Learning Machine Learning with a Game

Christoph Lürig
University of Applied Science Trier, Germany
luerig@hochschule-trier.de

Abstract: AIs playing strategic games have always fascinated humans. Specifically, the reinforcement learning technique Alpha Zero (D.Silver, 2016) has gained much attention for its capability to play Go, which was hard to crack problem for AI for a long time. Additionally, we see the rise of explainable AI (xAI), which tries to address the problem that many modern AI decision techniques are black-box approaches and incomprehensible to humans. Combining a board game AI for the relatively simple game Connect-Four with explanation techniques offers the possibility of learning something about an AI's inner workings and the game itself. This paper explains how to combine an Alpha-Zero-based AI with known explanation techniques used in supervised learning. Additionally, we combine this with known visualization approaches for trees. Alpha-Zero combines a neuronal network and a Monte-Carlo-Search-Tree. The approach we present in this paper focuses on two explanations. The first explanation is a dynamic analysis of the evolving situation, primarily based on the tree aspect, and works with a radial tree representation (Yee et al., 2001). The second explanation is a static analysis that tries to identify the relevant situation elements using the Lime (Local Interpretable Model Agnostic Explanations) approach (Christoforos Anagnostopoulos, 2020). This technique focuses primarily on the neuronal network aspect. The straightforward application of Lime towards the Monte-Carlo-Search-Tree approach would be too compute-intensive for interactive applications. We suggest a modification to accommodate search trees and sacrifice the model agnosticism specifically. We use a weighted Lasso-based approach on the different board constellations analyzed in the search tree by the neuronal network to get a final static explanation of the situation. Finally, we visually interpret the resulting linear weights from the Lasso analysis on the game board. The implementation is done in Python using the PyGame library for visualization and interaction implementation. We implemented the neuronal networks with PyTorch and the Lasso analysis with Scikit Learn. This paper provides implementation details on an experimental approach to learning something about a game and how machines learn to play a game.

Keywords: strategy games, reinforcement learning, explainable AI

1. Introduction and related work

AI and precisely machine learning became a field of increasing importance during the last decade (see [Makridakis, 2017]). Machine learning, specifically reinforcement learning (RL), has gathered much attention with the success of Go-playing programs. Many modern AI decision processes though powerful, are unintuitive and unexplainable. This paper aims to implement an explanation technique for the relatively simple board game Connect-Four. The game player can learn something about the applied reinforcement learning technique and the game itself. Therefore, the point of this paper is to implement a combination of explanation techniques for a Connect-Four playing AI. With increasing power in AI, the balancing problem of performance and explainability becomes evident. In the end, a human must burden with the responsibility of the decisions made. [Dosilovic et al., 2018] have given a good overview of this problem and the general approaches to explainable AI. Reinforcement Learning for board games incorporates a mixture of deep neuronal networks and search trees, as [D. Silver, 2016] explains. Silver [Silver et al., 2017] also generalized this approach in the concept of the Alpha-Zero AI to two-person board games of perfect information in general. This paper uses Connect-Four as an experimentation platform to explain the Alpha-Zero AI. The game of Connect-Four is well understood and theoretically solved, as explained by Allis [Allis, 1988]. The game's limited complexity and ease of understanding make it a perfect playground for explainable AI by a developer. It does not require domain-specific knowledge like real-world applications like medicine (see [Rokosna et al., 2020], for instance). The limited search space of that game, especially in comparison with Go, would make this game more suitable for an AI-based on the Alpha-Beta family [Schaeffer, 1989]. On the other hand, the game's reduced complexity makes experimentation with explanation components for an Alpha-Zero-based AI more convenient. The Alpha-Zero AI combines a deep convolutional residual net [He et al., 2015] and a Monte-Carlo-Search-Tree [Browne et al., 2012]. The neuronal net uses two heads, resulting in the current game situation's value and a policy on which actions to take. Both pieces of information are incorporated into the search tree approach to get upper confidence bound estimation of the value of the current situation [Rosin, 2011]. Alpha-Zero falls into the category of RL techniques. A good overview of explanation approaches for RL is given in [Puiutta and Veith, 2020]. The explanation approach we present in this paper would fall into the post-hoc model category with a local scope. The explanation system we implemented falls into two subcomponents: a dynamic and static analysis. The dynamic analysis displays an interactive version of a subtree of the Monte-Carlo search tree. We plot this tree in a radial layout fashion (see [Yee et al., 2001], [Huang et al., 2020]). As several different moves may lead to the same board constellation,

