unfortunately, this is not guaranteed to succeed as some of the steps taken in solving the puzzle might be too intricate given the ability of the human observer, possibly even based on multiple trial and error as opposed to human-like deduction strategies. The solver must instead be biased towards finding the simplest (best explainable) solution while solving the puzzle using only human-like deduction strategies. Here we present such a solver.

A. Strategy Abstraction

Our approach selects a (variable, value) pair iff that assignment can be deducted using one of the previously mentioned human-like Sudoku solving strategies. Although we lose the ability to solve Sudoku boards that cannot be solved using only those strategies, that is inconsequential as such solutions would be non-explainable. Furthermore, once a strategy has been chosen, say a sole candidate, that strategy is applied repeatedly before potentially switching to a new strategy. A parameter *batch size* controls the number of repetitions.

One can think of a chosen strategy and batch size as a *macro-action*, that is, instead of the action filling out a single grid cell, it fills out multiple cells at once. Not only does such an approach imitate how humans solve the puzzles (as supported by both the Sudoku literature, e.g. [17], and results from a general human-style solving model [13]), but it also reduces the search space significantly (roughly $s^{m/b}$ where s is the amount of Sudoku strategies and m is the amount of empty cells, and b is the batch size). However, using a batch size larger than one does potentially sacrifice optimality; that is, the algorithm might overlook a lowest-cost solution (according to our metric). For example, when exploring a higher-cost action, the algorithm is forced to repeatedly apply that high-cost action, even if lesser-cost actions become available as an aftereffect of a previous application of the high-cost action.

B. The Search Algorithm

The pseudo-code in Algorithm 1 describes the search, where each recursive call to the search function does the following: (1) for each available strategy we apply (up to) *batch size* many moves (not transitively considering those that arise from using the strategy); (2) the cost of the (partial) solution is computed using our cost function; (3) we recurse into the search once more with the changes from applying that strategy; (5) we undo everything from applying the strategy and try the next.

The search is depth-first branch-and-bound-like, keeping track of the least costly solutions found so far and pruning where applicable. It also uses a transposition table, for avoiding reexploring the same state if reached via two different paths, and macro-actions to reduce the search space. A significant characteristic is that the actions allowed for the algorithm are always explainable in terms of Sudoku strategies. Given that the algorithm has at its disposal sufficiently advanced

Algorithm 1 Searching for a least-cost path

```
map tt {transposition_table}
f_best = ∞
function Search(s, g, strategies)
if Terminal(s) then
    {Keep track of best solution so far}
    if g < f_best then
        f_best = g
    return 0
if g ≥ f_best then
    {A more costly path, no need to explore further}
    return ∞
if s in tt then
    {h* already found for s}
    if g + tt(s) < f_best then
        f_best = g + tt(s)
    return tt(s)
subtree_costs = []
for all strategy in strategies do
    {Get batch many moves for a given strategy}
    actions = GetActions(strategy)
    ApplyActions(s, actions)
    c = CostFunction(s, actions)
    h = Search(s, g + c, strategies)
    UndoActions(s, actions)
    subtree_costs.append(c + h)
tt(s) = h* = min(subtree_costs)
return h*
```

Table II
LEAST COSTLY PATH FOUND TO SOLVE THE BOARD IN FIGURE 3. COST 33

Available Moves				Chosen Moves		
sole	1d unique	2d unique	naked	Strategy	Value	Square
3	2	5	1	1d unique	5	I1
4	3	6	1	1d unique	5	E3
4	3	7	1	1d unique	8	E1
4	3	6	1	1d unique	2	H1
4	2	6	0	1d unique	8	H3
3	1	5	0	1d unique	3	H7
3	2	4	0	1d unique	3	E8
3	2	3	0	1d unique	2	E7
2	2	2	0	1d unique	2	I8
1	1	1	0	1d unique	7	I7

