Training Characteristic Functions with Reinforcement Learning: XAI-methods play Connect Four

Stephan Wäldchen [*1] Felix Huber [*1] Sebastian Pokutta [1]

Abstract

Characteristic functions (from cooperative game theory) are able to evaluate partial inputs and form the basis for attribution methods like Shapley values. These attribution methods allow us to measure how important each input component is for the function output—one of the goals of explainable AI (XAI). Given a standard classifier function, it is unclear how partial input should be realised. Instead, most XAI-methods for black-box classifiers like neural networks consider counterfactual inputs that generally lie off-manifold, which makes them hard to evaluate and easy to manipulate. We propose a setup to directly train characteristic functions in the form of neural networks to play simple two-player games. We apply this to the game of Connect Four by randomly hiding colour information from our agents during training. This has three advantages for comparing XAI-methods: It alleviates the ambiguity about how to realise partial input, makes off-manifold evaluation unnecessary and allows us to compare the methods by letting them play against each other.

1. Introduction

The safe deployment of AI-systems in high-stakes applications such as autonomous driving (Schraagen et al., 2020), medical imaging (Holzinger et al., 2017) and criminal justice (Rudin & Ustun, 2018) requires that their decisions can be subjected to human scrutiny. The most successful models, often based on machine learning (ML) and deep neural networks (DNN), have instead grown increasingly complex and are widely regarded to operate as black-boxes. This spawned the field of explainable AI (XAI) with the explicit aim to make ML models transparent in their reasoning.

[*]Equal contribution [1]TU Berlin & Zuse Institut Berlin. Correspondence to: Stephan Wäldchen <waeldchen@zib.de>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

1.1. Explainable Artificial Intelligence

Though XAI had practical success, such as detecting biases in established data sets (Lapuschkin et al., 2019), there is currently no consensus among researchers about what exactly constitutes an explainable model (Lipton, 2018). For a good overview see (Adadi & Berrada, 2018).

Models such as k-nearest neighbours, succinct decision trees or sparse linear models are deemed inherently interpretable (Arrieta et al., 2020), which makes them preferable (Rudin, 2019). However, the most impressive breakthroughs in the field of AI have only been possible with DNNs. In this light, a second paradigm emerged: to apply these successful models and explain them post-hoc.

In this work, we focus on *saliency* (or *relevance*) attribution methods. Given a classifier and input, these methods rate the importance of each feature for the classifier output, often displayed visually as a heatmap, called a saliency map. We give an overview over the proposed methods in Section 2.

1.2. Characteristic Functions

Cooperative game theory considers attribution problems very similar to saliency attribution, where a common pay-off is to be fairly distributed to a number of players. In the context of ML, players correspond to features and the pay-off is the classifier score. Let $d \in \mathbb{N}$ be the number of features, and $[d] = \{1, \dots, d\}$. One key concept is the *characteristic function* $\nu \colon 2^{[d]} \to \mathbb{R}$ which assigns a value to every possible subset of d, called a *coalition*. We refer the reader to (Chalkiadakis et al., 2011) for a good introduction.

For binary classifiers this led to the concept of prime implicant explanations (PIE) that search for the smallest coalition $S \subset [d]$ that ensures a value of $\nu(S) = 1$, see (Shih et al., 2018). These explanations can be efficiently computed for certain simple classifiers.

The Shapley values, an established method defined as

$$\phi_{\nu,i} = \sum_{S \subseteq [d] \setminus \{i\}} \binom{d-1}{|S|}^{-1} (\nu(S \cup \{i\}) - \nu(S)),$$

are the unique attribution methods that satisfy certain desirable fairness criteria, see (Shapley, 2016) and Appendix E.1.

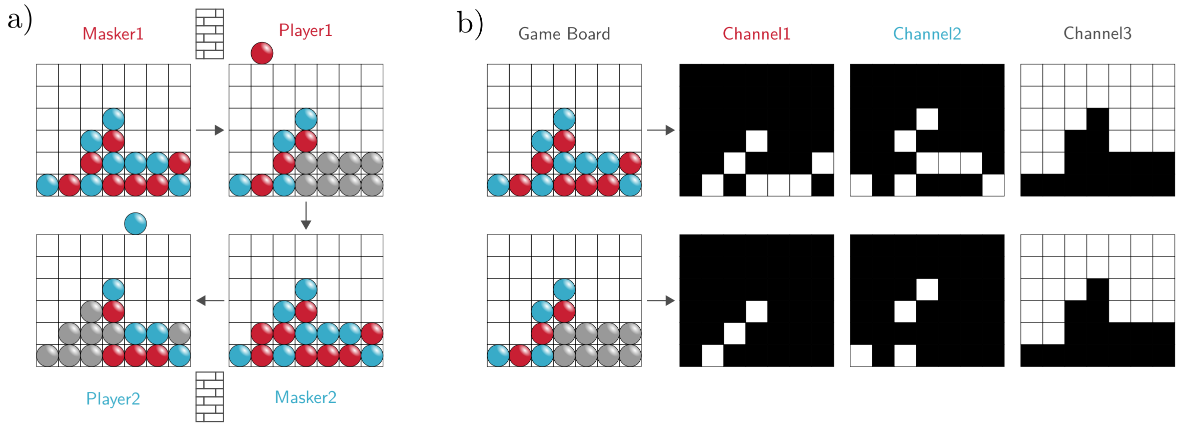


Figure 1. Connect 4 with missing colour information shown in grey. a) A game between two pairs of maskers and players. The maskers decide which colour information to pass on to the players with an upper limit of revealing half the played pieces. The player has to decide his move based on the information sent by the masker. The players are both modelled by the same policy network. Which information to reveal will later be chosen by different saliency methods. b) The game board is encoded in three input channels: Two binary matrices indicating the pieces of each player and one indicate the open fields.[2] With full information the encoding is redundant, with only partial information the agent is unsure about the colour of some pieces but can still make valid moves.

Both prime implicant explanations as well as Shapley values have inspired heuristic saliency methods for neural networks (see Section 2), though none maintain their desirable theoretical properties while at the same time being computationally efficient (Wäldchen et al., 2021).

1.3. Evaluating Saliency Methods

In (Doshi-Velez & Kim, 2017) the authors differentiate between *human-based* and *functionally-grounded evaluation*. The former has the advantage of measuring directly which explanations are legible and helpful to humans, but are costly and hard to generalise from one task to another. The latter aims to design proxy tasks whose success is correlated with the quality of the explanation, see e.g. (Pruthi et al., 2022). This allows for larger scale experiments, however, not all proxy-task are necessarily linked to good explanations (Biessmann & Refiano, 2021).

The proxy task we consider here is: *successful play of abstract games with limited information.* In our case study we investigate the game *Connect Four* (Allis, 1988). We make use of the fact that neural networks have emerged as one of the strongest models for reinforcement learning, and e.g., constitute the first human competitive models for Go (Silver et al., 2017) and Atari games (Mnih et al., 2015).

Our exact setup is illustrated in Figure 1. A *masker* and a *player* are paired against a second team of the same form. The masker, equipped with an XAI-method, presents a limited amount of information to their player, who selects the next move. The players are modelled by DNNs without memory and base their decision only on the information currently provided by the masker.

1.4. Our Contribution

We show that for image obfuscation, one of the most used evaluation metrics, the best-performing methods create features that were not present in the original image, a phenomenon resulting from evaluating classifiers off-manifold.

As a remedy, we directly train agents as characteristic functions for reinforcement learning, by randomly hiding colour features and show that this setup delivers results comparable to training solely on full information. Additionally, we demonstrate a relatively monotonous relationship between information and performance in the game, which justifies the setup explained in Figure 1 as a sensible proxy task for XAI methods.

Since our agents can handle partial input, we can directly compute Shapley values via sampling. These are theoretically well understood and rely only on counterfactual input the agent has been trained on—which makes this a sound attribution method. We demonstrate their usefulness by comparing to the ground truth available for certain board situations. Additionally, we learn that training on hidden input is not enough to ensure interpretability if the training is linked to the wrong objective.

We then compare a selection of XAI methods in a round-robin tournament in Connect Four, which has some advantages that image obfuscation comparisons generally lack:

1. It is canonically clear how missing information should be modelled (since it is included in the training).
2. There is no need to evaluate the classifier off-manifold.
3. We have a concrete task (Winning the game) to compare the XAI-methods.

2. Related Work

We restrict our analysis to local, post-hoc saliency methods for neural network classifiers, both model specific and model-agnostic. We differentiate the following three categories.

Local Linearisation Linear methods are considered interpretable, so it is a natural approach for nonlinear models to instead interpret a local linearisation. In this category we find gradient maps (Simonyan et al., 2013), SmoothGrad (Smilkov et al., 2017) and LIME (Ribeiro et al., 2016) which samples around the input and fits a new linear classifier.

Heuristic Backpropagation These methods replace the chain-rule of gradient backpropagation with different heuristically motivated rules and propagate relevance scores back to the input (Zeiler & Fergus, 2014). Newer methods include GuidedBackpropagation (GB) (Springenberg et al., 2015), DeepLift (Shrikumar et al., 2017), DeepShap (Lundberg & Lee, 2017a) and LRP (Bach et al., 2015). These methods have the advantage of being comparably fast.