the tree may become a directed acyclic graph. Zobrist style hashing technique [Mason et al., 2005] identifies identical game situations. The main contribution of this paper is the static analysis of the game board. We base the static analysis's core concept on Lime's approach (Local Interpretable Model Agnostic Explanations) [Ribeiro et al., 2016]. This approach has also become popular lately in the simulation community, as explained by [Christoforos Anagnostopoulos, 2020]. The general idea of Lime is to observe changes in the response of a learning system towards a locally perturbed stimulus. A simplified model then describes this response. Picking a linear model with an L_1-based regularization called Lasso [Tibshirani, 1996] is a popular choice in this context.

We analyze the situation by fitting the Lasso model to the value estimation of the Alpha-Zero AI. Doing a Lime analysis for a decision process requires computing the decision process several times. In the context of Connect-Four, this is in the order of magnitude of 100 situations as the board has 42 positions. Considering that one analysis with the search tree may already take longer than 100 ms, this is not an option. The performance problem may become relevant if one wants to apply the approach to more complex games. As a solution, we apply a Lasso analysis toward all relevant board constellations covered by the decision tree and use a weighted Lasso analysis [Bergersen et al., 2011] to get to the final estimation. The probabilities of the situations to occur determines the weights. The trick is that evaluating a neuronal net for several prior known inputs is cheaper than doing many individual evaluations that depend on the tree search results. The independent evaluations can be done on the GPU in parallel (see [Gu et al., 2014]). As the last step, we turn the resulting linear coefficients into a meaningful visualization of the game situation. We implemented the game and the analysis component in Python. For drawing the game board and the tree, we use the library PyGame [Shinners, 2011]. We use PyTorch [Paszke et al., 2019] for the neuronal network part. The Lasso approximation for the Lime analysis is made with Scikit-Learn [Pedregosa et al., 2011]. The remaining of the paper is structured as follows. In section 2, we will discuss the dynamic analysis of the game. Section 3 follows the static analysis of the Lime-based approach. Section 4 shows some results of the two explanation components. Finally, we will discuss some extension possibilities and future work in section 5.

2. Dynamic Analysis

The objective of the dynamic analysis is to display the potential development of the current situation. This analysis is closely related to the Monte-Carlo-Tree-Search (MCTS), one of the Alpha-Zero AI pillars. The tree is expanded node by a node in an iterative fashion. In our case, we use 100 expansion steps. Every MCTS consists of three rules. The first is the node selection for expansion, the second is the leaf evaluation, and the third is the propagation rule up the tree. In the case of Alpha Zero, every edge contains the number of visits and the q-value. The q-value is the long-term value of the corresponding action chosen in that situation. An upper bound confidence selection makes branch selection. The number of visits, the q-values, and the policy determines the upper confidence bound value. The leaves are evaluated either by the final game outcome or the neuronal net.

We evaluate the total number of visits per node and edge for the tree's visualization. The nodes and edges visits estimate the probability of that situation occurring. We use that information to prune the tree to a smaller version as a first step. We eliminate all nodes of a probability smaller than 10%. The tree's nodes and edges bear the player's color that takes a turn in that situation. We visualize the situation's probabilities and the transitions with the thickness of the nodes and edges. We radially display the tree where only a 90-degree angle is covered. The user can select nodes in the graph by picking. We indicated the selected node by rendering a square and displaying the corresponding situation on the left pane. The state value and the q-Values are shown in the player's color in the two headlines above that board. Below we display the transition probabilities. We render the transition probabilities that we have included in the graph in green. The left to right order in the graph corresponds to the slot order on the board. Fig. 3 is an example of the visualization. We took special care of the situation that different sequences of actions may lead to the same situation. Alpha-Zero takes care of this problem by keeping a Zobrist-style hashing table of the game boards. We take care of this by accounting for unique edges in the visualization tree that turn the tree into a directed acyclic graph. We display such a situation in fig. 1.