Table III
MOST COSTLY PATH FOUND TO SOLVE THE BOARD IN FIGURE 3. COST 342

Available Moves				Chosen Moves		
sole	1d unique	2d unique	naked	Strategy	Value	Square
3	2	5	1	naked	7	I7
2	2	4	0	sole	8	H3
3	3	5	1	naked	5	E3
3	3	4	1	naked	8	E1
2	2	3	0	2d unique	5	I1
2	2	4	0	sole	2	H1
2	2	4	0	sole	3	H7
2	2	3	0	1d unique	3	E8
2	2	2	0	2d unique	2	E7
1	1	1	0	2d unique	2	I8

strategies to progress in each step, it is complete (i.e., always finds a solution).

VI. RESULTS

Here we provide an empirical evaluation of our solver, both in terms of solution understandability and run-time performance. We also look at search-space properties of interest.

A. Experimental Setup

To evaluate our solver, we used an online tool *qqwing* [12] to generate Sudoku puzzles of varying difficulty (simple, easy, and intermediate), the difference being the sophistication level of the strategies required to solve the puzzles. We gathered 50 puzzles for each of the three chosen difficulty levels. All experiments were run on a AMD Ryzen 5950 CPU.

B. Solution Understandability

We start with an example, depicted in Figure 3. The Sudoku board is close to completion with multiple ways to continue. Tables II and III show two such alternative continuations: the former — proposed by our algorithm — uses only a single relatively straightforward strategy, whereas the latter alternates between several strategies of varied sophistication levels, which our algorithm estimates to be the most costly according to our cost function.

Which continuation is more logical? Although there is no definite answer, then the former is easier to comprehend and can be explained to humans of even novice solving abilities (as it requires only one relatively straightforward strategy). There are two one-dimensional unique candidates available to start with, I1 and H7[1]. Our solver proposes to fill I1, then E3 becomes a new one-dimensional unique candidate, and so on until the puzzle is solved. The solution shown in the latter table is much less approachable, requiring jumping between four different strategies and is possibly even incomprehensible to most novice human solvers. On the contrary, expert human solvers might find such an explanation preferable.

[1]We use a board coordinate system where the rows are indexed top-to-bottom by the letters A–I, and the columns left-to-right by digits 1–9.

3	5	7	4	2	1	9	8	6
4	2	1	8	9	6	5	7	3
6	8	9	7	3	5	4	1	2
1	9	3	2	8	7	6	4	5
	6		9	1	4			7
7	4	2	6	5	3	1	9	8
9	7	6	3	4	2	8	5	1
	1		5	7	9		6	4
	3	4	1	6	8			9

Figure 3. Unsolved board with 10 remaining moves

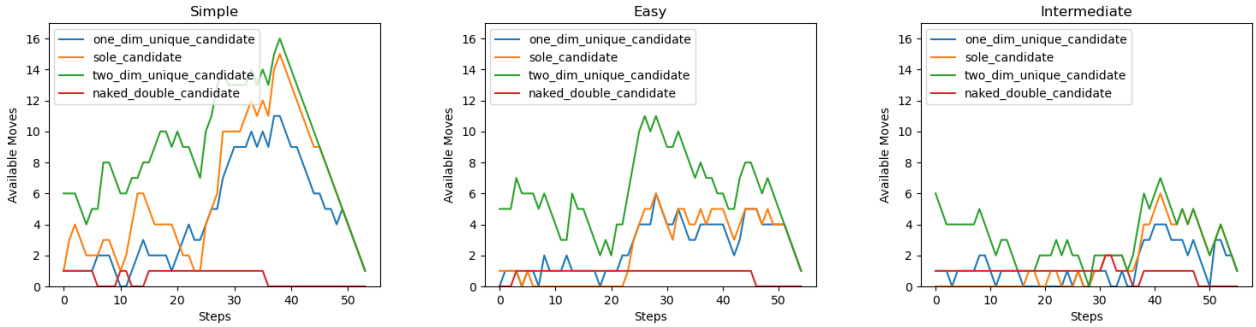


Figure 4. Number of available moves for the least-cost path for three puzzles with different move-availability: simple (left), easy (middle), intermediate (right)