Partial Input These methods rely on a characteristic function $\nu_{f,\mathbf{x}}$ for a classifier function f and an input $\mathbf{x}$. The standard way to define $\nu_{f,\mathbf{x}}(S)$ for a feature set S, put forth by (Lundberg & Lee, 2017b), is to regard the missing features $\mathbf{x}_{S^c}$ as random variables and take an expectation value conditioned on the given parameters $\mathbf{x}_S$, i.e.,

$$\begin{aligned}\nu_{f,\mathbf{x}}(S) &= \mathbb{E}_{\mathbf{y}}[f(\mathbf{y}) \,|\, \mathbf{y}_S = \mathbf{x}_S] \\ &= \int f(\mathbf{x}) p(\mathbf{x}_{S^c} \,|\, \mathbf{x}_S)\, \mathrm{d}\mathbf{x}_{S^c}. \end{aligned} \tag{1}$$

Being able to evaluate partial input, these methods either optimise an objective similar to prime implicant explanations, for example *Rate-Distortion Explanations* (RDE) (Macdonald et al., 2020) and *Anchors* (Ribeiro et al., 2018), or approximate Shapley values (Sundararajan & Najmi, 2020). This is computationally expensive, but has the advantage of being close to theoretically sound concepts. This soundness, however, depends strongly on how correctly the conditional distribution $p(\mathbf{x}_{S^c} \,|\, \mathbf{x}_S)$ is modelled, as we will discuss next.

The Problem with Off-Manifold Input These post-hoc methods share that they consider counterfactual information: *"What if I would change parts of the input, or leave them out completely?"*, see Appendix A for more detail.

These saliency methods can all be manipulated by principally the same idea: replace an existing model by another one that agrees on the data manifold but not off-manifold. This allows to hide biases in classifiers for *on-manifold* inputs almost at will, as demonstrated for gradient maps and integrated gradients (Anders et al., 2020; Dimanov et al., 2020), LRP (Anders et al., 2020; Dombrowski et al., 2019), LIME (Slack et al., 2020; Dimanov et al., 2020), DeepShap (Slack et al., 2020; Dimanov et al., 2020), Grad-Cam (Heo et al., 2019), Shapley-based(Frye et al., 2020) and general counterfactual explanations (Slack et al., 2021). Alternatives to saliency methods use generative models to produce counterfactuals that stay on-manifold (Nguyen et al., 2016; Booth et al., 2021; Wäldchen et al., 2022).

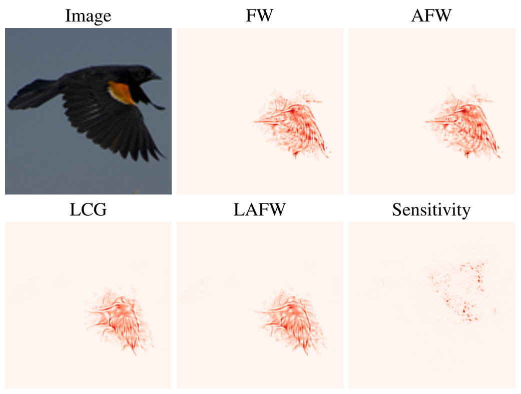


Figure 2. RDE-Explanations of a bird image taken from (Macdonald et al., 2021) with permission of the authors. The proposed optimisation methods (FW, AFW, LCG, LAFW) search for the smallest set of features that still maintain the classification of "bird". All produce a mask that creates a new bird head as a mask. The sensitivity map does not show this behaviour.

Image Obfuscation In the absence of human annotations (such as bounding boxes or pixel-wise annotations), some functionally grounded evaluation for image data obfuscate part of the image and measure how much the classifier output changes (Mohseni et al., 2021). The idea is this: keeping the relevant features intact should leave the classification stable, obfuscating them should rapidly decay the classifier score. This method was introduced as *pixel-flipping* (Samek et al., 2017) and used to evaluate XAI-methods for image recognition (Fong & Vedaldi, 2017; Macdonald et al., 2019) and Atari games (Huber et al., 2021).

In (Macdonald et al., 2019) the authors directly optimise for a mask that selects sparse features which maintain the classifier decision, severely outperforming competitors. In (Macdonald et al., 2021) this method gets improved further with methods from convex optimisation (Frank-Wolfe optimiser). However, visually inspecting the produced saliency maps reveals they create new features that were not present in the original image. This is possible over a mechanism similar to adversarial examples, see Figure 2, which means: *the optimal mask creates its own features!* This shows that proxy tasks have to be designed with care if they are meant to be useful for comparing saliency methods. Not only are the methods vulnerable to manipulation by going off-manifold, the evaluation tasks themselves can be exploited by making use of off-manifold inputs.

3. Setup

Instead of trying to turn a classifier function into a characteristic function, we propose to directly train on hidden features. Policy and value functions (see (Li, 2017)) for agents that play simple two-player, turn-based games (such as Go, Checkers, Hex) are particularly well suited for this task.

The logic of these games is complex enough to make the use of black-box functions such as neural network sensible, and in the case of Go the unbeaten standard. At the same time, the input is low-dimensional and discrete. Additionally, the input components have weaker correlations between neighbours which makes hiding information easier. Hiding random pixels in an image can be undone by inpainting, whereas the same process would be very difficult for a Connect Four game board. These factors facilitate sampling partial inputs during training.

3.1. Hiding the Player Colour

The most straight-forward way of hiding information from an agent would be to completely hide a game field. However, we want to preserve the ability to select legal moves, and let the sensibility of the move be the only concern of the agent (instead of legality). For a lot of the games (e.g., Connect Four, Go, Hex) knowing which fields are occupied allows to make valid moves. For others (Chess, Checkers) valid moves depend on the colour and type of the pieces, so we will concentrate on the former.

To hide the colour information, we represent a game position as three binary matrices indicating which fields are occupied by the first player (red), the second player (blue) and which remain free. The colour information can be hidden by setting the entries in the respective matrix to zero. We illustrate this concept in Figure 1 a) for the game of Connect Four.

3.2. Reinforcement Learning for Connect Four

The game of Connect Four was chosen because of its simplicity and low input dimension. Deep-RL has been applied to Connect Four in the form of both Deep-Q-Learning (Dabas et al., 2022), Policy Gradient (PG) (Crespo, 2019) and AlphaZero-like approaches (Wang et al., 2021; Clausen et al., 2021). The AlphaZero-like approach combines a neural network with policy and value head with a MCTS[3] lookahead to make its decision. Even though it has emerged as the most powerful method, we want to explain purely the network decision without the MCTS involved. We thus follow the PPO approach of Schulman et al., since it yields a strongly performing pure network-agent for Connect Four even without the help of the MCTS (Crespo, 2019).

[3]Monte-Carlo Tree Search, see (Silver et al., 2017)

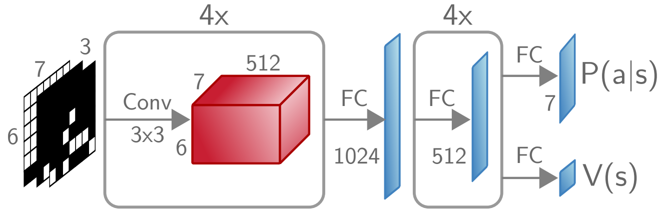


Figure 3. Network architecture for the PPO-agent. The convolutional layers (Conv) have 3×3 filters, 512 channels, 1×1 stride and use zero-padding. The fully connected (FC) layers for the policy and value head use softmax and tanh-activation respectively. All other layers use ReLU-activations.

Our PPO-agent We represent the input by a state $\mathbf{s} \in \{0,1\}^{3\times6\times7}$ with three channels as explained in Figure 1 (b). The value function that predicts the expected reward $V(\mathbf{s})$ and the policy function that determines the probability $P(a \mid \mathbf{s})$ of taking action $a \in [7]$ are both modelled by the same network with with a policy and a value head, see Figure 3. Our architecture extends the model of Crespo by two fully connected layers, which empirically yields better performance. Agents can play *competitively*, always choosing the most likely action $a^* = \operatorname{argmax} P(\cdot \mid \mathbf{s})$ or *non-competitively*, sampling from $P(\cdot \mid \mathbf{s})$. We give a full overview over architecture and training parameters in Appendix B.

Hiding features during training During self-play, we randomly hide the colour information of a certain percentage of fields by setting the respective entry in the the first and second input channel to zero, see Figure 1 b). The information that the field is occupied remains in the third channel. Every turn p_h is drawn uniformly from $[0, p_h^{\max}]$. Let $t \in [42]$ be the turn number, then we hide the colour information of $\lfloor p_h t \rfloor$ random pieces selected uniformly at random. We explored different values for $p_h^{\max}$ and trained the following agents:

- FI: PPO-Agent trained with full information.
- PI-50: Partial information with $p_h \sim \mathcal{U}([0, 0.5])$
- PI-100: Partial information with $p_h \sim \mathcal{U}([0, 1])$

3.3. Benchmarking

To demonstrate that this setup still trains capable agents we compare them to the benchmark results from the original setup presented in (Crespo, 2019). We let our agents play in competitive mode against an MCTS-agent taken from (Vogt, 2019). For three different difficulties, the MCTS is allowed to simulate 500, 1000 or 2000 games. Additionally, we used

a) Win Rate against MCTS

	Orig.	FI	PI-50	PI-100
MCTS 500	0.92	**0.972**	0.793	0.684
MCTS 1000	0.896	**0.936**	0.66	0.469
MCTS 2000	0.825	**0.91**	0.497	0.328

b) Number of Optimal Moves

Agent	Orig.	FI	PI-50	PI-100
Correct Moves	38	**39**	**39**	29

Table 1. Comparison of our agent with the original proposal by (Crespo, 2019) in competitive mode. The numbers show the winrate against the MCTS with different simulation limits over 1000 games. b) For a game between two Connect Four solvers, we tracked how many of the optimal moves were correctly predicted by the different agents, i.e., were given the highest probability in the policy output. The optimal game always takes 41 moves in total.

a game played by two perfect solvers[4] and measured how many of the 41 moves were predicted correctly by our agents. For the results see Table 1. Our FI-agents performs best, both against the MCTS and predicting the optimal moves. Incorporating partial information into the training leads to a worse performance by PI-50 and PI-100. Nevertheless, at least the PI-50 is a capable agent that could not be beaten by the authors. We thus opt to use the PI-50 agent to compare the saliency methods in Section 5.

