Toward Comparing Knowledge Acquisition in DeepRL Models

Anthony Marchiafava, Atriya Sen
Department of Computer Science
University of New Orleans
New Orleans, United States
email: amarchia@uno.edu, asen@uno.edu

***Abstract*—In order to better exploit Deep Reinforcement Learning (DeepRL) systems such as DeepMind's Alpha Go & Alpha Zero, it is desirable to understand how they acquire knowledge, and how human knowledge acquisition can contribute to or benefit from such an understanding. We analyze a series of DeepRL models trained to play the board game of chess in a human-like fashion, to study if these models acquire concepts differently from self-trained DeepRL models such as AlphaZero. Our preliminary results indicate that human chess players may acquire concepts very similarly to self-trained models. We further discuss some of the potential consequences of such an outcome.**

Keywords-Artificial Intelligence; Deep Reinforcement Learning; Reinforcement Learning; Deep Learning; Explainable AI.

I. INTRODUCTION

The game of chess has been called the "drosophila" of artificial intelligence (AI), referring to the extensive use of fruit flies (*drosophila*) in experimental biology. While traditional chess engines rely primarily on tree search with advanced heuristics, many modern approaches have exploited deep learning or deep reinforcement learning.

One such recent project is AlphaGo Zero, which uses a combination of Monte Carlo Tree Search (MCTS) and a deep neural network [1]. Leela Chess Zero is an open-source implementation of both the MCTS and the (convolutional) neural network of AlphaGo Zero, and achieves a similar level of performance (i.e., playing strength), which is to say, a superhuman level: capable of consistently defeating any known human player.

It may be reasonably hypothesized that such neural systems are learning implicit knowledge about chess-playing concepts and strategies. Understanding the internal knowledge acquisition processes of these and similar systems have the potential to provide insight into both chess as a game and the application of a similar process to varied adversarial domains, such as international trade, nuclear deterrence, and other negotiations.

The MCTS algorithm is used to examine the possible outcomes of the game depending on which move is chosen, by searching through trees generated from different choices the player could make, and examining which ones lead to the highest probability of winning [2]. These trees are generated by the deep neural network.

The deep neural network is fundamentally a two-state regression or classification model which accepts some input and produces one or more outputs [3]. The network will accept the input and produce derived features, which are then used to produce further derived features depending on network depth, and derived features are combined using an output function to produce the final output. Derived features are produced using linear combinations of the inputs and activation functions and other operations at different layers of the network. The first few layers more closely match the structure of the initial input, but as further derived features are generated, the derived features become more and more abstract.

The deep neural network accepts the current state of the chess board, prior states of the chess board, including a number of additional game-specific parameters such as the current castling status, and finally move count as input, and produces two outputs via dual network heads: (1) the policy head, which produces the probability distribution of possible moves, and (2) the value head, which produces the predicted outcome of the game, based on making the suggested move, as a win, lose, or draw. The MCTS uses the output of the neural network to choose the best candidate move. AlphaGo Zero learnt to play chess without exposure to human moves or more abstractly, playing styles, and generated implicitly expressed strategies sophisticated enough that it prevailed in a multi-game match against Stockfish, then a traditional search-based engine (Stockfish has now been updated to additionally use a neural model).

In an effort produce engines that behave more human-like at a variety of skill levels, Maia Chess was created [4]. Different versions of Maia were trained on specific games of human players at different skill levels, in lieu of using self-play, effectively training the neural network in human-style play. The different versions of Maia were able to produce gameplay choices similar to human players from 1100 ELO to 1900 ELO, where ELO refers to the ELO rating system, which is used almost exclusively in chess, and refers to a relative ranking of a particular player's odds of winning against another player of a different skill level (i.e., ELO rating). Maia was built on the Leela Chess Zero framework, an open-source engine inspired by Alpha Zero. However

Maia does not use an MCTS, but rather uses a deep learning model exclusively.

We will show how we can detect and compare the concepts that the various versions of Maia use. In section 2 we indicate what motivated our work here and what similar work was done in the past. In section 3 we show the technique we used to get concepts and how they are detected. In section 4, we show the results we were able to generate. In section 5 we show what further work can be done in the future.

