Playing for More than the Win

Susan L. Epstein

Hunter College and The Graduate Center of The City University of New York, 695 Park Avenue, New York, NY, USA

Abstract

In the drive to develop more powerful systems, AI practitioners sometimes forget that what they build is motivated by human needs and intended ultimately to serve some human purpose. This paper traces how decades of development have led to superhuman computer game players. It also demonstrates how AI's drive for skilled behavior can ignore significant aspects of intelligence and thereby lose functionality important to society.

Keywords

intelligence, symbolic and subsymbolic representation, knowledge, reasoning, learning, planning, memory, analogy, collaboration

1. Introduction

The premise of this paper is that artificial intelligence requires more than highly-skilled decision making. Game playing serves here as a lens through which to examine AI's development of superhuman performance on a set of challenging problems. It clarifies what such systems have lost in the drive to achieve their skill, and identifies to what AI must still aspire.

The formal study of AI began with the conjecture that

> ...every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. [1]

In this excerpt from the 1955 proposal to fund the Dartmouth Summer Research Project on Artificial Intelligence, "emulate" would have required AI to replicate how humans solve problems. Instead, "simulate" gave researchers license to create a black box, a system with specified input and output but without processing specifications. Although these innovative scientists intended to share what they knew with a computer, "learning" clearly indicates that they also expected the machine to discover some knowledge on its own.

There is a difference between data and knowledge. Data is merely statements about experience in the world: measurements, objects, events. In contrast, knowledge explains how data came to be, that is, how some part of the world works to produce it. From data, learning acquires and applies knowledge. In both humans and machines, however, learning about the world is primarily inductive. Without data that encompasses every possibility, any learner can only

AAAI 2022 FALL SYMPOSIUM SERIES, Thinking Fast and Slow and Other Cognitive Theories in AI, November 17-19, Westin Arlington Gateway in Arlington, Virginia, USA

susan.epstein@hunter.cuny.edu (S. L. Epstein)

http://www.cs.hunter.cuny.edu/~epstein// (S. L. Epstein)

0000-0002-9618-9228 (S. L. Epstein)

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

CEUR Workshop Proceedings (CEUR-WS.org)

CEUR-WS.org/Vol-3332/paper12.pdf

generalize over what it has experienced. Nonetheless, a learner that can build a faithful world model from its own experience, without instruction from humans, is impressive, particularly when that model outperforms human experts.

Although AI systems increasingly succeed on problems once thought beyond their grasp, human trust in those systems has rapidly declined. The recent flurry of calls to hold AI accountable for its impact on humans is a response in part to those systems' opacity. The issue is *explainability*, that is, how an AI system can reveal its processes and clarify its answers to support human understanding.

The key to explainability is *knowledge representation*, both how humans describe the problem to an AI system and what the system manipulates to reason and learn about that problem. Knowledge representation can be symbolic or subsymbolic. A *symbolic* representation is necessarily transparent to its designers and made more accessible to others through natural language and visualization. In contrast, a *subsymbolic* representation supports a computer's fast processing and frugal storage but is often data-hungry and opaque. Recent, widely publicized subsymbolic systems have explained why a joke is funny [2], created pictures to match verbal descriptions [3], and predicted proteins' three-dimensional structures with atomic accuracy [4]. These programs, however, all employ what has been called "thinking fast" (subsymbolically) rather than "thinking slow" (symbolically) [5].

Because games have challenged humans for centuries and intelligent behavior is often gauged by competition, board games were a natural and early AI target. A *game* here is an activity on a finite board where two players act with tokens (*move*). At any moment, a game's *state* indicates where each token is located and whose turn it is to move. A game's rules specify its *initial state* before any player has moved, and the moves permitted in each possible state. They also specify which states are *terminal* (end play) and the *outcome* of a contest based on its terminal state: win, loss, or draw from the perspective of the first player. (For clarity, "contest" here describes one full experience at a game, from the initial state to a terminal state.) Both players know the rules and have full and accurate (*perfect*) information about the current state. Furthermore, every move is deterministic, that is, produces a single new state that is known in advance. A game's *state space* describes how moves transition play from one state to the next. *Search* is organized movement through state space.

The competitive nature of games and the huge state space that challenging games engender have forced multiple compromises in the development of expert game-playing programs. The first compromise in their development was to abandon requirements for perfect decisions. Perfect play is intractable in a challenging game because there are too many states to examine. Given such a game, AI researchers initially sought to *strongly solve* it, that is, to produce an algorithm that would achieve an optimal outcome in every contest, whether or not the algorithm made the first move and regardless of the quality of both players' moves. Tic-tac-toe, for example, has so few possible states that it is trivial to produce a strong solver. To date, however, few much larger games have been strongly solved. Indeed, an announcement that a game has been "solved" typically means only that it has been *weakly solved*, that is, there is now an algorithm that always wins or draws for one player, against any moves by its opponent [6, 7]. Checkers, for example, has been only weakly solved [8].