To show that for each agent more information is indeed useful[5], we tracked their performance playing non-competitively for different amounts of randomly hidden colour features, see Figure 4. For the FI-agent and the PI-50 agent we see near-monotonous decay in game performance against the optimal agent, an MCTS-1000 and themselves with full information. This justifies our idea of a proxy task to compare saliency methods: select the most useful 50% of features that allow the PI-50 agent to win the game!

4. Explanations with Partial Information

Since our agents have been trained with missing information, their policy and value functions can be seen as a characteristic function with respect to the colour features. For $t \in [42]$ let $\mathbf{s}_t \in [0,1]^{3\times6\times7}$ be a board state after turn t and $S \subseteq [t]$ be a set of colour features. Then we define $\mathbf{s}^{(S)}$ as a partial board state including only the colour features in S. Let $a^* = \operatorname{argmax} P(\,\cdot\,;\mathbf{s})$, then we can define $\nu^{\text{pol}} : 2^{[t]} \to [0,1]$ and $\nu^{\text{val}} : 2^{[t]} \to [-1,1]$ as

$$\nu^{\text{pol}}(S) = P(a^*;\mathbf{s}^{(S)}) \quad \text{and} \quad \nu^{\text{val}}(S) = V(\mathbf{s}^{(S)})$$

[4]Taken from (Pons, 2019)

[5]This is not obvious: during training the agent could get used to an average amount of partial information and play worse if given more colour features.

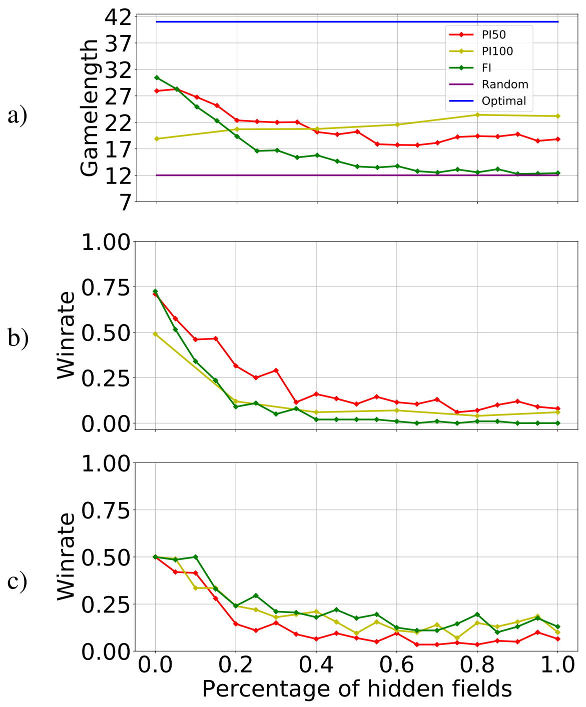


Figure 4. Relationship between the percentage of hidden fields and game performance, measured in game length against a perfect solver (a), winrate against an MCTS1000 (b) and against the agent itself with full information (c) for the FI, PI-50 and PI-100 agents. In (a) we give as reference a random agent who plays an average of 12 turns before losing and another optimal solver who always plays 41 turns. Our agents never win against the solver. The FI-agent start out strongest, but was never trained on hidden features, so drops towards the random agent. The PI-50 starts with slightly weaker game length but keeps an advantage over random even for no colour information at all. The PI-100 agents shows a weakest performance given full information but even rises slight with less information. In (b) and (c) all agents decrease in performance with fewer features revealed.

and interpret them as the characteristic functions from the policy and value output respectively. From now on, we use the characteristic function associated with the *policy* network, since this is the part that actually plays the games, whereas the value network is only involved in training.

This allows us to directly compute explanations for them in the form of Shapley values or prime implicant explanations. We now explain how we efficiently approximate both.

4.1. Sampling Shapley Values

Computing Shapley values is #P-complete (Deng & Papadimitriou, 1994) , but they can be efficiently approximated by sampling. The simplest approach is to utilise the fact that the Shapley values for t features can be rewritten as

$$\phi_i = \frac{1}{t!}\sum_{\pi\in\Pi([t])} (\nu(P_i^\pi \cup \{i\}) - \nu(P_i^\pi)), \qquad (2)$$

where $\Pi([t])$ is the set of all permutations of $[t]$ and P_i^π the set of all features that precede i in the order π. To approximate the whole sum we can sample uniformly from $\Pi([t])$. To stick true to our philosophy of evaluating ν only on-manifold, we can only use PI-100 to calculate the Shapley-values who has been trained on all levels of hidden features. To use the PI-50 agent, we define *partial* Shapley-values for a hidden percentage p_h by only sampling from

$$\Pi_i^{p_h} = \{\pi \in \Pi([t]) \text{ s.t. } |P_i^\pi| \geq p_h t\},$$

which are all permutations that have input i in the last $p_h t$ position. Sampling from $\Pi_i^{0.5}$ makes sure that at least 50% of colour information is disclosed. We show in Appendix C that we keep the symmetry, linearity and null player criterion, but lose efficiency.

To get an (ϵ, δ)-approximation $\bar{\phi}_i$ of ϕ_i in the sense that $\mathbb{P}\left[\left|\phi_i - \bar{\phi}_i\right| \leq \epsilon\right] \geq 1-\delta$ we need $N_{\epsilon,\delta} = \frac{1}{2}\epsilon^{-2}\log(2\delta^{-1})$ many samples, according to the worst-case bound given by the Hoeffdings-inequality (Hoeffding, 1994). For our comparison we choose a $(0.01, 0.01)$-approximation which amounts to ≈ 26500 samples.

More efficient methods to sample Shapley-values have been developed utilising group testing (Jia et al., 2019) or kernel-herding (Mitchell et al., 2021) both providing a quadratic improvement of the accuracy in terms of number of function evaluations although with some computational overhead.

4.2. Prime Implicant Explanations with Frank-Wolfe

Prime implicant explanations can be efficiently calculated for simple classifiers like decision trees or ordered binary decision diagrams (Shih et al., 2018). In (Macdonald et al., 2020) the authors extended the definition of prime implicants to a continuous probabilistic setting to explain neural networks. They optimise for implicants of size $k \in \mathbb{N}$ via convex relaxation, which forms the basis for RDE. Adapted to our scenario the corresponding objective becomes

$$S^* = \underset{|S|\leq k}{\operatorname{argmin}}(\nu([T]) - \nu(S))^2.$$

Whereas Macdonald et al. rely on an approximation to Equation (1), our formulation has no probabilistic aspect, since we directly access a characteristic function. In (Macdonald et al., 2021) the authors show how to minimise this functional efficiently with Frank-Wolfe solvers, a projection-free method for optimisation on convex domains (Pokutta et al., 2020). We copy their approach and apply it to find small prime implicant explanations for varying k, a saliency attribution which we call the *FW-method*. More details about the FW-method are included in Appendix D. The Frank-Wolfe method might attribute the maximum weight of 1 to multiple features, so we break ties randomly when selecting the most salient ones.

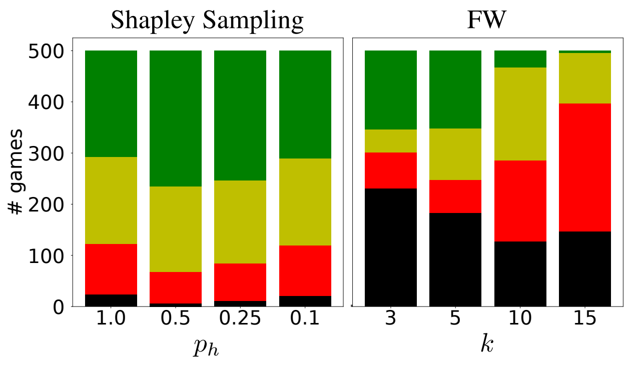


Figure 5. Comparison of the Frank-Wolfe-based and Shapley Sampling-based attribution methods with the ground truth. The bars show how often the method identified the 3 (green), 2 (yellow), 1 (red) or 0 (black) of the three most important game pieces. We observe that Shapley sampling is able to identify the most pieces correctly for a hidden percentage of $p_h = 0.5$. The Frank-Wolfe method performs generally worse than Shapley Sampling. Smaller k tend to polarise the results with more boards where either all three or zero of the correct pieces have been found.

4.3. Finding Ground Truth Pieces

The policy network of the PI-agent is able to find a winning move in 99% of cases. It stands to reason that the three pieces that are completed by the move can be seen as *ground truth* features for the decision.

We use this to compare partial Shapley Values with the FW-method. We let the agent play against itself and registered 500 final board states for which the agent was at least 90% sure of the winning move. All game pieces that form a line of four with the winning move[6] are considered as ground truth for the focus of the policy network. We track how often the three most salient pieces according to the attribution methods are among the ground truth pieces. For 500 games we note whether three, two, one or zero pieces are correct, see Figure 5, for partial Shapley values with varying p_h and the FW-method for varying k.

Shapley Sampling Partial Shapley values generally yield good results, finding at least two correct pieces in at least 75% of all boards. However, calculating the Shapley-values all the way up to $p_h = 1$ gives worse results than stopping at 0.5, both for PI-50 and PI-100. The plots for PI-100 can be found in Appendix E.1. Further investigating revealed that the output of the PI-100 policy function for $p_h > 0.5$ becomes essentially random for many board situations. A possible reason is that for low information the optimal move becomes ill-defined, and the entropy loss is to be too small to force the policy to be approximately uniform. We will come back to this point in Section 7.

[6]It can happen that one move completes multiple lines.