II. MOTIVATION & PRIOR WORK

Since deep neural networks (DNNs) are inherently black boxes, the ability to understand and explain the presumed acquisition of concepts and strategies by the network in chess and other adversarial domains is highly desirable for a variety of reasons, including training humans in "superhuman" strategies, and interpreting them in terms of human strategies. A DNN learns derived features generated using the backpropagation algorithm, which updates intermediate node weights based on the gradient between the observed output and the expected output at the final layer of the network, a process which is not directly human-interpretable.

One interpretability technique is to use the technique of *linear probing* to examine the detectability of concepts at the intermediate layers of the neural network, and the acquisition of knowledge those concepts entail [5]. This approach is derived from a technique for detecting image concepts in computer vision using *concept activation vectors* (CAVs) [6]. We separate examples of game states which have some concept in common, and examples which do not exhibit that concept. These classifications are matched to the activations of a particular layer in a neural network, whose input matches our game state. We then train a linear classifier to differentiate between the inputs of the two classes. This allows us to detect if the set of activations of a particular layer for a particular network contain the information needed to determine if a concept is present or absent at that layer. This has been the approach taken in [7], for interpreting concepts learnt by Alpha Zero.

In this paper we present the results of a similar examination using linear probing, to compare the behavior of concept and strategy knowledge acquisition across various versions of weights learnt by the Maia network, to compare the concept acquisition of a model trained on human play against a model trained by self-play. Our preliminary results indicate that human chess players may acquire concepts very similarly to self-trained models.

III. TECHNIQUE AND PROCESS

To understand the concepts we will compare we must detect the concepts, preprocess our input data to be interpretable by the modified version of the network we need to use, and get the activations from the intermediate layers we examine.

A. Concepts

The concepts we tested for in the DNN's chess understanding were material advantage and a modified version of material advantage from the perspective of the player with the white pieces. The concept of material advantage is defined by adding up the number of pieces one player has remaining on the board, adjusted by the assumed inherent value of those pieces, and subtracting the value of the other player's pieces. A pawn is worth 100 points, a knight is worth 320 points, a bishop is worth 330 points, a rook is worth 500 points, and a queen is worth 900 points. So, a player with three pawns and one queen is worth 1200 points and a player with only two rooks is worth 1000 points, so there would be a 200-point advantage for the first player over the second. The king is not assigned a material value, since losing the king is not possible in chess.

The second concept includes the previously mentioned material advantage modified by an increased weight for pieces in more advantageous positions and a penalty for disadvantageous positions. The weights of these positions are defined by a piece-square table. Each piece-square table is an 8x8 array of numbers where each defines a modifier for the *quality* of each piece in that position, referring to the postulated long-term strategic advantage or disadvantage of a piece being in that position.

We created a unique piece-square table for each type of piece. These piece-square tables are each oriented towards whoever is the player whose board position is being evaluated. We used publicly available human-play ranked games from the online chess platform Lichess to generate the game states, to generate the game states over which to check for the two material advantage concepts. Lichess is a popular platform and has many years of games to draw upon. The Lichess games were also in same format of the games which were used to train the Maia networks used, the Portable Game Notation (PGN) format, used to notate each move made by either player over the course of a single game. Combined with knowledge about chess boards and game states, PGN files are sufficient to generate every game state occurring over the course of a game. The files also included the metadata about the players, including their ELO ratings.

B. Preprocessing

We used the same tools used to generate the Maia training data, to create a dataset of 204,800 sample game states. First, we separated games by ELO using pgn-extract [8], a tool for extracting games using portable game notation formatted games. This allowed us to remove games which may have been trained on already, and allowed us to evaluate games which were not in the training dataset for a particular version of Maia. These games were then converted into a format suitable for providing the Leela Chess Zero (Lc0) neural network using trainingdata-tool [9], which is designed to convert from PGN games to the Lc0 format. These are stored in binary files which are not human-readable. Since each game was entirely converted into a series of inputs - one input for each move in the game - we also needed to know which game state corresponded to which concept.

Therefore, we converted each given input back into a format which could be evaluated for material advantage and modified material advantage. This provided both the input in the correct format and a more easily interpretable version that we maintained as linked to each other.

C. Activation Layers as Input

For each activation layer and Maia version we wished to examine, we then generated the activations of the neural network up to that layer and stored those activation values. Examining 19 activation values for the version of Maia trained to behave like a player in the 1200-1299 ELO range created examples of what concepts the network could detect at each of those hidden layers, in the samples provided.