Another compromise was to accept expert behavior as an ideal. In game playing, data is simply recorded experience, such as a set of *training examples* of the form (*state, move*), where

the player whose turn it was in *state* executed *move*. The *ground truth* about a training example is whether or not *move* was indeed optimal from *state*. Because in challenging games the ground truth may be unknown, the most valued training examples were collected from contests between top human experts. A dataset of such training examples does not guarantee that it represents optimal play, nor does it indicate how individual training examples relate to one another or what motivated them. Therefore, knowledge that generalizes over such a dataset is at best an approximation of expert decision making.

Yet another compromise was to limit what is remembered. Expert human players acquire, maintain, and apply large stores of knowledge. They identify play in roughly three stages: the opening, the middlegame, and the endgame. An *opening* is a sequence of moves by the first player annotated with the opponent's anticipated responses. Openings vary in length and complexity, and branch in response to the opponent's moves. A chess grandmaster, for example, typically maintains an *opening book* of about a dozen well-practiced openings and replaces a few of them annually. Expert human players also memorize many *endgames*, strongly solved states where relatively few pieces remain in play. In addition, they have access to a wealth of historic contests between experts. A *plan* is an intended sequence of actions to achieve a specified goal. In games, however, a plan must anticipate the moves of the opponent. A *tree-based* plan represents alternative decisions as a tree where a state with a set of legal moves has a branch for each of the next states those moves produce. Openings are tree-based plans to gain an initial advantage, and endgames are tree-based plans to reach a desired outcome.

The next section of this paper describes two early approaches to game playing, one that sought to learn a model of how the world works and another that intensively engineered software and hardware to explore millions of game states per second. Subsequent sections examine how game-playing programs with symbolic knowledge representation matured and how powerful subsymbolic approaches ultimately eclipsed them. The final sections discuss what subsymbolic systems lose in their drive to outperform humans, and considers to what AI should next aspire.

2. One-game players

Early game-playing programs, such as Arthur Samuel's checker player, were restricted to a single game [9]. To improve its skill, the program needed more knowledge. Rather than extract and encode human expert knowledge, Samuel introduced the term "machine learning" and identified two possible approaches to it: a network-like system that would be rewarded or punished for its initially random behavior, and a "highly organized" system that would be restricted to a specific challenge. Given the resources of 1959 computers and checkers' 5×10^{20} possible states, he quite reasonably chose the specifically targeted approach.

Samuel's early version of a checker player was both formative and prescient. It described states with multiple, hand-crafted descriptive *features* (e.g., total mobility) intended to capture significant properties of the game. The program reasoned with an *evaluation function*, a weighted linear sum of feature values that calculated a *score* for how "good" a state was. To evaluate the quality of a move m that led directly to state s, the algorithm searched ahead from s, growing a game tree of states and moves to reach them as long as subsequent moves improved the next

state's score. To conserve memory and speed search, the program used alpha-beta pruning, and then extracted a path from the current state to the state with the highest computed score.

Samuel imbued his program with his own knowledge and then had it learn weights for its evaluation function. The program drew its training examples from contests it played against a human or against itself (*self-play*), and corrected its weights based on the outcome of the contest from which the example was drawn. It also memorized some states along with the outcome for the player who made the move. In later experiments, Samuel investigated the relevance and interdependence among his 40 checkers features, and sought an optimally weighted evaluation function on a subset of features [10]. While his work eventually resulted in presentable amateur-level play, Samuel wrote that he longed for a better way to learn both weights and new, relevant features.

Decades later, as computational resources expanded, other AI researchers sought to engineer optimal play with more knowledge and better search algorithms. Jonathan Schaeffer led a team that developed Chinook, the first program ever to win a contest against a human checkers world champion. Schaeffer used Chinook to support a 16-year effort that weakly solved checkers, that is, he proved checkers was a draw with a game tree that considered all possible states if both players played perfectly. Chinook represented states with a small set of handcrafted features. Its reasoning merely extracted and executed a perfect line of play from the game tree, followed by the relevant entry from the endgame database [8]. There was no learning; Chinook stored only its game-tree proof and the endgame database. In retrospect, Schaeffer claimed that fast, clever, deep search supported by carefully honed knowledge representation and parallel processing implicitly garnered the knowledge necessary to solve checkers [11]. Chinook knew everything necessary to play checkers perfectly. On the other hand, it could offer no explanation beyond "there's a better move" for why unrecorded choices would be incorrect.