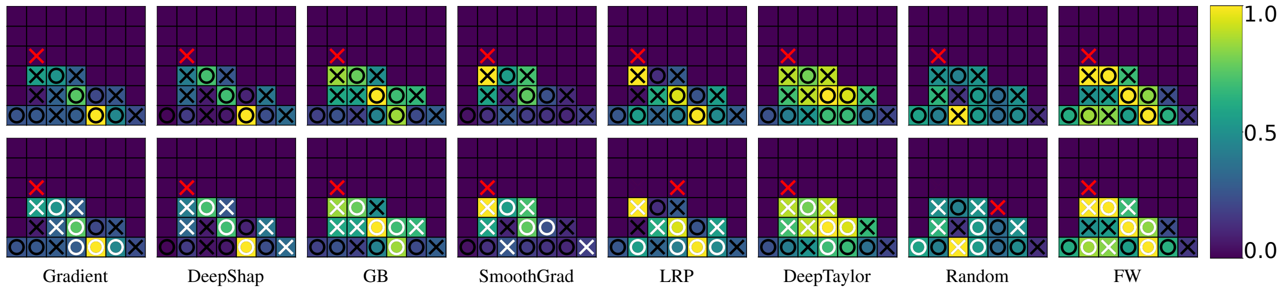


Figure 6. Top: One move explained by different saliency maps: The PI-50 agent placed a piece in the second column, blocking a potential win by their opponent (red cross). The saliency maps for this move are shown for each method. *Bottom:* The agent is presented with the colour information of the 50% most salient pieces according to each method (marked in white). This can lead to a different decision, here for LRP and Random, allowing the opponent to win.

Frank Wolfe Solver The FW method is faster than Shapley sampling, but exhibits worse performance. Interestingly, for small k it polarises, having a higher percentage of boards where it finds either all or non of the ground truth pieces. For large k it usually converges to an attribution that selects many pieces with maximum value of 1. Breaking ties randomly then leads to average results. For small k it oftentimes selects pieces that suggest the right move but for a different reason than the ground truth pieces. Additionally, the FW performance suffers from relying on convex relaxation of set membership, which means optimising over continuous colour features thus going off-manifold. This can be seen by the fact that the continuous input values almost always lead to the right policy, but selecting the k most salient features, thus thresholding the input back to binary values, sometimes leads to a different policy.

5. Comparing XAI-methods

We can now compare different saliency methods via the setup explained in Figure 1. As our agent we select PI-50 and allow to show 50% of colour features, thus remaining on-manifold. To implement the masker, we let an XAI-method explain the decision of the policy function for the most probably action. For every occupied field we sum the absolute value of saliency score from the first two colour channels (that represent which player occupies the field). The saliency scores on empty fields and on the third channel are ignored. Then we select the 50% (rounded up) highest scoring colour features, breaking ties randomly, and hide the rest from the board state (set them to 0). This state is then used by the player, (PI-50, non-competitive) to select a move. We present an illustration for all used saliency methods in Figure 6.

We compare the saliency methods Gradient, GuidedBackprop (GB) SmoothGrad, LRP-ϵ, DeepTaylor and Random from the Innvestigate[7] (Alber et al., 2018) and DeepShap from the SHAP[8] toolboxes with recommended settings. Random, which assign a random Gaussian noise as saliency value, and the FW-method explained in the previous chapter serve as comparison for the other methods.

We let each saliency method compete in a round-robin tournament with 1000 games for each encounter. In case of a draw, both methods score half a victory. We display the number of victories for each encounter in Figure 7 together with two challenges described in Section 4.3 and Section 3.3.

Results The results are displayed in Figure 7. The methods form two groups of performance. DeepShap, GuidedBackprop and the Frank-Wolfe-based method perform best and equally well. The second group is formed by gradient, LRP and DeepTaylor who show a weaker performance. SmoothGrad has the worst showing, potentially owing to the fact that it does not automatically sample other valid board situations. These results are confirmed by the information-performance graphs, introduced in Section 3.3 for play against PI-50 with full information, where instead of randomly selecting the revealed features they are selected by each saliency method, see Figure 7. We complement the tournament results with their standard deviation over 10 training runs in the appendix in Figure 11.

Regarding FW, we showed in Section 4 that the optimiser does not always find what we consider the important pieces but rather the pieces that ensure the policy network makes the right decision. Thus DeepShap and GuidedBackprop compare very favourably considering FW optimises directly for winning the game.

Additionally, DeepShap, Gradient, LRP-ϵ and GuidedBackprop severely outperform Shapley Sampling in finding the ground-truth pieces. This shows that heuristic methods can have merit over theoretically founded one. We explain a possible reason for the weak performance of the Shapley values and a way forward to improve it in Section 6.

[7]https://github.com/albermax/innvestigate

[8]https://github.com/slundberg/shap

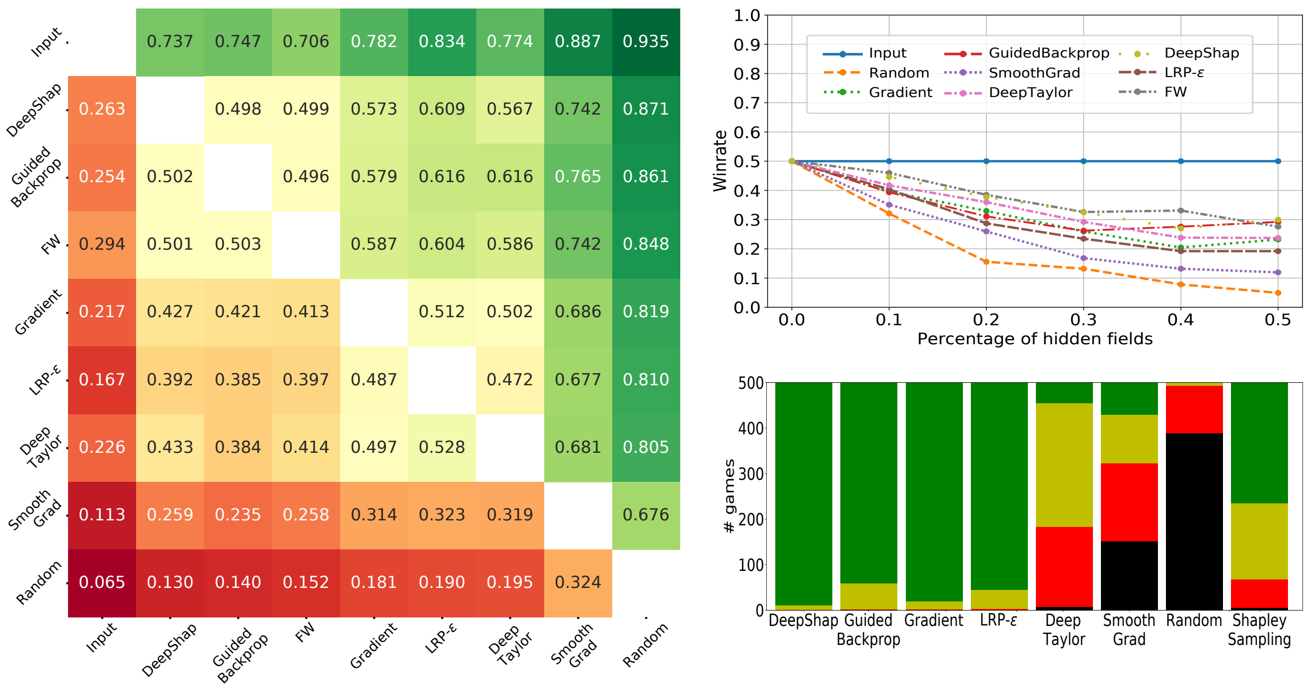


Figure 7. Comparison of different XAI-methods playing Connect Four. *Left:* Win rates row vs column, with draws counting as $\frac{1}{2}$. *Top right:* Win rate of PI-50 with varying percentage of hidden information against itself. The partial colour features have been selected using the XAI-methods. The win rate of the agent decays slower, when the method is better at selecting the crucial information. *Bottom right:* Comparison of the XAI-methods in the ground truth task described in Section 4.3. We can see that DeepShap and GuidedBackprop perform well in all these tasks.

6. Limitations and Outlook

We considered the game Connect Four since legal moves can reliably be made when only colour information is missing. This allowed us to train our agents to make almost no illegal moves. However, our setup can be generalised beyond such games via a model-based RL-agent. E.g., the AlphaGo-policy network filters out illegal moves before the final softmax of the policy layer. Following this approach extends the setup to any abstract game.

A further extension to real-world data is indeed nontrivial. Complex data like images or videos often encodes information redundantly, so hiding it is an involved task with many uncertainties. This is precisely why we propose simple games as a proxy task. We compare the XAI-methods in one domain (optimal for evaluation) to decide which ones are promising enough to employ in another (e.g., real-world). We can expect the performance to carry over if the method itself is based on sound principles. So far, no method has been proven sound since there are complexity barriers (Macdonald et al., 2020), but the property can at least be falsified using theoretical and applied sanity checks (Adebayo et al., 2018; Nie et al., 2018; Sixt et al., 2020). Thus, both practical evaluation on proxy-tasks and theoretical analysis of the methods are necessary.

One problem we discovered is that the sampled Shapley values become less performant for higher rates of missing information. In our setup the policy network is trained to predict sensible game actions—a task that becomes increasingly ill-defined for low information input. The entropy term in the policy loss, see Appendix B, was not strong enough to force the policy towards a uniform distribution over all actions. Thus the network output in this regime was essentially random which can dominate the summation in Equation (2). Using a well-trained value head instead of the policy head could stabilise this behaviour. The value head should conservatively estimate the win probability close to 0.5 for different configurations of large hidden information. These terms will then cancel out in Equation (2).

Training a value head capable of playing the game can be achieved through Q-learning—even in scenarios where it performs slightly worse than PPO as it only indirectly optimises for policy. Q-learning trains a value function to obey an consistency condition in form of the Bellmann-equation (Sutton & Barto, 2018). This indirectness can become useful since the objective remains well-defined even if almost no information is given, and the agent defaults to a conservative estimate. A similarly option for supervised learning on partial information is to include a default "I don't know" output that is preferred to a wrong answer. Then the objective on low information data becomes well defined and techniques such as Shapley sampling can be used as a theoretically sound saliency method. Tempering with the model to hide biases would require changing on-manifold behaviour which could be detected through performance tests.

