The Games AIs Play - A Comprehensive Review

Thomas R. Robbins
East Carolina University, Greenville, NC
(robbinst@ecu.edu)

ABSTRACT

Competitive games have long served as structured and measurable testing grounds for AI research. Many foundational AI innovations emerged from efforts to achieve superhuman performance in domains including board games (Chess, Go), knowledge-based competitions (Jeopardy), and real-time strategy games (StarCraft, Diplomacy). The evolution of AI in games offers valuable lessons for business applications, particularly in decision-making, strategic planning, and automation. However, much of the research in this area is found in technical publications and pre-print repositories, with limited engagement from business scholars. This paper synthesizes key developments in AI applied to games, highlighting their technological evolution and drawing insights relevant to business applications.

1 INTRODUCTION

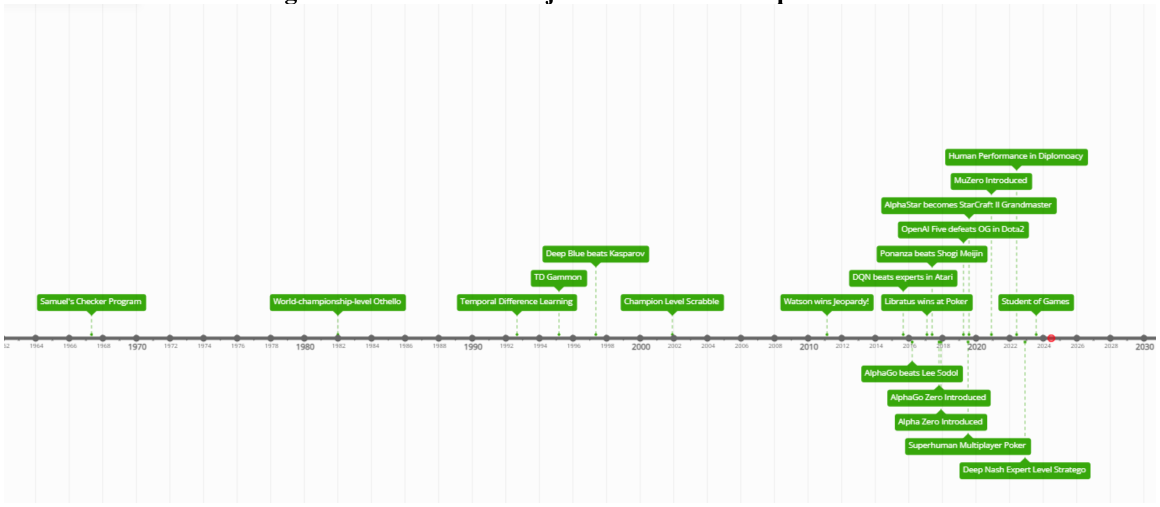
Games have played an important role in the development of Artificial Intelligence since the field was first conceptualized. Games provide a controlled and well-defined environment in which to operate and test AI methodologies. Games such as chess and Go have long been considered among the most intellectually challenging tasks in which humans can compete. Achieving human-level or super-human level performance in these games has been a longstanding goal for AI developers, one that for many years was considered beyond the abilities of computers.

The popularity and visibility of games has also been critical in bringing attention to AI development. Deep Blue's triumph over undefeated world champion Gary Kasparov in chess, Watson's defeat of champions Ken Jennings and Brad Rutter in Jeopardy, and AlphaGo's defeat of Lee Sedol in Go, have not only showcased the remarkable advancements in AI but also captivated public interest and demonstrated the potential of AI to excel in complex, strategic tasks. Game playing AI developers have been frequently featured on Lex Fridman's Artificial Intelligence podcast.

Work related to intelligence in computers dates back to Turing's work in the 1940s and the publication of the famous Turing Test concept in his 1950 paper *Computing Machinery and Intelligence* (Turing 1950). The term Artificial Intelligence was coined by Jon McCarthy at his famous two-month seminar at Dartmouth in the summer of 1956 (Russell and Norvig 2020). Of the 10 attendees at that conference four of them, Claude Shannon, Allen Newell, Herbert A. Simon, and Arthur Samuel, published work related to computer game playing. Shannon's work on a computerized approach to chess pre-dated the Dartmouth conference by some 6 years.