Another major game target was chess, with a far larger state space and *branching factor* (number of alternative moves to choose among). In 1997, Deep Blue was the first program to defeat a human world chess champion, Gary Kasparov. To play a contest, Deep Blue began play with an opening book that referenced more than 700,000 grandmaster contests. It then used its 8000-term weighted linear evaluation function to search a traditional game tree 6 to 8 moves ahead (but occasionally as deep as 20). Once no more than five (sometimes six) pieces remained on the board, Deep Blue followed the tree-based plan retrieved from its endgame database. The weights for Deep Blue's evaluation function were hand-tuned on training examples drawn from thousands of expert contests, including some against Kasparov. Its tree-based plans were its human-curated openings and endgames. A team of international grandmasters helped tune its heuristic evaluation function and improve its knowledge base of openings and endgames [12]. If asked why it had made a particular move m, Deep Blue might have been coded to cite its opening, provide its endgame entry, or exhaustively list the 8000 weights and features that scored m, perhaps with some lower-scoring alternatives.

3. In Search of Knowledge and Representation

3.1. The (Human) Expert Way

Because AI targets intelligent human behavior, research on game playing has long studied the most expert human players. Although they may be drawn or encouraged toward a single game quite young, there is currently no evidence that expert human players are born as experts. Young experts-to-be play and study a single game obsessively, and traditionally receive intensive instruction from more expert players. The top young Japanese Go students, for example, attend boarding schools whose entire curriculum is Go, and often become professional players by age 12 [13].

Expert human players incorporate both strategies and tactics in their decisions. A *tactic* is a sequence of moves that repeatedly threaten the opponent's pieces (e.g., a fork), while a *strategy* is a longer sequence of moves whose advantage is cumulative. Although the goal in chess is to checkmate one's opponent, chess analysts have written extensively about how tactics and strategies differ from one stage of the game to the next.

Expert human players appear to perceive data and store knowledge in several forms. It is difficult for human experts themselves to report what they see, although eye-tracking devices can record where they look. Early studies suggested that chess masters saw a game board as an assembly of *chunks*, small salient patterns [14]. Later, however, psychologists found no evidence that Go masters relied on chunks or had an extensive memory of them [15]. Instead, they appeared to have a representation that dynamically invested Go stones with multiple roles in a parallel stream of ongoing skirmishes on different parts of the board. As for search, at least as they report it, both chess masters and Go masters consider far fewer moves and search far less deeply than their machine opponents [16, 17].

3.2. The GOFAI Way

For decades, expert game-playing programs relied on a carefully crafted or learned evaluation function, extensive databases of openings, endgames and expert contests, plus search. This was *GOFAI* (Good Old-Fashioned AI), symbolic computation pushed to its extreme. From every possible move m in the current state, the program would look multiple states ahead to predict the outcome of the contest after m. Ideally, this search would eventually reach an extensive endgame database that recorded an inevitable outcome and the moves to achieve it. Chinook's power, for example, relies heavily on an endgame database for each of the 39 trillion states where 10 or fewer of the original 24 pieces remain on the board. More often, however, time constraints halt search at an intermediate state where GOFAI can only estimate the outcome with an evaluation function. In both cases, the highest scores are propagated backward along the search path to evaluate a move under consideration.

There are several reasons to be chary about such search. Because it depends on estimation by a heuristic evaluation function, the result may be deceptive; trouble may lurk just beyond where search stopped. (Quiescence algorithms check to confirm that the score of particularly attractive states is maintained several moves deeper [18].) Typically, multiple moves are worthy of consideration from each state and the order in which they are searched is also determined

heuristically. When more than one sequence of moves can reach the same state, a tradeoff arises between the cost to re-evaluate states and the bookkeeping necessary to remember their scores.

Expert human knowledge has traditionally been important in computer game playing. Many strong programs were coded by researchers who were already expert or master players at that particular game. Those programmers designed state features that reflected their own knowledge or intuition about the game and took pains to calculate their features efficiently. They also observed and revised the evaluation function to drive it toward the decisions they believed were correct, often with help from human master players. For example, a newly-involved team of grandmasters received much of the credit for the difference between Deep Blue's loss to Kasparov in 1996 and its triumph in 1997.

At the AAAI-1998 Hall of Champions exhibition. three of the programs on display made clear why GOFAI succeeded. They all had extensive knowledge bases and relied on deep, fast, clever search, but the key to their prowess in every case was the way they most often made decisions: their evaluation function. They played backgammon [19], Othello [20], or Scrabble [21] at the same level as their human champion opponents, and their creators agreed that learning the right evaluation function had been essential.

All the learning for those evaluation functions occurred offline, however, in advance of any competition. GOFAI could no longer contend with the size of the state space and the branching factor of these games, and construction of specific hardware for each new game was not an option. Moreover, there were targeted games like shogi where the branching factor increases rather than decreases as a contest progresses, so that search toward the end of play is more demanding. Indeed, a speaker at AAAI-1999 warned that game targets like shogi and Go would not succumb to GOFAI, and encouraged researchers to revisit how humans make decisions [22].