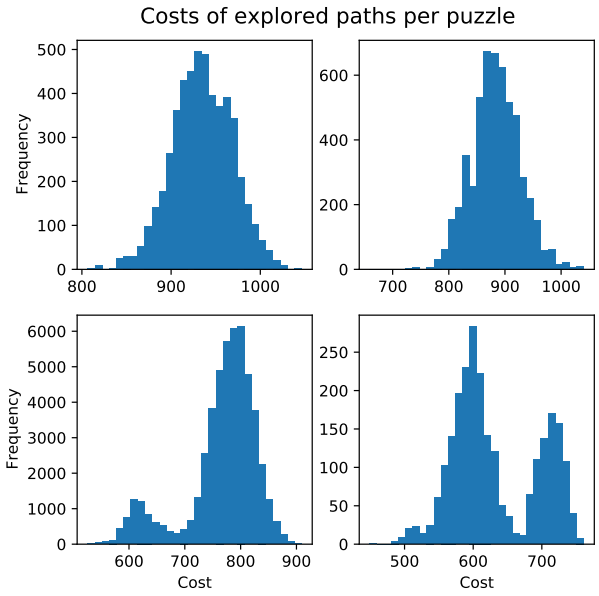


Figure 5. Costs of the various paths explored during the search, for four puzzles.

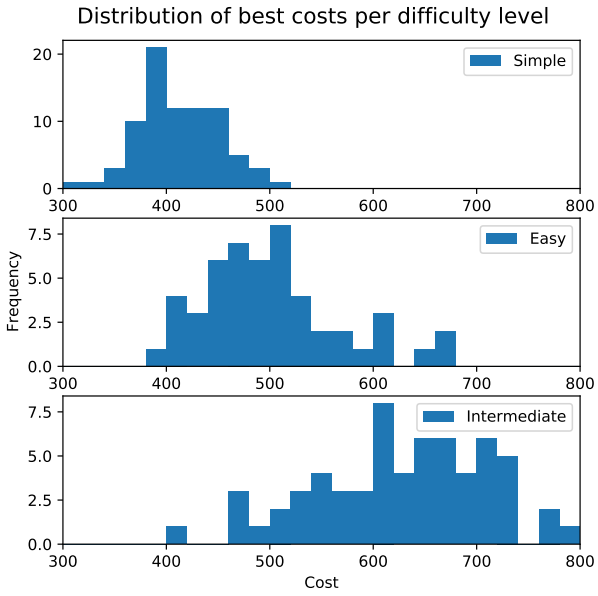


Figure 6. Resulting distributions of the best costs found from solving 50 tables of each difficulty levels: Simple(top), Easy(middle), and Intermediate(bottom)

C. Solution Cost Analysis

The understandability of the possible solutions to a puzzle may vary greatly, as we saw. Figures 4 and 5 cast some more light on this. The former shows the availability of different strategies as the search progresses (the board fills up), whereas the latter shows the cost distributions of all possible solutions to four different puzzle instances. The distributions take different shapes, although in all four the tails are slim, that is, there are only a few very low-cost and high-cost cases.

The availability of strategies (moves) varies throughout the search, this gives us insight into the difficulty of the given solution. A solution that has few available moves throughout the search can in general be considered harder than one with an abundance of possible continuations at each step. For the more simple puzzles the board is open, that is, there are multiple simple moves available, whereas for the intermediate board there is frequently only a few possible continuation. For example, we see in Figure 4 rightmost graph that at one point the only continuation is to use a Naked Double strategy, highlighting the difficulty of that puzzle.

D. Puzzle Difficulty vs. Cost Metric

The puzzles we used for our evaluation were all pre-labeled with a difficulty ranking of simple, easy, or intermediate. Figure 6 shows the distribution of best solution costs, as measured by our cost metric, over all puzzles, and how it relates to the pre-labeled difficulty ranking. The puzzles in the more difficulty-ranked puzzle categories have higher solution costs in general, although there is some overlap. Table IV gives the mean- and median-cost for each category.