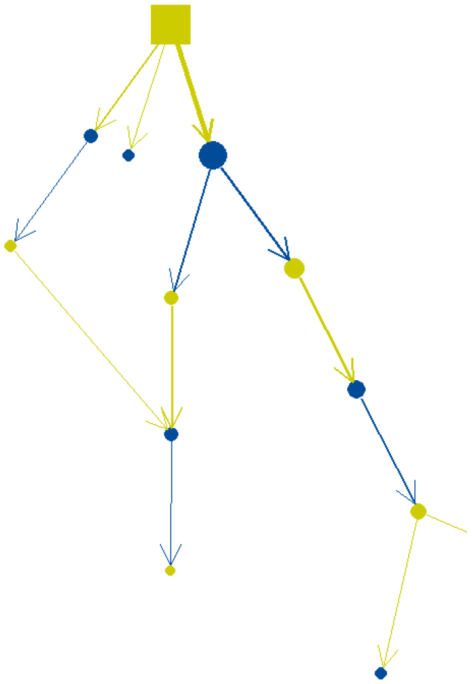


Figure 1: This figure shows an example of a degenerated tree. The same game situation is reached over two different paths.

3. Static Analysis

The static analysis is the main contribution of this paper. This analysis aims to visualize the current and probable future situation elements contributing most to the current situation evaluation. As explained in section 1, we derive the approach mainly from the Local Interpretable Model Agnostic Explanations (Lime) approach. The idea of Lime is to find a simplified model that describes the behavior of a target value in the vicinity of an input value. A summary of the description [Ribeiro et al., 2016] is as follows. Let the model we want to describe be given by $f: \mathbb{R}^d \rightarrow \mathbb{R}$. We are looking for $g \in G$ as an explanation where G is the class of potentially interpretable models. Let $\Omega(g)$ be a measure of complexity. For a linear model, Ω may be the number of non-zero weights. As determining this is an NP-complete problem, an L_1 regularized solution is often suited as an approximation [Tibshirani, 1996]. Let $\pi_x(z)$ be a proximity measure between x and z. Let $L(f, g, \pi_x)$ be a loss function describing how well g approximates f in the locality described by π_x. In total, we try to find $\xi: \mathbb{R} \rightarrow (\mathbb{R}^d \rightarrow \mathbb{R})$.

$$\xi(x) = \underset{g \in G}{argmin}\, L(f, g, \pi_x) + \alpha \cdot \Omega(g) \qquad (1)$$

The α factor appears in the Lasso approach as the L_1regularization factor. In our case, $\alpha = 0.0005$ turned out to be valid. However, two questions arise when we try to apply this to our problem. The first question is the vicinity π_x, and the second question is the computational time needed if we would like to sample the complete Alpha Zero AI with the MCTS system several times.

As for the first question, we have chosen to perturb the existing situation by replacing existing stones with the opposite or no stone. We do the perturbation for exactly one position each. If we have n stones on the current board constellation occupied, we will get 2n variations. We have chosen not to perturb the empty spaces. The player can only place stones on top of the current row. If we would perturb all empty spaces, we would also analyze impossible board configurations. As for stones being placed in the potential future, this issue is solved in combination with the second problem: the computational time needed. The complete Alpha Zero AI evaluation for one constellation takes more than 100ms, rendering the Lime approach to the total solution impossible. Instead, we include the board constellations of every node in our analysis if the probability of this situation occurring is at least 1%. Consequently, we still need to evaluate the neuronal net for our target

objective's values f several times, but the net input values do not depend on each other. The independent evaluation allows for efficient batch processing on the GPU. As we have explained in section 2, we estimate the probability for the situation to occur. We use this probability as a weighting factor in our Lasso analysis. [Bergersen et al., 2011] describe the weighted Lasso analysis. As the last step, we need to visualize the results of the Lime analysis in combination with the game board in a meaningful way. Let us start with the game board.

We can use the fact that once a stone is placed in Connect-Four, it will remain there till the end of the game. Consequently, we can render all the stones of the root node the traditional way. As for the other stones that may be placed in the future, we render only those stones that the Lasso algorithm has selected by assigning non-zero linear weights. We build the average of the eventual future game boards weighted by their probability of happening for those positions. This probability gets mapped to the saturation of the stone color. A white circle is included in the visualization to mark this explicitly as a potential future stone. As a next step, we visualize the linear regression coefficients of the Lasso analysis. The individual coefficient gets multiplied with the stone index −1 for hostile stone and 1 for one's stone. A positive regression coefficient for the respective stone position means that it would be good if that stone would be the player's stone, and a negative would mean it would be better if it were a hostile one. If the regression coefficients and the stone match in signs, it is suitable for the current player and otherwise deficient. As an input for the visualization, we can use the product of the linear regression coefficient and the stone coefficient. The current value of the situation is displayed on top of the game board. Sample visualizations can be found in fig. 4 and fig. 5. Fig. 2 summarizes the Lime-based static analysis's overall data flow. We start with the annotated tree, which results from the Alpha-Zero AI. Here we extract all relevant future situations with their probability. This information goes directly into the Lime Analysis to perturb all different game boards and the weighted board's averager. The probabilities of future boards and the perturbation results go into the Lasso analysis. The results of the Lasso analysis form in the combination of the average board and the visualization elements. The weighted average board itself generates the image of current and potential future stones. The Lasso component filters the potential stones.