4. Knowledge and Symbols

Initially, Samuel considered "switching networks" to ferret out useful knowledge, but dismissed the idea as impractical because they seemed slow to learn and computationally expensive [9]. Artificial neural networks (ANNs) are the modern version of what Samuel had in mind. Figure 1 is an example of a simple feed-forward network, where data flows from left to right. Each circle in Figure 1 represents a *functional unit* that sums the weighted input values it receives, applies a function (e.g., sigmoid or tanh) to introduce some nonlinearity, and then forwards the result as its output. Each edge in Figure 1 represents transmission of some weight times the value output by its left endpoint as input to the unit at its right endpoint.

Layers of units (shown vertically in Figure 1) are connected only to units in the immediately preceding layer and the immediately following one. Each unit in the input layer reports a feature value; each output unit reports a result (e.g., outcome probability) given its input. Between the layers for input and output, intermediate *hidden layers* also form weighted combinations of values. This *compositionality* is a powerful computational tool that allows each hidden layer to construct new features from the features built in the preceding layer. Indeed, any continuous function can be approximated to arbitrary accuracy by a network with two hidden layers, although more layers may lead to simpler networks [23].

The key to expert decisions, as Samuel realized, is a good evaluation function, one with

incisive features and appropriate weights. Given a feature-based state representation as input for state *s*, an ANN can calculate a weighted non-linear score for *s*. Moreover, given a large dataset of (*state*,*move*) training examples, temporal-difference learning can train an ANN's weights to score each *move* highest for its associated *state*. When it is the program's turn to move in state *s*, it can then simply choose the move from *s* that has the highest score. Samuel's other wish, to learn new features for a game, is provided by ANNs' compositionality. Thus a program with no information about an appropriate evaluation function might learn one with an ANN that has multiple hidden layers and enough expert training examples. This is *supervised learning* because the correct move is available to train the weights.

ANNs are subsymbolic, that is, they extract a world model only from the experience in their dataset. An ANN's weights are initially set to small random values, without regard to what the input features represent in the world they are intended to describe. ANNs achieved their first successes on problems that involved perception, where there was ready access to vast quantities of associated data but little knowledge about how to apply it. Those ANNs harnessed spatial or temporal continuity in their input to detect relationships among its features. As the next section indicates, with temporal difference learning ANNs can learn to apply those relationships and thereby drive a program to make the right decisions.

5. Multi-game players

In 1979, Hans Berliner's backgammon player BKG became the first program ever to defeat a human world-champion at a challenging game [24]. BKG used tree search with only six handcrafted, computationally intensive features in a linear evaluation function. Berliner was also a world-class competitor in both chess and bridge. To simulate someone who learned to play multiple challenging games expertly, GOFAI would require knowledge about how to learn to play games in general, rather than how to learn a specific game. Subsymbolic programs have risen to that challenge.

In the last few years, DeepMind has produced a sequence of increasingly ambitious game

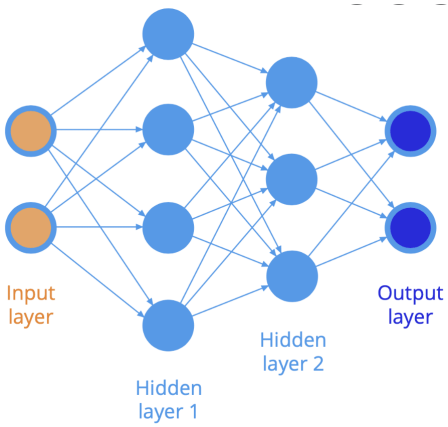

Figure 1: A simple neural network.

learners with little or no expert knowledge. They all apply reinforcement learning to large, deep (i.e., many-layered) neural networks, and their only input is the pixel values in a 19×19 image of the board's initial state. The first target of these game learners was Go, a considerably larger game than chess, with approximately 3^{361} possible states and a branching factor of 250.

AlphaGo was a *semi-supervised* reinforcement learning program for Go. It learned to play first on 30 million expert training examples but, once it became a strong player, it continued to learn only on examples drawn from self-play. This program had no Go-specific features, no suggestions about expert strategy or tactics, and no endgame database or opening book. Nonetheless, in 2016 AlphaGo became the first program ever to defeat a 9-dan professional (top-ranked) human Go player, and thereby earned the same rank [25].