Table IV
COST OF DIFFICULTY CATEGORIES

Difficulty	mean	median	std
Simple	411	402	38
Easy	501	496	66
Intermediate	630	632	86

Table V
OVERVIEW OF STRATEGIES BEING USED IN LOWEST AND HIGHEST BEST COSTS FOUND FOR GIVEN DIFFICULTY

	Lowest Best Cost			Highest Best Cost		
	Simple	Easy	Inter	Simple	Easy	Inter
1d unique	17	12	19	13	18	11
sole	15	36	19	11	8	11
2d unique	0	0	0	15	1	14
naked	0	0	0	0	0	1
Cost	319	384	400	500	671	806

To illustrate what the variability in the different costs in Figure 6 might indicate, we show in Table V the solutions statistics for the highest and lowest costs for each difficulty level. The solutions with a lower cost are able to solve the puzzle by using fewer strategies and using the simpler strategies relatively ofter.

E. Run-Time Analysis

One of the configuration parameters to the algorithm is the batch size, that is, the maximum number of actions to perform of a single strategy before considering switching to a different one. A batch size of one indicates that one can freely switch between strategies (as in regular search), a batch size of two makes the algorithm apply that strategy twice (as a single macro-action), if available, before trying the next one, etc. A batch size of infinity means that all available application of a given strategy are played as a single macro-action. The main purpose of the batched actions is both to improve the algorithm's run-time efficiently, but also to have it mimic better how humans typically solve the puzzles (repeatedly apply the same strategy while available). Figure 7 shows the average run-time of the algorithm using different batch sizes. The general trend is that the run-time decreases with increased batch size, however, there is an interesting anomaly when going from batch size one to two, which increases the run-time. [2]

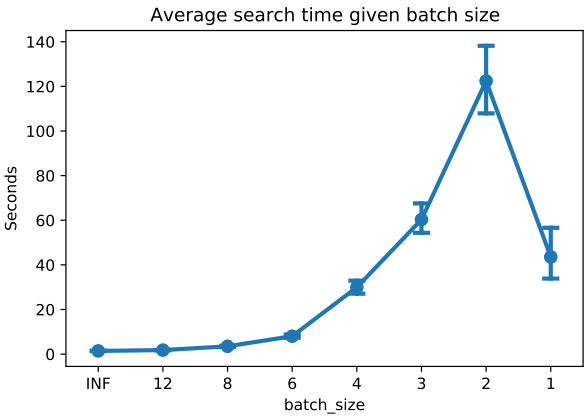


Figure 7. Time / batch size runtime

VII. RELATED WORK

There exists an extensive research literature on model interpretability (e.g., see [8] for a survey) whereas research into the explainability of (heuristic) search is still in its infancy.

In the context of intelligent autonomous agents, particularly in planning [2], the ability of an agent to explain the reasoning behind its decisions has been labeled *explainable agency* [10], and which requires four distinct abilities: (i) the agent must be able to explain decisions made during plan generation, (ii) report which actions it executed at different levels of abstraction, (iii) show how actual events diverged from planned ones and what adaptions were necessary, and, finally, (iv) communicate its decisions and reasoning effectively in a formalism natural to humans. Furthermore, work on explainable agency in systems based on heuristic search tends to distinguish between two types of self-explanations: *process* vs. *preference* oriented. The former emphasizes the (thought) process leading to finding the solutions, whereas the latter focuses on the solutions themselves without concerns about how they were found [10]. In this paper, we focused on the former.