7. Conclusion

We have demonstrated that simple game setups can be used to train agents capable of handling missing features. This allowed us to design a proxy task for XAI-methods based on the idea that such agents make better decisions if provided more relevant information. We evaluate a collection of saliency methods and see strong performances for DeepShap and GuidedBackprop. Whether this performance will carry over to real-world applications depends on the theoretical soundness of the methods. We thus encourage further investigation into sanity checking the well-performing methods.

Prime Implicant Explanations and Shapley values are defined over desirable properties, and the actual algorithm to compute them depends on the model. In contrast, most saliency attribution methods for neural networks are defined directly over algorithmic instructions and lack definite properties that make them useful. Thus experimental evaluation is necessary to assess their merit. We thus proposed a new proxy-task that allows us to alleviate shortcomings present in other setups.

As explained in Section 2, using proxy tasks that evaluate the classifier off-manifold can have paradoxical consequences. The optimal strategy can lie in using a kind of "super-stimulus" mask, analogously to adversarial examples. In this case the mask is not adversarial to the original classification but rather ensures that the network output stays the same by manufacturing new features. However, as for adversarial examples this is done by exploiting off-manifold bahaviour of the network. To avoid this, we directly train our networks on the partial information used for evaluation.

The resulting characteristic function given by the policy network allows us to estimate Shapley values via sampling. This attribution method is theoretically well understood and only relies on on-manifold inputs which justifies trusting the resulting saliency maps. We observed, however, that the Shapley values become unreliable for high rates of missing information. We discuss why using the value function should be more stable and encourage to research how the Shapley sampling performs for a Q-learning setup.

In our investigation the heuristic saliency methods compared very favourably to the more theoretically founded methods. If they can be made robust to manipulation, e.g., by approaches from (Frye et al., 2020) and (Anders et al., 2020), they could constitute promising tools for XAI.

ACKNOWLEDGEMENTS

This research was partially supported by the DFG Cluster of Excellence MATH+ (EXC-2046/1, project id 390685689) and the Research Campus Modal funded by the German Federal Ministry of Education and Research (fund numbers 05M14ZAM,05M20ZBM).

References

Adadi, A. and Berrada, M. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. Sanity checks for saliency maps. *arXiv preprint arXiv:1810.03292*, 2018.

Alber, M., Lapuschki, S., Seegerer, P., Hägele, M., Schütt, K. T., Montavon, G., Samek, W., Müller, K., Dähne, S., and Kindermans, P. iNNvestigate neural networks! *CoRR*, abs/1808.04260, 2018. URL http://arxiv.org/abs/1808.04260.

Allis, L. V. A knowledge-based approach of connect-four. *J. Int. Comput. Games Assoc.*, 11(4):165, 1988.

Anders, C., Pasliev, P., Dombrowski, A.-K., Müller, K.-R., and Kessel, P. Fairwashing explanations with off-manifold detergent. In *International Conference on Machine Learning*, pp. 314–323. PMLR, 2020.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46, 07 2015. doi: 10.1371/journal.pone.0130140.

Biessmann, F. and Refiano, D. Quality metrics for transparent machine learning with and without humans in the loop are not correlated. *arXiv preprint arXiv:2107.02033*, 2021.

Booth, S., Zhou, Y., Shah, A., and Shah, J. Bayes-trex: a bayesian sampling approach to model transparency by example. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11423–11432, 2021.

Chalkiadakis, G., Elkind, E., and Wooldridge, M. Computational aspects of cooperative game theory. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(6):1–168, 2011.

Clausen, C., Reichhuber, S., Thomsen, I., and Tomforde, S. Improvements to increase the efficiency of the alphazero algorithm: A case study in the game'connect 4'. In *ICAART (2)*, pp. 803–811, 2021.

Crespo, J. Reinforcement learning for two-player zero-sum games. Master's thesis, Tecnico Lisboa, `https://fenix.tecnico.ulisboa.pt/downloadFile/1689244997260153/81811-joao-crespo_dissertacao.pdf`, 2019.

Dabas, M., Dahiya, N., and Pushparaj, P. Solving connect 4 using artificial intelligence. In *International Conference on Innovative Computing and Communications*, pp. 727–735. Springer, 2022.

Deng, X. and Papadimitriou, C. H. On the complexity of cooperative solution concepts. *Mathematics of operations research*, 19(2):257–266, 1994.

Dimanov, B., Bhatt, U., Jamnik, M., and Weller, A. You shouldn't trust me: Learning models which conceal unfairness from multiple explanation methods. In *SafeAI@ AAAI*, 2020.

Dombrowski, A.-K., Alber, M., Anders, C. J., Ackermann, M., Müller, K.-R., and Kessel, P. Explanations can be manipulated and geometry is to blame. *arXiv preprint arXiv:1906.07983*, 2019.

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

Fong, R. C. and Vedaldi, A. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE international conference on computer vision*, pp. 3429–3437, 2017.

Frye, C., de Mijolla, D., Begley, T., Cowton, L., Stanley, M., and Feige, I. Shapley explainability on the data manifold. *arXiv preprint arXiv:2006.01272*, 2020.

Heo, J., Joo, S., and Moon, T. Fooling neural network interpretations via adversarial model manipulation. *Advances in Neural Information Processing Systems*, 32: 2925–2936, 2019.

Hoeffding, W. Probability inequalities for sums of bounded random variables. In *The collected works of Wassily Hoeffding*, pp. 409–426. Springer, 1994.

Holzinger, A., Biemann, C., Pattichis, C. S., and Kell, D. B. What do we need to build explainable ai systems for the medical domain? *arXiv preprint arXiv:1712.09923*, 2017.

Huber, T., Limmer, B., and André, E. Benchmarking perturbation-based saliency maps for explaining deep reinforcement learning agents. *arXiv preprint arXiv:2101.07312*, 2021.

Jia, R., Dao, D., Wang, B., Hubis, F. A., Hynes, N., Gürel, N. M., Li, B., Zhang, C., Song, D., and Spanos, C. J. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1167–1176. PMLR, 2019.

Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., and Müller, K.-R. Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, 10(1):1–8, 2019.

Li, Y. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

Lipton, Z. C. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 4765–4774. Curran Associates, Inc., 2017a.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pp. 4768–4777, 2017b.

Macdonald, J., Wäldchen, S., Hauch, S., and Kutyniok, G. A rate-distortion framework for explaining neural network decisions. *arXiv preprint arXiv:1905.11092*, 2019.

Macdonald, J., Wäldchen, S., Hauch, S., and Kutyniok, G. Explaining neural network decisions is hard. In *XXAI Workshop, 37th ICML*, 2020.

Macdonald, J., Besançon, M., and Pokutta, S. Interpretable neural networks with frank-wolfe: Sparse relevance maps and relevance orderings. *arXiv preprint arXiv:2110.08105*, 2021.

Mitchell, R., Cooper, J., Frank, E., and Holmes, G. Sampling permutations for shapley value estimation. *arXiv preprint arXiv:2104.12199*, 2021.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Mohseni, S., Zarei, N., and Ragan, E. D. A multidisciplinary survey and framework for design and evaluation of explainable ai systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 11(3-4):1–45, 2021.

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., and Clune, J. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *Advances in neural information processing systems*, 29, 2016.

Nie, W., Zhang, Y., and Patel, A. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In *International Conference on Machine Learning*, pp. 3809–3818. PMLR, 2018.

Pokutta, S., Spiegel, C., and Zimmer, M. Deep neural network training with frank-wolfe. *arXiv preprint arXiv:2010.07243*, 2020.

Pons, P. Connect 4 game solver, 2019. URL `https://github.com/PascalPons/connect4`.

Pruthi, D., Bansal, R., Dhingra, B., Soares, L. B., Collins, M., Lipton, Z. C., Neubig, G., and Cohen, W. W. Evaluating explanations: How much do explanations from the teacher aid students? *Transactions of the Association for Computational Linguistics*, 10:359–375, 2022.

Ribeiro, M. T., Singh, S., and Guestrin, C. " 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*, pp. 1135–1144, 2016.

Ribeiro, M. T., Singh, S., and Guestrin, C. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

Rudin, C. and Ustun, B. Optimized scoring systems: Toward trust in machine learning for healthcare and criminal justice. *Interfaces*, 48(5):449–466, 2018.

Samek, W., Binder, A., Montavon, G., Lapuschkin, S., and Müller, K.-R. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 28(11):2660–2673, 11 2017. ISSN 2162-237X. doi: 10.1109/TNNLS.2016.2599820.

Schraagen, J. M., Elsasser, P., Fricke, H., Hof, M., and Ragalmuto, F. Trusting the x in xai: Effects of different types of explanations by a self-driving car on trust, explanation satisfaction and mental models. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 64, pp. 339–343. SAGE Publications Sage CA: Los Angeles, CA, 2020.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shapley, L. S. *17. A value for n-person games*. Princeton University Press, 2016.

Shih, A., Choi, A., and Darwiche, A. A symbolic approach to explaining bayesian network classifiers. *arXiv preprint arXiv:1805.03364*, 2018.

Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pp. 3145–3153. PMLR, 2017.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Sixt, L., Granz, M., and Landgraf, T. When explanations lie: Why many modified bp attributions fail. In *International Conference on Machine Learning*, pp. 9046–9057. PMLR, 2020.

Slack, D., Hilgard, S., Jia, E., Singh, S., and Lakkaraju, H. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 180–186, 2020.

Slack, D., Hilgard, S., Lakkaraju, H., and Singh, S. Counterfactual explanations can be manipulated. *arXiv preprint arXiv:2106.02666*, 2021.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. A. Striving for simplicity: The all convolutional net. In *ICLR (Workshop)*, 2015.