AlphaZero learned to play multiple games at a superhuman level, exclusively from self-play, without any expert knowledge or expert training examples [4]. This was *unsupervised* learning; each training example included only a state from a contest the program had played against itself. Its targets were chess, Go, shogi, and a suite of 57 Atari games. For each game, AlphaZero knew the rules and used them to learn a *prediction network* with Monte Carlo Tree Search (MCTS). This method explores the value of a state with multiple *rollouts*, each a sequence of randomly selected legal moves that ends in a terminal state [26]. Whenever it reached a previously unexplored state s, AlphaZero performed a fixed number of rollouts and backed up their' rewards to calculate the expected value of the reward it would receive after each possible move from s. It also tabulated a probability distribution to describe how likely each move was to be the best choice.

MuZero pushed the envelope further [27]. MuZero does not know the rules when it begins to learn a new game, so it cannot look ahead with tree search. Instead, it learns the rules. When it is MuZero's turn to move, it is given the legal moves from the current state, and when the contest ends it receives a numeric reward. Like AlphaZero, MuZero has no expert knowledge. The only features it begins with are the pixel values in an image of the initial state, and its only goal is to accumulate rewards. It also stores in memory the most recent million contests it has played against itself, along with prior network models it built for the game it is learning. MuZero takes an approach similar to AlphaZero's but learns three networks. First, because it is given no sophisticated features to describe its states, MuZero learns a *representation network* that imagines the current state as a *hidden state*. Second, instead of reinforcement learning's traditional transition table, MuZero learns a *dynamics network* to capture how moves change hidden states. The dynamics network accepts the current hidden state and a move and generates a reward and a new hidden state. Third, like AlphaZero, MuZero learns a prediction network with MCTS, but MuZero must first infer the current state with its representation network. Then, once it begins to grow an MCTS tree, it moves through it with the representation network and the dynamics network, imagining a contest with hidden states and their outcomes. So successful behaviors learned in MuZero's imagined contests perform well during actual play, all three networks are initialized at random and trained simultaneously on decision subsequences randomly selected from its stored contests. Although it begins with less knowledge than AlphaZero, MuZero learns faster, plays better Go, and plays the other games at least as well.

Finally, all the games discussed thus far have been *deterministic*, that is, when a player in state s makes move m, their opponent must then move from the next state s' which is known to both players. In a *stochastic* game, however, s' is modified non-deterministically into a *chance*

state s″ from which the opponent is to move. For example, backgammon is a stochastic game because on each turn a player rolls a pair of dice to determine their next legal moves. Given a perfect stochastic simulator of dice rolls to predict *s″*, AlphaZero learned to play backgammon as well as GNUbg Grandmaster, a superhuman, state-of-the-art backgammon player with a perfect dice-roll simulator and multiple backgammon-specific features [28]. MuZero, however, is deterministic and so lacks any knowledge about how dice behave. As a result. MuZero played backgammon less well than AlphaZero. To address this, a new version, Stochastic MuZero, now learns an additional network that calculates the chance state and applies it to perform stochastic tree search. This allows Stochastic MuZero to match the performance of both AlphaZero and GNUbg on backgammon [29]. Moreover, if allowed an order of magnitude more rollouts than it used for its other games, Stochastic MuZero outplays GNUbg. To date, however, there are no available results for the program on any other two-person stochastic game.

6. Aspirations for Intelligence

Samuel's two approaches to game playing culminated in the GOFAI archetype Deep Blue and the subsymbolic archetype MuZero. Deep Blue derived its prowess from improved search, human expert knowledge, and special-purpose hardware tailored to its knowledge representation and reasoning. It represented game states with human-designed handcrafted features and was tuned to play one game very well. Deep Blue reasoned with a learned weighted linear model, relied on game-specific hardware and software that drove fast, clever search, and drew on an extensive memory of experts' openings, endgames, and historical expert contests. In contrast, MuZero learns to play multiple games in only days (or even hours) of processing, on hardware and software designed for rapid parallel computation. From a single image, MuZero learns an unknown representation and an unknown model of the world that it uses to make decisions. All its knowledge is represented in three large neural networks whose weights it learns with *tensors*, mathematical objects that represent relationships in vector space but are not problem-specific. MuZero remembers only the last million contests it has played against itself and old versions of its three networks.

Both Deep Blue and MuZero play as if they were experts. Consider, however, another excerpt from the first paragraph of the Dartmouth proposal:

> An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves.[1]

Clearly as AI agents Deep Blue and MuZero are incomplete. Their missing aspects are discussed next, as aspirations for AI that go beyond game playing.

6.1. Remembering and Forgetting

Consider Fool's Mate, a full but brief contest to which novice chess players are routinely introduced. White, the first player to move, need make only two seemingly innocuous initial choices to find itself in checkmate after four moves. There are eight different lines of play for Fool's Mate, along with several six-move variations. All of them result in checkmate by

Black's queen, although in the initial state that queen is constrained by its own pawns from any movement and quite far from White's king. Chess players recall the first time they encountered Fool's Mate with a smile or a grimace but rarely with indifference.