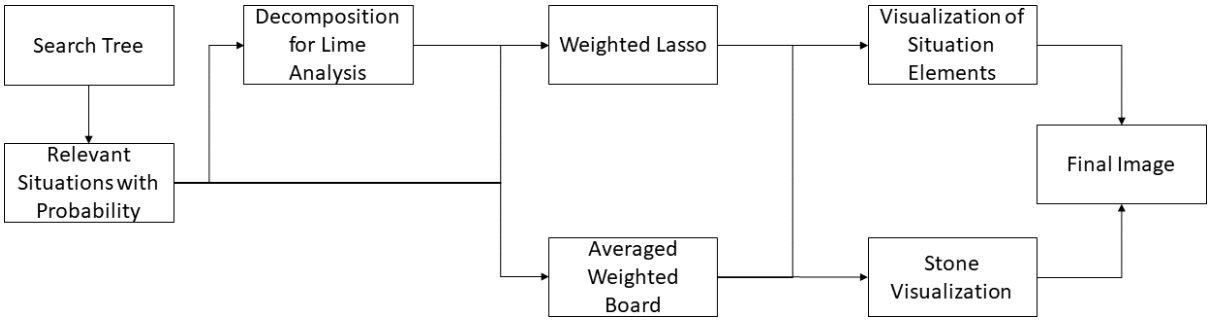


Figure 2: The data flow of the static analysis.

4. Results

This section discusses some game situations of Connect-Four and shows how the interpretation results are visualized. As explained in the previous two sections, we have a dynamic and static analysis. The dynamic analysis focuses on the potential development of the current game situation. The static analysis shows how the different situation elements contribute to the overall evaluation of the situation.

We begin the discussion with the dynamic analysis. Fig. 3 shows the game layout with the board game itself on the left side and the decision tree visualization on the right side. The picture shows the decision tree in the case of the dynamic analysis. We apply the dynamic analysis before we make a move. The tree's root node represents the game situation before the move is made. In the shown picture, we have already selected a very late leaf. One can see this at the small square on the lower right branch. Next, we display the q-values and the current value of the situation. The q-values are the values from the corresponding move of the slot. They represent the expected outcome of the game if the corresponding move is made. The value of the situation is the maximum of the q-values as the player is likely to pick the best of all options. We print all values in the current player's color, which is yellow in this case. The probability of the move to be picked is shown in the line below. The probabilities are generated by the policy of the Alpha zero algorithm. The option with the highest probability is written in green, the others in red. The color of the graph node visualizes the player that is taking turns at that point. The line thickness between nodes visualizes the transition probabilities. The user can pick different nodes in the graph to see how the AI judges the most likely development of the situation.

Fig. 4 shows the static analysis, which does not use the right pane. The static analysis contains stone positions that are already placed and positions that are likely to be placed in the future. We render a white dot into the center of stones that are likely to appear at this position in the future. The more likely it is, the more saturated the color. We derive the likelihood of stone positions from the search tree. Red and green rings around the stones visualize the contribution to the overall situation. Green means that it contributes positively to the situation, and red negatively. The color is determined by the computed coefficients of the Lasso analysis explained in section 3.

Fig. 3 and 4 show the same situation. The first figure is the dynamic, and the second figure is the static analysis. Here one can see the correspondence of the late dynamic situation with the future stones in the static analysis. We selected a relatively late move in the dynamic analysis, as indicated by the small square in the middle leaf. Suppose one analyzes the single moves made along the way more precisely. In that case, one will find that the intensity of the uncertain stones' saturation in fig. 4 decreases the further one goes into the future. Naturally, the certainty of the prediction decreases the further one goes into the future. The static analysis shows a gap between the existing yellow and potential future blue stone on the right side. This probably yellow stone in between is not included in the visualization of fig. 4 because it does not contribute to evaluating the situation. As this is an early stage of the game, naturally, all own stones contribute positively to the situation and all opponent stones negatively. Both fig. 4 and 3 show the value of the situation on top of the board. They differ because, in the dynamic analysis (fig. 3), the value is derived from the situation flagged with the box in the tree and all following potential situations. The static analysis estimation (fig. 4) refers to the game situation marked as the second node from the root in the dynamic analysis. In consequence, it covers more situations and information.