Sundararajan, M. and Najmi, A. The many shapley values for model explanation. In *International Conference on Machine Learning*, pp. 9269–9278. PMLR, 2020.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Vogt, M. D. P. T. Carlo connect, 2019. URL `https://github.com/MoritzDPTV/CarloConnect`.

Wäldchen, S., Macdonald, J., Hauch, S., and Kutyniok, G. The computational complexity of understanding binary classifier decisions. *Journal of Artificial Intelligence Research*, 70:351–387, 2021.

Wäldchen, S., Sharma, K., Zimmer, M., and Pokutta, S. Merlin-arthur classifiers: Formal interpretability with interactive black boxes. *arXiv preprint arXiv:2206.00759*, 2022.

Wang, H., Preuss, M., and Plaat, A. Adaptive warm-start mcts in alphazero-like deep reinforcement learning. *arXiv preprint arXiv:2105.06136*, 2021.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T. (eds.), *Computer Vision – ECCV 2014*, pp. 818–833, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10590-1.

A. Saliency Methods and Off-Manifold Counterfactuals

Considering a classifier function $f : [0,1]^d \to [0,1]$ and an input $\mathbf{x} \in [0,1]^d$, saliency methods attribute importance (or relevance) values to each input feature x_i, with $i \in [d]$, for the classifier decision $f(\mathbf{x})$. In a sense they describe what the classifier focuses on. For a good introduction we refer to (Adebayo et al., 2018). We now argue that the three categories of saliency methods introduced in Section 2 all rely on counter-factual inputs that lie off-manifold.

Local Linearisation For LIME (Ribeiro et al., 2016) this is clear since the method samples new inputs $\mathbf{y}$ around $\mathbf{x}$, labels them $f(\mathbf{y})$ and fits a linear classifier to these new data points. Arguably, gradient-based methods are always off-manifold for highly non-linear classifiers if the gradient itself is not part of the objective function of the training. In this case, there is principally no reason why the gradient should contain any useful information about the classification. The fact that it often does can be explained for models trained via gradient descent, which implicitly enforces useful gradient information. This however, quickly breaks down when the models are manipulated after training, see (Dimanov et al., 2020). Likewise, for piece-wise constant models that are trained by pseudo-gradients the gradient information is always zero.

Heuristic Backpropagation For backpropagation-based methods these counterfactual inputs are less obvious. Lundberg & Lee explain this connection for DeepLift, DeepShap and LRP, which compare the inputs of every layer to baseline values that depend on the specific method (Lundberg & Lee, 2017a).

Partial Input Methods that derive characteristic functions from standard classifiers do this mostly via expectation values (Frye et al., 2020) over a conditional distribution of counterfactual inputs as in Equation (3). However, if the distribution is not modelled correctly, which is difficult for real-world data, it is supported mainly off-manifold. Explanation models that use such a characteristic function, e.g in the form of prime implicants (such as RDE, (Macdonald et al., 2020) or anchors (Ribeiro et al., 2018)) or for Shapley values will inherit this flaw, as explained in (Frye et al., 2020).

B. Description of the Training Process

Our training setup is based on **Algorithm 1** ("PPO, Actor-Critic Style") in (Schulman et al., 2017). This setup was applied to Connect Four as described in (Crespo, 2019), and we adopt most of the hyper-parameters for the training of our agents.

Network Architecture We use a modified version of the architecture proposed by Crespo with two additional fully connected (FC) layers, described in Figure 3. We changed the input dimension to $3 \times 6 \times 7$, representing the fields occupied by the first player (red), the second player (blue) or no one respectively. The input gets transformed by a series of 4 convolutional layers of filter size 3×3 with stride 1, 512 channels and ReLU activations and zero-padding for the first two layers to keep the board shape of 6×7, which is then reduced to 4×5 and 2×3 after the last two conv-layers respectively. The resulting tensor is flattened and passed through a series of FCs with ReLU activation, one of shape 3072×1024, one 1024×512 and three 512×512. Then we split the output into the policy head with an FC of size 512×7 and into the value head with size 512×1. The policy head has a softmax activation function, the value head a $\tanh$ activation.

Training Parameters Our PPO-agent plays against itself and for every turn saves state, value output, policy output, reward and an indicator for the last move of the game. We give a reward of 1 for wins, 0 for draws, -1 for losses and -2 for illegal moves. Illegal moves end the game and only the last turn gets saved to memory. We make use of a discount factor $\gamma = 0.75$ to propagate back reward to obtain discounted rewards for each state. For clipping the policy loss, we set $\epsilon = 0.2$. The total loss weighs the policy loss with 1.0, the value loss with 0.5 and the entropy loss with 0.01. Every 10 games we update the network parameters with Adam on torch standard settings and a learning rate of $l = 0.0001$ for 4 steps.

C. Partial Shapley Values

The Shapley Values are the most established attribution method from cooperative game theory, as they are unique in satisfying the following desirable properties: linearity, symmetry, null player and efficiency (Shapley, 2016). They can be defined as a sum over all possible permutations of the set $[d]$ of d players as follows:

$$\phi_i(\nu) = \frac{1}{d!} \sum_{\pi \in \Pi([d])} \left(\nu(P_i^\pi \cup \{i\}) - \nu(P_i^\pi)\right),$$

where $\Pi([d])$ is the set of all permutations of $[d]$ and P_i^π the set of all features that precede i in the order π.

The size of the coalition P_i^π corresponds to the number of colour features in our Connect Four board states. In our investigation ν is based on the policy layer of the PI-50 agent who has only been trained up to 50% missing information. To avoid off-manifold input we thus want to define partial Shapley values that sample only permutations that ensure a coalition size of at least pd for some $p \in [0,1]$.

To achieve this, we define for every player $i \in [d]$ a set of permutations

$$\Pi_i^p = \{\pi \in \Pi([d]) \text{ s.t. } |P_i^\pi| \geq pd\},$$

and define partial Shapley values as

$$\phi_i^p(\nu) = \frac{1}{d!} \sum_{\pi \in \Pi_i^p} (\nu(P_i^\pi \cup \{i\}) - \nu(P_i^\pi)). \tag{3}$$

Since the (full) Shapley values are the unique attribution method that fulfil the criteria of symmetry, linearity, null player and efficiency, we loose at least one property. We now show that we retain every property except for efficiency.

Symmetry If two players i, j are equivalent, i.e., $\nu(S \cup i) = \nu(S \cup j)$ for all coalitions S that contain neither i nor j, then the symmetry property requires $\phi_i(\nu) = \phi_j(\nu)$.

When choosing a different set of permutations Π_i for each i then this property holds as long as the collection $\{\Pi_i\}_{i=1}^d$ is symmetric in the sense that $\forall i, j \in [d]: \ \Pi_i = \Pi_j[i \leftrightarrow j]$, where $\Pi_j[i \leftrightarrow j]$ means that we exchange the position of the features i and j for every ordering in Π_j. It is easy to see that for our $\{\Pi_i^p\}_{i=1}^d$ this is indeed the case. Thus all terms in Equation (3) are symmetric between i and j and thus the partial Shapley values are symmetric.

Linearity Linearity means that $\forall i \in [d]: \phi_i(\nu + \omega) = \phi_i(\nu) + \phi_i(\omega)$. Since the expression in Equation (3) is still linear in ν, the linearity property remains

Null Player The value $\phi_i(\nu)$ is zero for any null player i, and i is a null player if $\nu(S) = \nu(S \cup \{i\})$ for all coalitions S that do not contain i. This property is trivially true for the partial Shapley values since all summands are zero for a null player.

Efficiency The partial Shapley values are not necessarily efficient anymore. Consider for example the characteristic function

$$\nu(S) = \begin{cases} 1 & |S| \geq pd, \\ 0 & |S| < pd. \end{cases}$$

In this case for every $i \in [d]$ and $\pi \in \Pi_i^p$ we have $\nu(P_i^\pi \cup \{i\}) = \nu(P_i^\pi) = 1$ and thus $\phi_i^p = 0$. In that case $\sum_i \phi_i(\nu) = 0 \neq 1 = \nu([d]) - \nu(\varnothing)$, which is required by the efficiency criterion.

D. The FW-method

To find a small set of colour features that ensure a sensible move from the agent we follow the ideas presented in (Macdonald et al., 2020) and define a rate-distortion functional over a convex set. Our setup is slightly simplified, since we have an advantage in that we can directly deal with partial input without the need to replace the missing features with random variables from a base distribution.

Let $\mathbf{x} \in \{0,1\}^{3\times6\times7}$ be a state describing the game board defined as $[\mathbf{x}^r, \mathbf{x}^r, \mathbf{x}^o]$, where $\mathbf{x}^r, \mathbf{x}^b, \mathbf{x}^o \in \{0,1\}^{6\times7}$ indicate the fields occupied by the red and blue player as well as the open fields respectively, as illustrated in Figure 1. For a continuous mask $\mathbf{m} \in [0,1]^{6\times7}$ that indicates which colour information to show, we define the masked state as $\mathbf{x}[\mathbf{m}] = \left[\mathbf{m} \odot \mathbf{x}^r, \mathbf{m} \odot \mathbf{x}^b, \mathbf{x}^o\right]$, where $\odot$ is element-wise multiplication. Let $a^* = \operatorname{argmax} P(\,\cdot\,; \mathbf{x}[\mathbf{m}])$ be the chosen action by the policy layer, then we define the policy distortion with regards to the mask $\mathbf{m}$ as

$$D^{\text{pol}}(\mathbf{m}) = (P(a^*; \mathbf{x}) - P(a^*; \mathbf{x}[\mathbf{m}]))^2.$$

For a chosen rate of $k \in \mathbb{N}$ we define the optimal mask m^* as

$$\mathbf{m}^* = \operatorname*{argmin}_{\mathbf{m} \in \mathcal{B}_k^{6\times 7}} D^{\text{pol}}(\mathbf{m}), \qquad \text{where} \qquad \mathcal{B}_k^d = \left\{ \mathbf{v} \in [0,1]^d \,\middle|\, \|\mathbf{v}\|_1 \leq k \right\}$$

is the k-sparse polytope (see (Pokutta et al., 2020)) with radius 1 limited to the positive octant. To optimise the objective we use the solver of Pokutta et al. made available at `https://github.com/ZIB-IOL/StochasticFrankWolfe` with 50 iterations. For a given state $\mathbf{s}$ and most likely action a^* the FW-method thus returns $\mathbf{m}^*$ as a saliency map. Oftentimes, multiple m_i^* converge to 1, so we break ties randomly when selecting the most relevant features according to this method.