Fool's Mate is a salient lesson for two reasons. First, it is important to learn from failure as well as from success. Knowledge that can be harnessed only to win is often deceptive when one tries to transfer it. (For example, expert play of tic-tac-toe has no obvious resemblance to expert play for lose tic-tac-toe, a game where players must avoid three pieces in a row, column, or diagonal.) The lesson to be learned from Fool's Mate is that mobile but geographically distant pieces are threats too, but if queried Deep Blue and MuZero could only report that there are better openings. Second, human memories often come with emotional tags that elicit appropriate actions. Deep Blue only saves expert openings, and after a million contests MuZero could eventually forget Fool's Mate. In contrast, humans' selective memory facilitates the retrieval of significant experience that both cues behavior and might spur exploration.

6.2. Abstraction, Analogy, and Planning

It is not enough to memorize experience; for it to be useful it must be organized. Pattern recognition helps humans organize their experience. When they detect a set of objects, events. or actions that play a similar role in their environment, humans often abstract it as a *concept*. Human game-playing concepts, for example, include "retreat", "trap," and "fork." Within a single game, a concept is a set of states with a similar relationship among their constituents. A fork in chess, for example, is a move that involves at least three pieces, where one or more of a player's pieces threaten to capture at least two of the opponent's, and no single move by the opponent can prevent all captures. Deep Blue might have a term in its evaluation function to detect forks but has no way to indicate how significant that term was in its decision. MuZero might have a fork alert among its learned features, but likely an unrecognizable one.

Another way to organize experience is to store a sequence of actions with a descriptive label and an executed solution. Many human decision makers, from emergency rescue teams to military commanders, rely on such case-based reasoning to devise solutions in novel situations under pressure [30]. This approach supports transfer learning that organizes experience with thoughtful indexing to support rapid retrieval for pertinent analogical reasoning. Once a human understands a fork in tic-tac-toe, for example, it is far easier for them to identify and employ one in chess. Such analogical thinking is a hallmark of human intelligence. Concepts help humans learn faster and develop expectations about what to look for when, for example, they learn to play a new game or enter another competitive environment.

Humans also organize knowledge temporally, and even the fastest programs may benefit from temporally-aware resource allocation. In a game, for example, the rules may limit elapsed wall-clock time. Moreover, time measured by the number of moves prior to the current state delineates openings and endgames.

Concepts often form natural hierarchies that support partial planning. For example, a fork is a kind of attack. When a problem includes multiple agents, however, particularly human ones, even partial plans are fragile. In such a dynamic world it may be necessary to replan when the environment changes. For example, when the advantage shifts from one game player to another it may be necessary to reconsider the provenance of the experience on which an original plan

was chosen. Rollouts, for example, may be less predictive than historical contests against the current opponent.

6.3. Collaboration, Explainability, and Creativity

It is not enough to store and organize experience; to have extensive impact that knowledge must be shared. Human experts dispense their knowledge with examples. When master players teach or write about their game, they show why a move is a bad choice by the perils it creates and the benefits that accrue from alternative moves. When human masters describe individual tactics, they create illustrative situations, and when they describe strategies, they recount them with important historical contests. This is when memorable failures like Fool's Mare are as salient as successes. All this, of course, requires natural language.

For collaboration, plans too should be readily translatable into natural language. Long or elaborate plans may be better understood when described with higher-level concepts than with a sequence of moves. Deep Blue's plans are its openings and endgames designated by experts and might succumb to some incisive natural language processing. MuZero's plans are implicit in its three networks; their description in natural language would be a daunting task.

Human experts collaborate. Chess grandmasters, for example, often travel to a tournament with a team that analyzes contests immediately after they are played. The team attempts to identify a particular opponent's strategy and the crucial moves that supported a winning tactic. Although they may not achieve unanimity, these discussions highlight salient experience and provide fodder for analogy, all in natural language. An expert program that could communicate in natural language could make important contributions to such a discussion.

MuZero's most remarkable feature is its ability to learn without any indication from humans about appropriate knowledge representations for states and moves. Its superhuman skill clearly indicates that its representation network has learned knowledge well adapted to its stored data and its learned prediction network. Such synchrony could support faster retrieval and storage, so that parallel training processes retrieve more accurate information. MuZero may also be discovering novel and incisive but opaque game-specific features; Deep Blue's features were fixed and weighted by grandmasters. In both cases, human experts could benefit from an explanation of the features. When an expert calls a program's move "creative," they are really saying that they have never seriously considered it before. A high-level explanation of the motivation for that move, along with an illustrative example like Fool's Mate, would be more readily retrieved and repurposed.