Fig. 5 shows a static analysis of the same game in one of its early stages (fig. 5a) and near the very end of the game (fig. 5b). In the situation depicted in fig. 5bit is blues turn to drop the next stone. Wherever blue will go, yellow will in the next move. This move is flagged by marking the potential blue stones in the empty slots with a green circle. Even if this is an opponent stone, the future placement is flagged as something positive as this is a prerequisite for yellow to finish the game with a winning move. The two central yellow stones are marked with the light green circle in fig. 5b are both parts of one of the potential winning constellations of connected four yellow stones that may arise from the current situation. Those same stones are already marked in the early-stage analysis in fig. 5a. However, the yellow stone in the bottom center has lost its strategic value over time. The loss of strategic value is visible if we compare the saturation of the green between fig. 5a and fig. 5b.

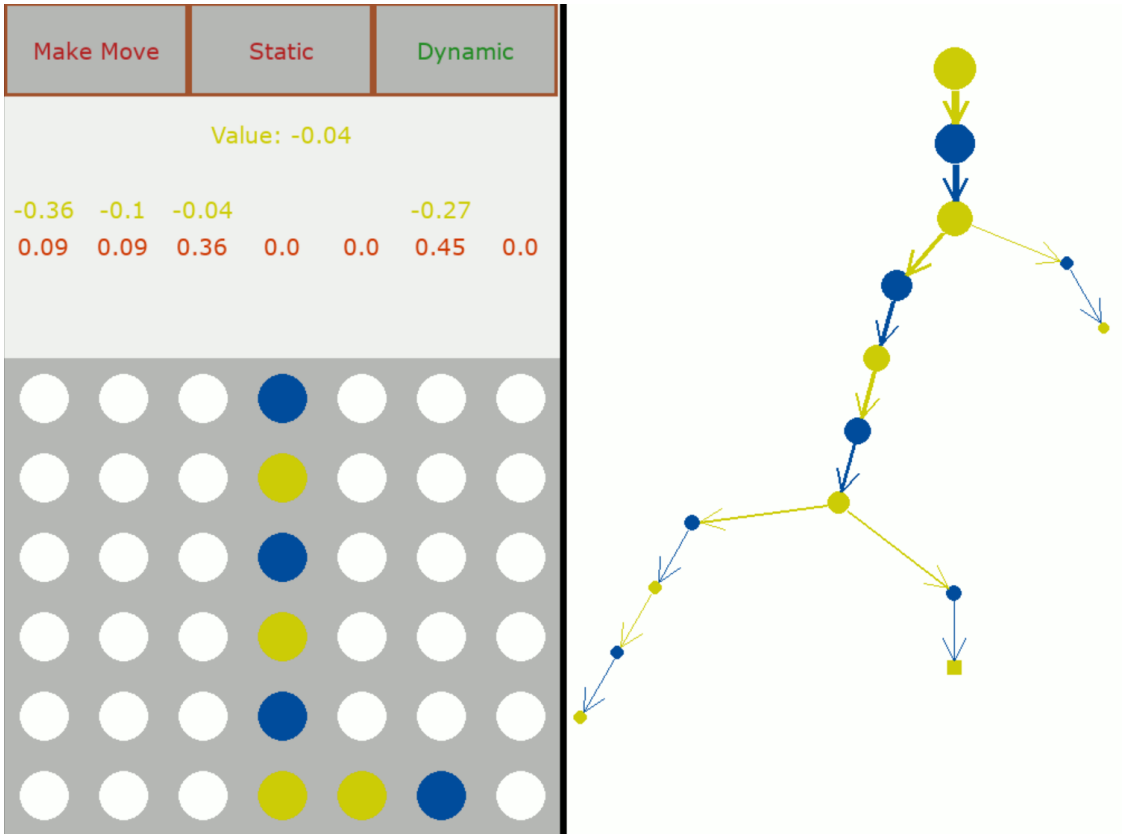


Figure 3: The dynamic analysis of the initial moves. The image corresponds to the static analysis shown in fig. 4