The challenges arising in explaining planning systems are outlined in [5], and the gap between current planning algorithms and human problem-solving acknowledged. Some concrete preliminary steps towards plan explainability are also taken. Interactive (or human-in-the-loop) planning, where computer agents actively participate with humans in the plan generation, do use both some plan-recognition and human-understandable explanation capabilities [3]. Also, recent work on a fully autonomous planning system, which is capable of some rudimentary plan explanations, uses a greedy algorithm to generate so-called minimal plan explanations by searching over the space of abstract models. [16]

For CSPs, the main focus has been on deriving efficient constraint propagation techniques (see e.g. [14] for an overview), with little attention paid to how a solution is found. In many cases, this is perfectly acceptable as the goal is simply to find a good solution (performance-based approach), whereas in other cases —like the one we studied here— such an approach is not helpful because understanding the solution process is the main goal (process-based approach). A few notable exceptions falling into the latter category, include using domain-specific constraints to solve logic puzzles to find more human-like solutions [15], and using an user-understandable tree-hierarchy in an explanation-based constraint programming system [9].

There has also been some recent work in theorem proving systems in generating more human-style proofs for elementary

[2]We have investigated this further, including ruling out (the best we can) that this is caused by incorrect program behaviour. As of now we do not have an bullet-proof explanation for this somewhat unexpected behaviour.

mathematical problems [7].

In Sudoku, the MITS system [1] is an intelligent autonomous tutor that acts interactively with a student to complete a puzzle. It retains a model of acceptable next actions using a dynamic strategy graph that is continuously adapted throughout the game. Either the tutor or the student can direct how to play (mixed-initiative). Good Sudoku [6] is a commercially available Sudoku app. It uses a custom puzzle generator to generate puzzles with varying difficulty, e.g., requiring specific strategies. These puzzles are accompanied by an algorithm that runs in the background and provides hints while the player solves the puzzle.

VIII. Conclusions and Future Work

Many intelligent decision-making systems employ both models and search as an integral part of their decision-making process. The decisions made by such systems must ideally be transparent, verifiable, and, where applicable, instructive for humans. The research focus of XAI has so far mostly been limited to model interpretability, in particular, providing insights into concepts learned by (deep) neural networks.

In this work, we put the focus on the search part of the decision-making process. We use Sudoku as our test-bed and provide a solver — a hybrid of a heuristic- and constrained-based — that biases the search towards finding solutions that are easily explainable to humans with different levels of domain expertise. We provide a concrete method for achieving this and show that such a solver is feasible in our test domain. We also provide an analysis of various aspects of the search- and solution space to better reveal the challenges faced by the solver.

There are several avenues for taking this work further. In particular, it would be of interest to compare more quantitatively the solutions generated by our strategy-aware solver to those generated by a traditional constraint-based Sudoku solver. For example, how understandable are the solutions as judged by our perceived mental-cost metric or, more ideally, by human players. The model we currently use for estimating solution difficulty could also be further refined, for example, based on observing better how humans play. Finally, the domain of Sudoku is just our first stop on the journey into exploring explainability issues in constraint- and heuristic-based search.

The plan also is to generalize the proposed techniques to be more widely applicable. Coarsely speaking, the algorithm, as is, applies to other state-based problem domains where: (i) the search task is to find a solution, as opposed to finding an optimal one, and; (ii) a monotonically non-decreasing cost metric applies for evaluating the perceived understandability of the different solutions path, and (iii) there is a close correspondence between the understandability of individual actions in the solution path and how easily the solution can be explained as a whole (for example, we assume that explaining the solution is achievable by explaining each individual action in the solution independently). Sudoku conforms nicely to the above restrictions and, therefore, is ideal as a first domain for research into explaining heuristic-search processes. A natural next step is to move to a somewhat more challenging search-based problem domain where some of the above conditions are relaxed. For example, such an approach could potentially be useful in planning, in particular for tasks where some subsets of actions are preferable over others from a human-style solving point of view, even though all suffice to solve the planning task.

References

[1] Allan Caine and Robin Cohen. Tutoring an entire game with dynamic strategy graphs: The mixed-initiative sudoku tutor. *JCP*, 2(1):20–32, 2007.