7. Conclusion

Problems are no longer "reserved for humans" as they were in 1955. AI now tackles many challenging tasks, including language translation, disease diagnosis, and drug design. Research starts small, on a relatively simple version of a problem. When it succeeds, researchers make the problem harder. When progress eventually stalls, their drive for skilled behavior sacrifices human-friendly components in favor of increasingly better "results." Researchers want their agents to make the right decisions and watch them carefully to see if they do. Modern game-

playing programs have taken this to an extreme — to gauge the skill of these superhuman players, testing can now only pit them against one another.

The previous section illustrated what an artificial agent can lose when all that matters is the win. While superhuman behavior is impressive, human society wants more from superior expertise than skilled decisions. Humans are reluctant to trust "algorithms," which they increasingly blame for outcomes that disappoint or infuriate them. If humans are to trust an expert program, it must help them build a mental model of how it perceives and reasons [31, 32]. It is one thing to compete against an unknowable Go opponent, but quite another to submit oneself to an unknowable surgeon. Humans want algorithms that are both skilled *and* knowledgeable. They want clear explanations, not reams of data accompanied by trillions of weights.

Challenges still remain in game playing. For example, the models built by subsymbolic programs are based on finitely-many possibilities sampled with rollouts and chance nodes. Real-world problems, however, often include real-valued features that offer infinitely many choices. Moreover, learning that induces knowledge, even if it is asymptotically correct, can still lead to costly mistakes. AI practitioners, those who tout AI, and those who seek to control AI must all remember that while machine learning enables generalization over more data than people ever experience, a perfect inductive learner remains unreachable.

Fundamentally, what intelligence seeks is an omniscient model of a world, a model that accurately predicts what will happen as a result of taking any action in any state. This paper has described how AI research progressed through a sequence of superhuman programs for challenging games. Along the way, like humans, AI abandoned perfection for reliability. AI also accepted that experience and memory are finite. GOFAI modeled human knowledge and behavior, but to learn to play all games that way would have required a team of multi-game human experts — or a vast dataset of game-playing experience. Self-play generated massive quantities of data within which subsymbolic programs found powerful opaque patterns, but they should not be viewed as solutions.

As subsymbolic programs sometimes dazzle us and sometimes fumble, researchers must remember that AI is intended to serve society. It is increasingly clear that human experts will collaborate extensively with computers to develop expertise, both their own and that of the machine. A machine's drive to win, however, may not be enough. Problem solving requires a toolbox of methods, many of which are cued by human behavior; developers should not dismiss them as computationally intractable or unnecessary. Programs that "think fast" (subsymbolically) can react to protect humans and achieve surprising skill, but they remain insistently opaque. Programs that "think slow" (symbolically), can more readily be adapted to plan and converse and collaborate with humans. AI needs both: a pair of experts for a problem, one fast and data-laden, the other slow and more accessible.

The premise of a program like Stochastic MuZero is that random sampling of states in a dynamic world will ferret out its most important elements. Randomness lacks the emotional tags humans attach to salient experience, and relies only on fortuitous exploration for creativity. Even if a program successfully imagines a complex dynamic reality, however, a limited memory will eventually discard well-understood pitfalls or significant concepts it might have used to instruct others. That would be a significant loss. A problem, after all, is about more than its answer.

Human expert players develop only with years of training and typically excel at only one

game. When human players perceive a state, how they represent it to support reasoning remains unknown. Either they learn representations that are different from those that programs learn or they reason with them differently, or perhaps both. Protocols indicate, however, that human experts remember and apply lessons drawn from some states, and formulate and execute partial plans. Moreover, they embed all this knowledge and learning in natural language laden with illustrated concepts that label experience. It could be that humans do all this to compensate for their fallible and limited memories. Or perhaps humans are playing for more than the win.

References

[1] McCarthy, J., Minsky, M., Rochester, N., Shannon, C.E., A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, http://raysolomonoff.com/dartmouth/boxa/dart564props.pdf, 1955. Accessed: 2022-8-22.

[2] McCarthy, J., Minsky, M., Rochester, N., Shannon, C.E., PaLM: Scaling Language Modeling with Pathways, https://arxiv.org/abs/2204.02311, 2022. Accessed: 2022-10-4.

[3] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, Mark Chen, Hierarchical Text-Conditional Image Generation with CLIP Latents, https://arxiv.org/abs/2204.06125, 2022. Accessed: 2022-10-4.

[4] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al., Highly accurate protein structure prediction with alphafold, Nature 596 (2022) 583–589.

[5] D. Kahneman, Thinking, Fast and Slow, Farrar, Straus and Giroux, 2011.

[6] L. Allis, Searching for Solutions in Games and Artificial Intelligence, Ph.D. thesis, Maastricht University, 1994.