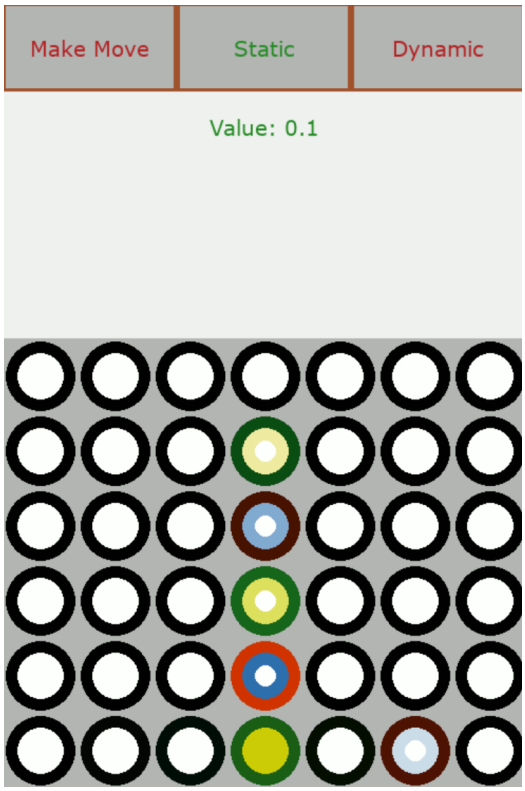
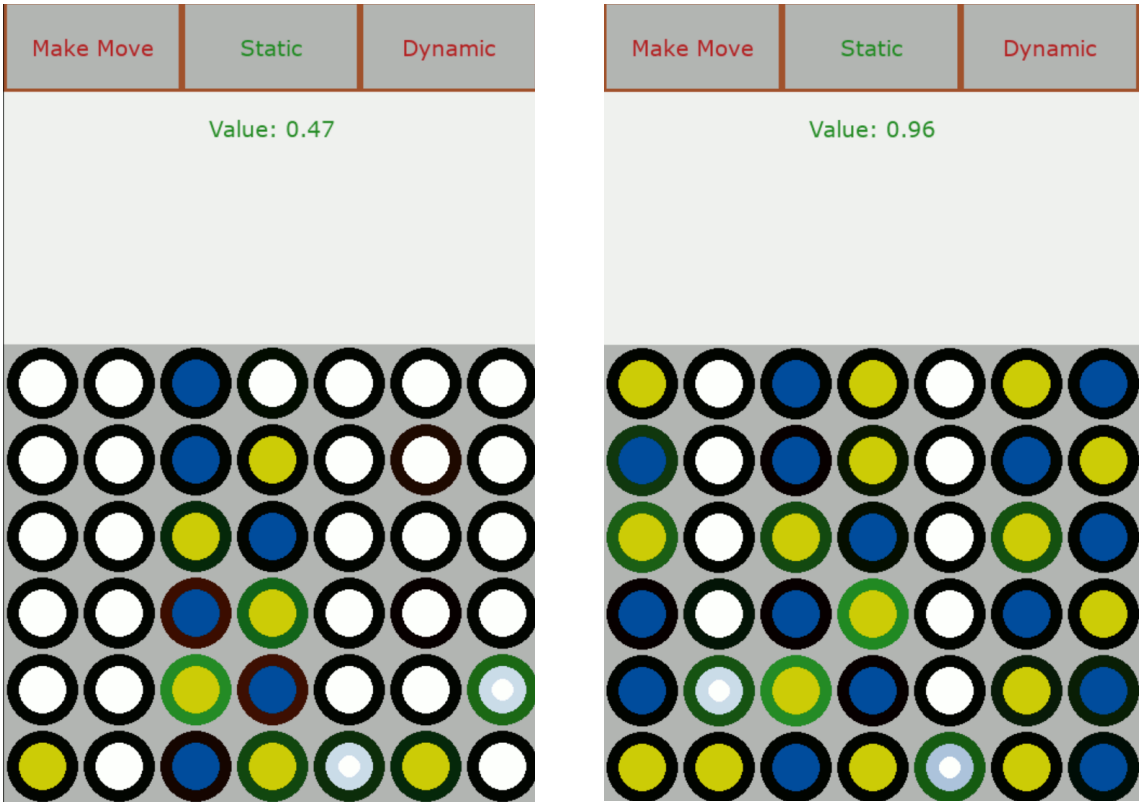