E. Supplementary Numerical Experiments

In this section we add a number of supplementary experiments that help put the results in the main paper into context and can guide future numerical investigations.

E.1. Shapley Sampling for PI-100

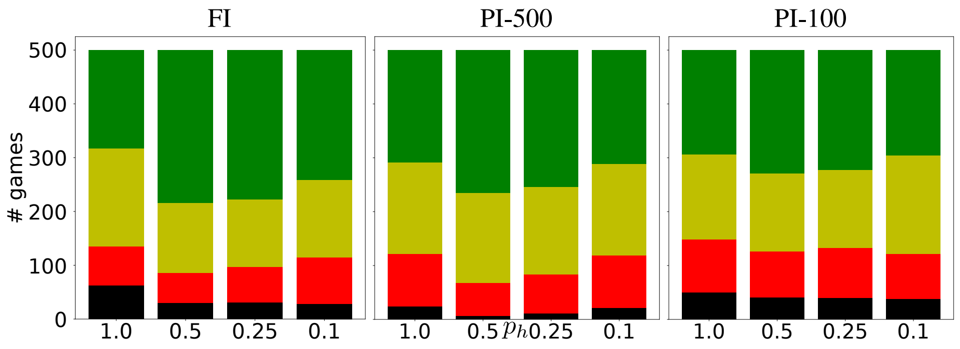


Figure 8. Comparison of Shapley Sampling for our different agents. The bars show how often the method identified the 3 (green), 2 (yellow), 1 (red) or 0 (black) of the three most important game pieces. The method gives the best results (finding the most pieces) for the PI-50 agent.

We compare the Shapley Sampling approach for the ground-truth task described in Section 4.3 for different agents in Figure 8. The method works best for the PI-50 agent with $p_h = 0.5$, presumable because it uses the most capable agent with the largest set of permutations that still ensure staying on-manifold.

E.2. Parameter Search

The methods SmoothGrad and LRP depend on tunable parameters. The SmoothGrad explanation depends on the number of samples that are drawn for a Gaussian distribution around the input value. LRP uses ϵ as a numerical stabilisation term in the denominator, see (Bach et al., 2015). For both values, we used the default value recommended by the Innvestigate package, see `https://github.com/albermax/innvestigate`. We compare different parameters in Figure 9 and confirm a week dependence of the performance.

E.3. Including the Third Channel

To select the most important colour features according to a saliency method we sum the absolute values of the first and second channel for each played piece and select the k pieces with the largest values. The third channel (indicating occupancy) does not factor in. The occupancy information could be relevant because of the piece itself, or just to know where a new piece would end up when inserted in a column. So the importance of the occupancy does not necessarily indicate importance of the colour. In Figure 12 we show the performance of the individual saliency methods if we included the absolute value of the third channel in our sum.

Both GuidedBackprop and DeepTaylor perform somewhat better. The Frank-Wolfe-based saliency attribution performs worse. This is to e expected since now the optimisation includes the third channel which does not correspond to the

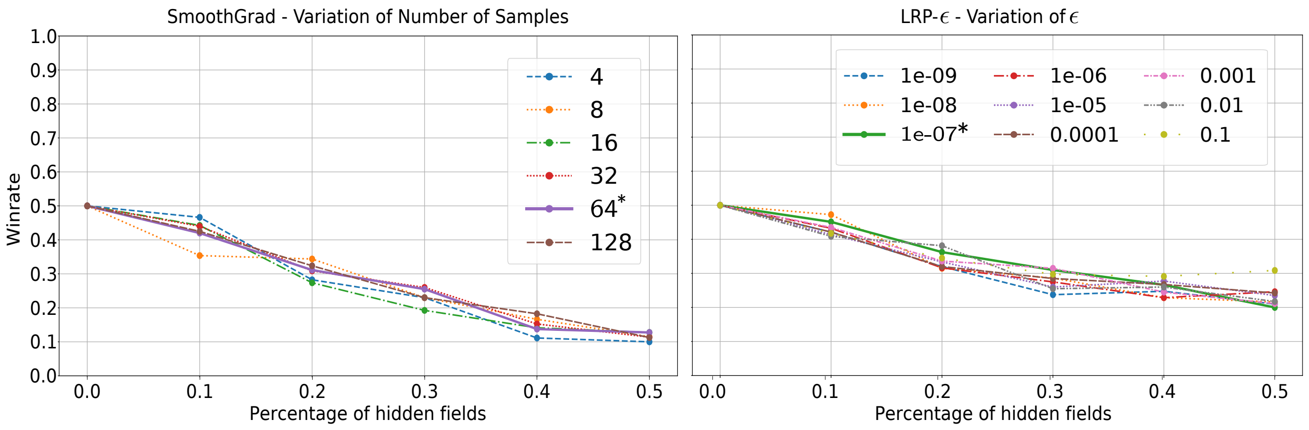


Figure 9. Parameter search for SmoothGrad and LRP-ϵ: For both methods we observe only weak dependence on the parameters. *Left:* Varying the number of samples drawn for SmoothGrad. Section 5. *Right:* Varying the stabilising parameter ϵ for LRP-ϵ.

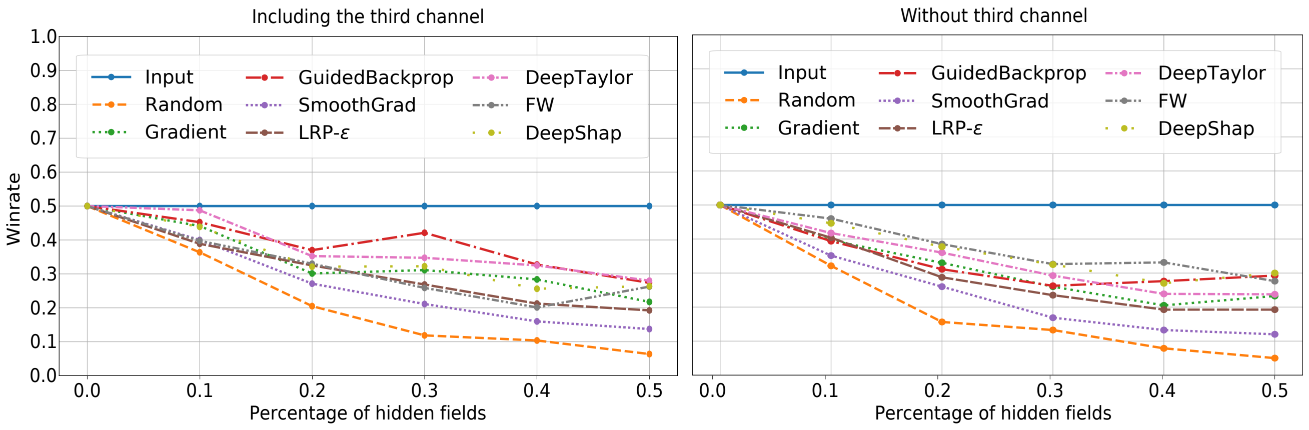


Figure 10. Including the third channel to determine which colour information is selected for each method. We observe that both GuidedBackprop and DeepTaylor perform better, whereas the FW-method decreases in performance. We do not want to put too much importance on this result and include it for the sake of comprehensiveness.