Games played an important role in the development, popularization, and evolution of AI since its inception. Figure 1 provides a timeline of some of the selected game related AI milestones.

Figure 1- Timeline of Major AI Game Accomplishments



During the early stages of AI, classical techniques of directed search, often referred to as Good Old-Fashioned AI (GOFAI), AIs were able to develop human level and sometimes super-human level performance in games such as checkers, chess, backgammon, Scrabble, and Jeopardy. But it was not until the reemergence of neural networks and the development of deep learning in the late 2010s that game playing AIs matured and begin to dominate a range of games previously thought beyond the scope of AI. A series of increasingly capable AIs from Google's Deep Mind conquered the game of Go. As was the case for AI in general, these programs transitioned from expert-based systems where human experts documented their expertise in the form of hand-crafted heuristics, to self-learning systems provided with little more than a basic set of game rules.

AI programs, increasingly built using a deep reinforcement learning approach, have conquered board games such as chess, Shogi, Go, and Stratego. They have also achieved high-level performance in classic Atari video games without any human guidance. Moreover, AIs have reached expert-level proficiency in more advanced video games like Dota 2 and StarCraft II. In the realm of card games, the AI Libratus excelled in the two-player version of No-Limit Texas Hold'em by incorporating economic game theory concepts, including Nash equilibria. Subsequently, another AI called Pluribus advanced to dominate six-player competitions in the same game.

Recently, a Meta-based AI named Cicero has demonstrated high-level capability in the game of Diplomacy, which requires not only strategic decision-making but also natural language communication skills. Cicero utilizes standard AI game tools, such as deep learning and reinforcement learning, while integrating a large language model for communication. More recently, a group of researchers from Deep Mind and academia have developed the Student of Games, a unified learning algorithm that can play a wide variety of games, including games of both perfect and imperfect information. Student of Games incorporates a wide range of AI techniques including guided search, self-play and game theoretic reasoning. Student of Games is a general-purpose algorithm that can achieve high level play in games that include chess, poker and Scotland Yard, a further step toward a generalized AI capability, at least withing the domain of games.

Most of the AI game literature has been published outside of standard business-oriented journals. Some are published in computer science journals or proceedings, or general interest journals such as Science and Nature. Many of the seminal papers in AI are published as industry working papers posted to the website ArXiv. Much of this research has been conducted in industry labs at IBM, Google DeepMind, Facebook Artificial Intelligence Research (FAIR) and OpenAI. In this paper we review the development of AI as it relates to games and in the process highlight the major conceptual and architectural advancements of AI in general. We identify the major milestones and the key papers that document the development of this subset of Artificial Intelligence. Our goal is to present this stream of research to the business oriented academic community.

2 GAMES AND GAME CHARACTERISTICS

In this section we review the basic dimensions that characterize and differentiate games. The characteristics of the game influence the type of technology that can be applied to play the game. We then briefly introduce the games that are included in our review and document each game's characteristics. Details of each game will be discussed later in the paper.

Game Dimensions

Games can be characterized along several different dimensions. Here we outline the dimensions most relevant for our review.

- **Certainty**: games can be deterministic, with no randomness affecting the outcomes, or stochastic, incorporating elements of chance or randomness.
- **Players**: games can involve a single player, two players, or multiple players.
- **Information Set**: games can feature perfect information, where all players have access to all game-related information, or imperfect information, where some players possess private or hidden information.
- **Symmetry**: games can be symmetric, where all players have identical resources and options, or asymmetric, where players have different resources or options.
- **State Space**: games can have a discrete state space, with a finite number of possible states, or a continuous state space, with an infinite number of possible states.
- **Timing**: Games can be real-time, where actions take place continuously and simultaneously, or turn-based, where players take turns to make their moves.

- **Environment**: can be physical, involving tangible components like in board games, or digital, existing in electronic formats like video games.

The difficulty of solving a game algorithmically is related to the complexity of the game. Complexity is a combinatorial concept that bounds the search space of a game. Multiple methods exist to measure game complexity. The state-space complexity is the number of legal positions reachable from the game's initial configuration. Game tree size is the total number of unique games that can be played ('Game complexity' 2025). In some cases an exact measure of game complexity cannot be made, but an upper bound can be specified.