Figure 4: The static analysis of the initial move. The image corresponds to the dynamic analysis shown in fig.3



(a) Static analysis at the beginning

(b): Static analysis at the end

Figure 5: Static analysis

5. Discussion and Future Work

In this paper, we have shown how to implement an explanation technique for a board game so that a player can learn something about the game and the applied AI Alpha-Zero. The explanation technique contains a static and dynamic analysis. We have shown that a static and a dynamic analysis complement each other. Both systems derive information from the neuronal net and the search tree. The dynamic analysis is based on the search tree and the static more on the neuronal nets. The focus of this paper is the static analysis. We made Lime practically applicable to the Alpha-Zero AI by constructing a weighted Lasso analysis of the current probable future situations. This variation of Lime enables us for efficient calculation on the GPU (graphics processing unit) and meaningful visualization of the current situation.

xAI (explainable artificial intelligence) also incorporates visualization, computer-human interfaces, and domain-specific knowledge. With this respect, games in general and board games specifically are a convenient domain for experimentation. Games have a limited knowledge domain and are easy to understand by most people. Understandability is a significant advantage over application domains like medical and engineering subjects that require the cooperation of a domain expert. As the next step, we apply the general approach to the game of Othello. Othello is a relatively easy game to understand but has, compared to board games, few active players. Consequently, we only have a slight variance in the test group that we will use for user testing.

References

Allis, V. (1988). A knowledge-based approach of connect-four. the game is solved: white wins. Master's thesis, Department of Mathematics and Computer Science, Vrije Universiteit, Amsterdam.

Bergersen, L., Glad, I., and Lyng, H. (2011). Weighted lasso with data integration. Statistical Applications in Genetics and Molecular Biology, 10:39–39.

Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P. I., Tavener, S., Perez, D., Samothrakis, S., Colton, S., and et al. (2012). A survey of Monte Carlo tree search methods. IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI.

Christoforos Anagnostopoulos, S. R. (2020). Trusting a black box: explaining complex simulation outcomes using Lime. Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC).

Dosilovic, F. K., Brcic, M., and Hlupic, N. (2018). Explainable artificial intelligence: A survey. In 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pages 0210–0215.

D.Silver, A. (2016). Mastering the game of go with deep neural networks and tree search. Nature.

Gu, J., Zhu, M., Zhou, Z., Zhang, F., Lin, Z., Zhang, Q., and Breternitz, M. (2014). Implementation and evaluation of deep neural networks (dnn) on mainstream heterogeneous systems. APSys '14: Proceedings of 5th Asia-Pacific Workshop on Systems.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. CoRR, abs/1512.03385.

Huang, G., Li, Y., Tan, X., Tan, Y., and Lu, X. (2020). Planet: A radial layout algorithm for network visualization. Physica A: Statistical Mechanics and its Applications, 539:122948.

Makridakis, S. (2017). The forthcoming artificial intelligence (ai) revolution: Its impact on society and firms. Futures, 90:46–60.

Mason, D., Hudson, T., and Sutton, A. (2005). Fast recall of state-history in kinetic Monte Carlo simulations utilizing the Zobrist key. Computer Physics Communications, 165(1):37–48.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai,

J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay,

E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830.

Puiutta, E. and Veith, E. M. (2020). Explainable reinforcement learning: A survey.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. CoRR, abs/1602.04938.

Rokosna, J., Babic, F., Trtica-Majnaric, L., and Pusztova, L.(2020). Cooperation between data analysts and medical experts: A case study. In CD-MAKE.

Rosin, C. D. (2011). Multi-armed bandits with episode context. Ann. Math. Artif. Intell., 61(3):203–230.

Schaeffer, J. (1989). The history heuristic and alpha-beta search enhancements in practice. IEEE Transactions on Pattern Analysis and Machine Intelligence, 11:1203–1212.

Shinners, P. (2011). Pygame. http://pygame.org/.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M.,Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T. P., Simonyan, K., and Hassabis, D. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. CoRR, abs/1712.01815.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological), 58(1):267–288.

Yee, K.-P., Fisher, D., Dhamija, R., and Hearst, M. (2001). Animated exploration of dynamic graphs with radial layout. Proc. Information Visualization