[2] T. Chakraborti, A. Kulkarni, S. Sreedharan, D. E. Smith, and S. Kambhampat. Explicability? legibility? predictability? transparency? privacy? security? the emerging landscape of interpretable agent behavior. In *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)*, pages 86–96. AAAI Press, 2019.

[3] T. Chakraborti1, K. P. Fadnis, K. Talamadupula, M. Dholakia, B. Srivastava, Jeffrey O. Kephart, and R. K. E. Bellamy. Visualizations for an explainable planning agent. In *Proceedings of the Twenty-Eight International Conference on Automated Planning and Scheduling (ICAPS 2018)*, pages 5820–5822. AAAI Press, 2018.

[4] Conceptis. Conceptis Sudoku solving techniques, 2019. Accessed on 19-11-2019, https://www.conceptispuzzles.com/index.aspx?uri=puzzle/sudoku/techniques.

[5] M. Fox, D. Long, , and D. Magazzeni. Explainable planning. In *IJCAI 2017 Workshop on Explainable Artificial Intelligence (XAI)*, 2017.

[6] Zach Gage and Jack Schlesinger. Good Sudoku is software for generating and solving Sudoku puzzles, 2008. Accessed on 29-06-2021, www.playgoodsudoku.com.

[7] M. Ganesalingam and W. T. Gowers 0001. A fully automatic theorem prover with human-style output. *J. Autom. Reasoning*, 58(2):253–291, 2017.

[8] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv*, 51(5):93:1–93:42, 2019.

[9] Narendra Jussien and Samir Ouis. User-friendly explanations for constraint programming. In Anthony J. Kusalik, editor, *Proceedings of the Eleventh Workshop on Logic Programming Environments (WLPE'01), Paphos, Cyprus, December 1, 2001*, 2001.

[10] Pat Langley, Ben Meadows, Mohan Sridharan, and Dongkyu Choi. Explainable agency for intelligent autonomous systems. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4762–4764. AAAI Press, 2017.

[11] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[12] Stephen Ostermiller. QQwing is software for generating and solving Sudoku puzzles, 2005. Accessed on 19-11-2019, www.qqwing.com.

[13] Radek Pelánek. Difficulty rating of sudoku puzzles: An overview and evaluation. *CoRR*, abs/1403.7373, 2014.

[14] Francesca Rossi, Peter van Beek, and Toby Walsh, editors. *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*. Elsevier, 2006.

[15] Mohammed H. Sqalli and Eugene C. Freuder. Inference-based constraint satisfaction supports explanation. In William J. Clancey and Daniel S. Weld, editors, *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, USA, August 4-8, 1996, Volume 1*, pages 318–325. AAAI Press / The MIT Press, 1996.

[16] S.Sreedharan, S.Srivastava, and S. Kambhampati. Hierarchical expertise-level modeling for user specific robot-behavior explanations. In *Proceedings of the Twenty-Eight International Conference on Automated Planning and Scheduling (ICAPS 2018)*, pages 4829–4836. AAAI Press, 2018.

[17] Paul Stephens. *Mastering Sudoku week by week*. Duncan Baird Publishers, 2007.

Appendix E

Supplementary Material

E.1 Training a AlphaZero-like agent to play Breakthrough

As a testbed for the explainability research, we developed an AlphaZero-like agent to play Breakthrough. The focus was to develop a framework that can play on GPUs and CPUs with asynchronous training and self-play. As well as being flexible enough to train both MuZero (not included in the thesis) and AlphaZero algorithms, the framework handles them similarly with a slight adjustment in the MCTS and the model class. The framework is based on the pseudo-code released from Deepmind and inspired by MIT-licensed frameworks such as [60] using ray.

RAY [61] is a framework to build distributed applications for parallelization, able to run similarly on a local machine and a cluster. It uses actors, each with a local object store, which can communicate and ship objects between each other.

In the developed framework, the first steps are creating the replay buffer, shared storage, and (multiple threads of) self-play actors. The network's initial weights are stored in the shared storage, and the weights are routinely shipped from the shared storage to each running self-playing actor. After each self-playing game, the game record is shipped to the replay buffer.