We then created a new DNN model whose input was the original input and whose output was the activation values of the layer we wished to examine in the original network, creating 19 sets of activations for each input. These were then used as input to a classifier whose output was the presence or absence of the material advantage concept. We then trained a classifier to determine if a layer's activation was correctly classified. That is, for a game state which shows a material advantage for the active player, that game state when converted to an input should produce layer activations which can be classified correctly if that layer includes the information that the concept classification requires.

IV. RESULTS

We examined the odd numbered activation layers for both the concepts previously mentioned, across the three ELO categories of 1200, 1400, and 1900, for both simple material advantage and material advantage incorporating the weights found in the piece-square tables. The more basic material advantage was detectable with roughly 73% accuracy across the three categories and across all the activations examined. To evaluate this, we split our classifier data into a training set of 200,000 samples and a testing set of 4,800 samples.

TABLE I. MATERIAL ADVANTAGE

Maia	Activation Layer Classification Accuracy		
	Activation_1	***Activation_9***	***Activation_19***
1200	0.737708	0.737916667	0.738958
1400	0.738542	0.738125	0.738125
1900	0.737708	0.7375	0.738333

TABLE II. MODIFIED MATERIAL ADVANTAGE

Maia	Activation Layer Classification Accuracy		
	Activation_1	***Activation_9***	***Activation_19***
1200	0.534167	0.550625	0.529792
1400	0.537083	0.544375	0.544375
1900	0.535208	0.546041667	0.543958

Comparable results were obtained from an examination of AlphaGo Zero in [7], the conclusion being that material advantage as a concept is relatively easy to detect, even from the inputs without activations, and provides a good baseline to evaluate the concept detection system. Each version of Maia was fully capable of detecting the concept to a similar degree: we can conclude that this concept is not sufficiently different across the different ELO categories of Maia models.

The results of examining for modified material advantage show our technique to be less accurate. This may be because our modified version of material advantage is not sufficiently aligned with a concept that any version of Maia is looking for. There may be some weighted version of material advantage that Maia may use, but the specific concept we attempted to detect does not appear to be one used by Maia. This indicates that it is necessary to explore other concepts to further understand the different behaviors of the Maia models.

V. CONCLUSION AND FURTHER WORK

The specific domain concepts examined here represent a proof of concept of our strategy. Since the accuracy of each version of Maia is similar across the ELO ranges used, other more subtle concepts may be more effective at showing the differences between the human-trained models. Or, if the concept detection is the same across all versions of Maia for most concepts, further work is necessary to understand the difference in behavior but similarity in concept detection. If, for example, the data necessary to detect a particular concept differs between versions of Maia or Lc0, then we can say that part of that concept is potentially used in differentiating the final behavior.

A more thorough examination of the behavior of a self-trained model which exactly uses the Maia network's structure would be additionally worth comparing to, as the default Leela Chess Zero weights did not match with the version used by Maia. Further work on comparing a self-play trained model such as Lc0 to one trained entirely on human generated data such as Maia, may show novel rationale for the difference in quality and behavior between these systems.

REFERENCES

[1] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140-1144, 2018.

[2] M. Świechowski, K. Godlewski, B. Sawicki and J. Mańdziuk, "Monte Carlo Tree Search: A Review of Recent Modifications and Applications," arXiv, 2021.

[3] T. Hastie, R. Tibshirani and J. Friedman, The Elements of Statistical Learning, New York: Springer New York Inc, 2001.

[4] R. McIlroy-Young, S. Sen, J. Kleinberg and A. Anderson, "Aligning Superhuman AI with Human Behavior: Chess as a Model System," in 2020 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2020), 2020.

[5] A. Guillaume and Y. Bengio, Understanding intermediate layers using linear classifier probes, arXiv, 2016.

[6] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas and R. Sayres, "Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)," arXiv, 2017.

[7] T. McGrath, A. Kapishnikov, N. Tomašev, A. Pearce, D. Hassabis, B. Kim, U. Paquet and V. Kramnik, "Acquisition of Chess Knowledge in AlphaZero," arXiv, 2021.

[8] https://www.cs.kent.ac.uk/people/staff/djb/pgn-extract/, last accessed July 12th, 2022.

[9] https://github.com/DanielUranga/trainingdata-tool, last accessed July 12th, 2022.