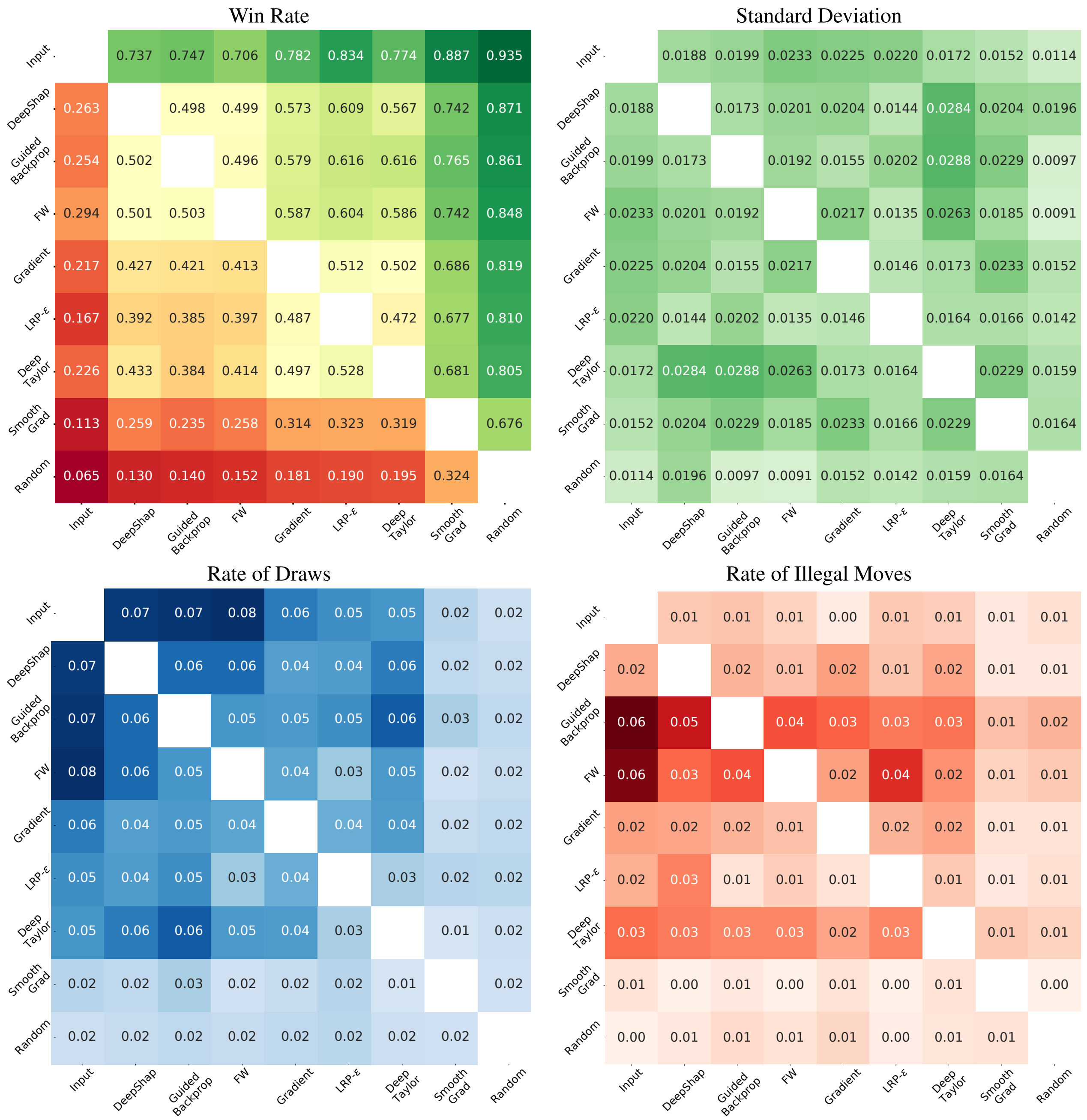
information that is actually hidden. Thus the FW-method might select some pieces where the third channel is relevant for the decision but where the colour information was not important.

E.4. Supplementary Tournament Results

In Figure 11, we again present the win rates in the tournament as in Figure 7 and supplement the standard deviation over 10 training runs, the rate of games ending in a draw as well as the rate of games ending with illegal moves. We see that the standard deviation, rate of draws and rate of illegals is small and do not alter our ranking in Section 5.

E.5. Shapley Sampling

We include the Shapley sampling saliency method with $p_h = 0.5$ described in Section 4.1 and compare it to the other methods. We see that it performs worse than almost all other methods. We give a possible explanation for the bad performance of Shapley sampling in Section 6.



Win Rate

	Input	DeepShap	Guided Backprop	FW	Gradient	LRP-ε	Deep Taylor	Smooth Grad	Random
Input		0.737	0.747	0.706	0.782	0.834	0.774	0.887	0.935
DeepShap	0.263		0.498	0.499	0.573	0.609	0.567	0.742	0.871
Guided Backprop	0.254	0.502		0.496	0.579	0.616	0.616	0.765	0.861
FW	0.294	0.501	0.503		0.587	0.604	0.586	0.742	0.848
Gradient	0.217	0.427	0.421	0.413		0.512	0.502	0.686	0.819
LRP-ε	0.167	0.392	0.385	0.397	0.487		0.472	0.677	0.810
Deep Taylor	0.226	0.433	0.384	0.414	0.497	0.528		0.681	0.805
Smooth Grad	0.113	0.259	0.235	0.258	0.314	0.323	0.319		0.676
Random	0.065	0.130	0.140	0.152	0.181	0.190	0.195	0.324	

Standard Deviation

	Input	DeepShap	Guided Backprop	FW	Gradient	LRP-ε	Deep Taylor	Smooth Grad	Random
Input		0.0188	0.0199	0.0233	0.0225	0.0220	0.0172	0.0152	0.0114
DeepShap	0.0188		0.0173	0.0201	0.0204	0.0144	0.0284	0.0204	0.0196
Guided Backprop	0.0199	0.0173		0.0192	0.0155	0.0202	0.0288	0.0229	0.0097
FW	0.0233	0.0201	0.0192		0.0217	0.0135	0.0263	0.0185	0.0091
Gradient	0.0225	0.0204	0.0155	0.0217		0.0146	0.0173	0.0233	0.0152
LRP-ε	0.0220	0.0144	0.0202	0.0135	0.0146		0.0164	0.0166	0.0142
Deep Taylor	0.0172	0.0284	0.0288	0.0263	0.0173	0.0164		0.0229	0.0159
Smooth Grad	0.0152	0.0204	0.0229	0.0185	0.0233	0.0166	0.0229		0.0164
Random	0.0114	0.0196	0.0097	0.0091	0.0152	0.0142	0.0159	0.0164	

Rate of Draws

	Input	DeepShap	Guided Backprop	FW	Gradient	LRP-ε	Deep Taylor	Smooth Grad	Random
Input		0.07	0.07	0.08	0.06	0.05	0.05	0.02	0.02
DeepShap	0.07		0.06	0.06	0.04	0.04	0.06	0.02	0.02
Guided Backprop	0.07	0.06		0.05	0.05	0.05	0.06	0.03	0.02
FW	0.08	0.06	0.05		0.04	0.03	0.05	0.02	0.02
Gradient	0.06	0.04	0.05	0.04		0.04	0.04	0.02	0.02
LRP-ε	0.05	0.04	0.05	0.03	0.04		0.03	0.02	0.02
Deep Taylor	0.05	0.06	0.06	0.05	0.04	0.03		0.01	0.02
Smooth Grad	0.02	0.02	0.03	0.02	0.02	0.02	0.01		0.02
Random	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	

Rate of Illegal Moves

	Input	DeepShap	Guided Backprop	FW	Gradient	LRP-ε	Deep Taylor	Smooth Grad	Random
Input		0.01	0.01	0.01	0.00	0.01	0.01	0.01	0.01
DeepShap	0.02		0.02	0.01	0.02	0.01	0.02	0.01	0.01
Guided Backprop	0.06	0.05		0.04	0.03	0.03	0.03	0.01	0.02
FW	0.06	0.03	0.04		0.02	0.04	0.02	0.01	0.01
Gradient	0.02	0.02	0.02	0.01		0.02	0.02	0.01	0.01
LRP-ε	0.02	0.03	0.01	0.01	0.01		0.01	0.01	0.01
Deep Taylor	0.03	0.03	0.03	0.03	0.02	0.03		0.01	0.01
Smooth Grad	0.01	0.00	0.01	0.00	0.01	0.00	0.01		0.00
Random	0.00	0.01	0.01	0.01	0.01	0.00	0.01	0.01	

Figure 11. Top Left: Win rate for different pairings in the tournament according to the setup described in Section 5 (same as in Figure 7). *Top Right:* Standard Deviation of the win rate over 10 different training runs. We see generally small standard deviation compared to the mean win rates. *Bottom Left:* Rate of draws for each encounter between saliency methods. *Bottom Right:* Rate of illegal moves by either agent for each encounter.

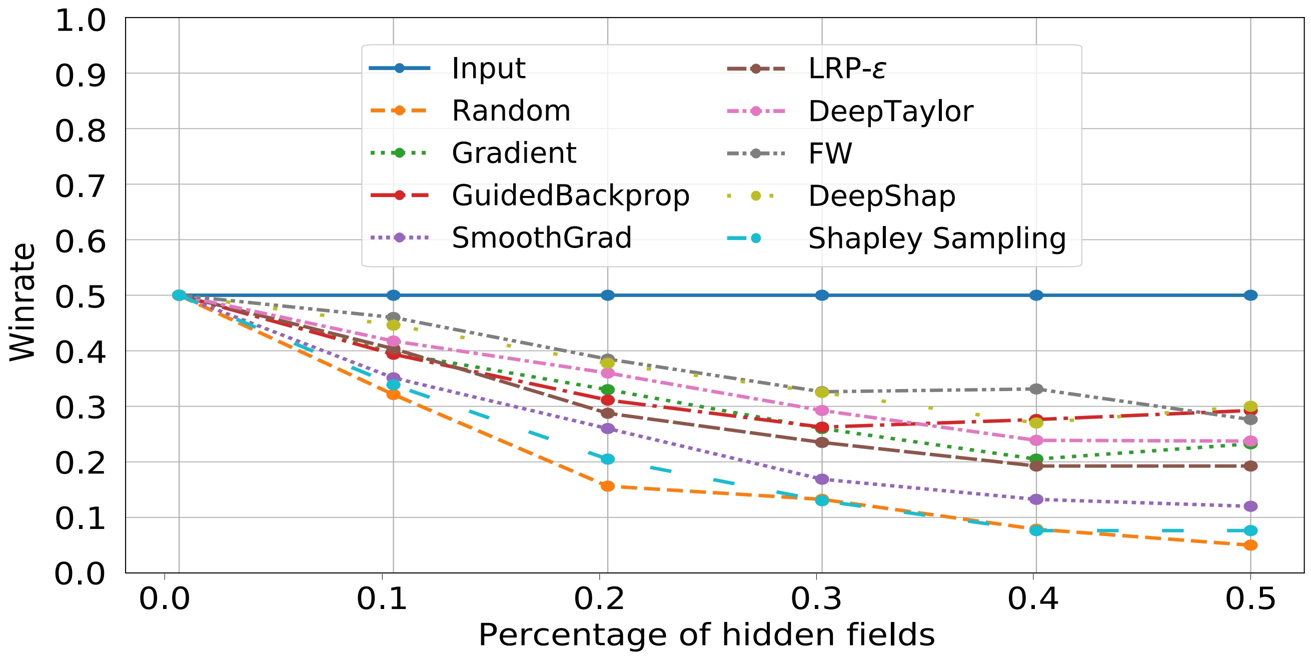


Figure 12. Caption