After we have enough games to start training, we initialize the trainer actor, who trains continuously (with a specified time delay between iterations to avoid overfitting).

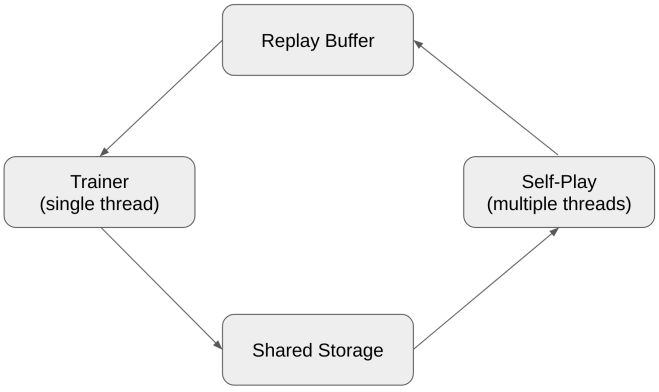


Figure E.1: Visualization of the training framework

The replay buffer ships batches of games to the trainer, and after each iteration, the trainer ships updated weights to the shared storage.

The majority of the time was spent in roughly equal proportions on (i) implementing the algorithms (the AlphaZero and MuZero Neural Networks, MCTS etc), (ii) Implementing game-logic and tuning the algorithms and (iii) framework development. The framework's development focused on handling the amount of data created during self-play, encountering issues such as memory leaks and efficient storage of the data generated.

On a more practical note, we made some learnings during the development of the algorithms:

- The optimal parameters found so far for the game Breakthrough:
  1. batch_size = 512
  2. mcts simulations = 200
  3. Decaying learning rate 0.0001 to 0.00005 with periodic restarts
  4. Neural network with 5 blocks and 64 filters
  5. 30 seconds between training epochs
- Most failed attempts are related to wrongly assuming that the number of games outweighs the quality and diversity of the games. To generate self-playing games faster, we used a lower number of MCTS simulations (e.g. 50) to grow the replay buffer faster, resulting in poorer performance.
- Accelerating training by initializing the replay buffer with self-play games from a pre-trained net seemed to overfit that specific policy and hinder further improvements.
- Keeping older (maybe less accurate) games in the replay buffers seems to be a regularization role and should not be discarded (according to our experiments).

E.2 Hiding Concepts from a Stockfish Agent

In Paper C, we train modified versions of the Stockfish Agent. We intentionally remove concepts from the training dataset and evaluate the impact on the agent's playing strength.

In the Stockfish NNUE trainer repo [14], the developers include all necessary code to train its neural network. They also include external links to recommended training datasets. Training the agent from scratch takes approximately 12 hours using a GeForce RTX 3080 graphics card.

The implementation (in training_data_loader.cpp) uses a highly optimized data loader, which includes a function called make_skip_predicate. This function allows the trainer to skip predetermined positions without losing some of the training dataset's compression ability.

This functionality is very convenient for our use case; we only needed to extend the make_skip_predicate ability to recognize the relevant concepts and remove the desired ones during training. Removing the concepts had minimal effect on the training procedure; even though it skips training samples, it will still fill in the same minibatch size, and thus, we can keep the hyperparameters the same when modifying the training.

Table E.1: Hyperparameters used to train Stockfish Agents

Setting	Value
smart-fen-skipping	True
random-fen-skipping	3
batch-size	16384
threads	2
num-workers	32
gpus	1

For the training we use a dataset generated by *Leela Chess Zero* that is listed as a quality dataset (*training_ data* at [62]). We trained each agent for 500 epochs using the following hyperparametres:

To estimate the drop in playing strength we had them play in a gauntlet style tournament, where all agents agents only play against the Regular agent. Each modified agent plays 3000 matches at search depths 14, 15 and 16 each, thus a total of 9000 matches. Using the game results we evaluate the difference in playing strength using Elo ratings [63].