[7] H. J. van den Herik, J. W. Uiterwijk, J. van Rijswijck, Games solved: Now and in the future, Artificial Intelligence 134 (2002) 277–311.

[8] J. Schaeffer, Y. Björnsson, N. Burch, A. Kishimoto, M. Müller, R. Lake, P. Lu, S. Sutphen, Solving Checkers, in: Proceedings of IJCAI-05, 2005, pp. 292–297.

[9] A. L. Samuel, Some Studies in Machine Learning Using the Game of Checkers, IBM Journal of Research and Development 3 (1959) 210–229.

[10] A. L. Samuel, Some Studies in Machine Learning Using the Game of Checkers. II - Recent Progress, IBM Journal of Research and Development 11 (1967) 601–617.

[11] J. Schaeffer, N. Burch, Y. Björnsson, A. Kishimoto, M. MÃŒller, R. Lake, P. Lu, S. Sutphen, Checkers is Solved, Science 317 (2007) 1518–1522.

[12] F.-h. Hsu, M. S. Campbell, A. J. Hoane, Deep Blue System Overview, in: Proceedings of the 9th International Conference on Supercomputing, (ICS '95), Association for Computing Machinery, New York, NY, USA, 1995, p. 240–244.

[13] Agence France-Presse, An entire school dedicated to game of Go, https://www.scmp.com/yp/discover/lifestyle/features/article/3058853/entire-school-dedicated-game-go, 2016. Accessed: 2022-8-25.

[14] A. de Groot, Thought and Choice in Chess, Mouton Publishers, The Hague, 1965.

[15] R. Grimbergen, Cognitive Modeling of Knowledge-Guided Information Acquisition in

Games, in: H. J. van den Herik, X. Xu, Z. Ma, M. H. M. Winands (Eds.), Computers and Games, Springer, Berlin, Heidelberg, 2008, pp. 169–179.
[16] F. Gobet, H. A. Simon, Expert chess memory: Revisiting the chunking hypothesis, Memory 6 (1998).
[17] A. Yoshikawa, Y. Saito, Hybrid Pattern Knowledge: Go Players' Knowledge Representation for Solving Tsume-Go Problems, in: Proceedings of the South Korean International Conference on Cognitive Science, 1997.
[18] D. F. Beal, A generalised quiescence search algorithm, Artificial Intelligence 43 (1990) 85–98.
[19] G. Tesauro, Temporal Difference Learning and TD-Gammon, Communications of the ACM 38 (1995) 58–68.
[20] M. Buro, From Simple Features to Sophisticated Evaluation Functions, in: H. J. van den Herik, H. Iida (Eds.), Computers and Games, Springer Berlin Heidelberg, 1999, pp. 126–145.
[21] B. Sheppard, World-championship-caliber Scrabble, Artificial Intelligence 134 (2002) 241–275.
[22] S. L. Epstein, Game Playing: The Next Moves, in: Proceedings of the Sixteenth National Conference on Artificial Intelligence, 1999, pp. 987–993.
[23] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control, Signals, and Systems (MCSS) 2 (1989) 303–314.
[24] H. J. Berliner, Backgammon computer program beats world champion, Artificial Intelligence 14 (1980) 205–220.
[25] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of Go with deep neural networks and tree search, Nature 529 (2016) 484–489.
[26] R. Coulom, Efficient selectivity and backup operators in Monte-Carlo tree search, in: Proceedings Computers and Games 2006, Springer-Verlag, 2006, pp. 72–83.
[27] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al., Mastering Atari, Go, chess and shogi by planning with a learned model, Nature 588 (2020) 604–609.
[28] Free Software Foundation, Gnu backgammon, https://www.gnu.org/software/gnubg, 2004. Accessed: 2022-10-9.
[29] I. Antonoglou, J. Schrittwieser, S. Ozair, T. K. Hubert, D. Silver, Planning in stochastic environments with a learned model, in: International Conference on Learning Representations, 2022. URL: https://openreview.net/forum?id=X6D9bAHhBQ1.
[30] G. Klein, R. Calderwood, Decision models: Some lessons from the field, IEEE Transactions on Systems, Man, and Cybernetics 21 (1991) 1018–1026.
[31] T. Kulesza, S. Stumpf, M. Burnett, S. Yang, I. Kwan, W.-K. Wong, Too Much, Too Little, or Just Right? Ways Explanations Impact End Users' Mental Models, in: 2013 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), IEEE, 2013, pp. 3–10.
[32] A. Bussone, S. Stumpf, D. O'Sullivan, The Role of Explanations on Trust and Reliance in Clinical Decision Support Systems, in: 2015 International Conference on Healthcare Informatics (ICHI), IEEE, 2015, pp. 160–169.