Over the course of this review, we will discuss AIs that have been applied to a wide range of games. We will discuss the specific details of those games as we work thorough the review, but the basic characteristics of those games are documented in Table 1.

Table 1- Categorization of Games Analyzed

Game	Certainty	Players	Information Set	Symmetry	State Space	Environment	Timing
Checkers	Deterministic	Two	Perfect	Symmetric	Discrete	Physical	Turn-based
Chess	Deterministic	Two	Perfect	Symmetric	Discrete	Physical	Turn-based
Go	Deterministic	Two	Perfect	Symmetric	Discrete	Physical	Turn-based
Othello	Deterministic	Two	Perfect	Symmetric	Discrete	Physical	Turn-based
Backgammon	Stochastic	Two	Perfect	Symmetric	Discrete	Physical	Turn-based
Scrabble	Stochastic	Multi	Imperfect	Symmetric	Discrete	Physical	Turn-based
Jeopardy	Deterministic	Multi	Perfect	Symmetric	Discrete	Physical	Real-time
Poker	Stochastic	Two/Multi	Imperfect	Asymmetric	Discrete	Physical	Turn-based
Stratego	Deterministic	Two	Imperfect	Asymmetric	Discrete	Physical	Turn-based
Shogi	Deterministic	Two	Perfect	Symmetric	Discrete	Physical	Turn-based
Atari Games	Varies	Single	Varies	Varies	Varies	Digital	Real-time
Dota 2	Stochastic	Multi	Imperfect	Asymmetric	Discrete	Digital	Real-time
Starcraft 2	Stochastic	Multi	Imperfect	Asymmetric	Discrete	Digital	Real-time
Diplomacy	Deterministic	Multi	Imperfect	Symmetric	Discrete	Physical	Turn-based
Scotland Yard	Deterministic	Multi	Imperfect	Asymmetric	Discrete	Physical	Turn-based

In most cases these classifications are clear, but in some cases, there is some ambiguity we will address. For example, Jeopardy is for the most part a game of perfect information, but in the final Jeopardy round a player's wager is private information. In terms of environment, we classify video games as digital and all other games as physical. Games such as poker can be played on-line but are not video games in the sense that the on-line version is not fundamentally different than the game played with physical cards.

3 TECHNICAL FOUNDATIONS

This paper is about the history of Artificial Intelligence (AI) applied to games. It is appropriate to begin with a definition of AI. This is surprisingly difficult as AI means different things to different people. In the seminal text *Artificial Intelligence: A Modern Approach* (Russell and Norvig 2020), now in its 4th edition, the authors present four alternatives based on thinking/acting and humanly/rationally before settling on the rational agent approach. In this view an AI is represented as a rational agent that acts so as to achieve the best expected outcome in some situation. While this definition is relatively broad, for the purpose of this analysis an AI is an agent that seeks to win a game while operating within the rules of that game. Whether the agent acts or thinks how a human would is of no concern.

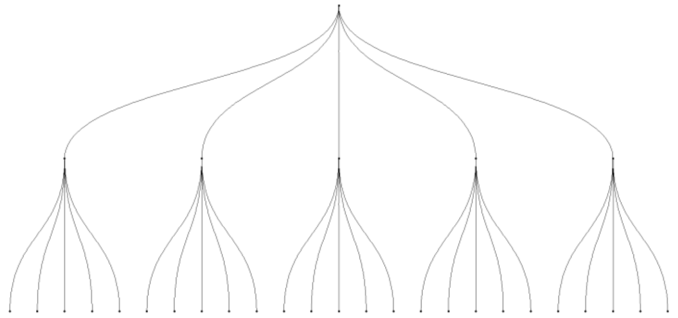
Learning is often considered a sign of intelligence, but by our definition of AI, learning is not required. Some of the AIs we will examine learn on the job by playing games, others only do what their creators explicitly tell them. As we will see, several of the most prominent and successful game playing AIs had limited learning capability. Others start off with only the capacity to learn and develop all their abilities through experience.

There are many different techniques that can be applied to develop game playing agents, individually and in combination. We will discuss the details of each approach as we review the specific models, but in this section, we review at a basic level some of the common approaches we will encounter.

Adversarial Search

Adversarial search is an approach well suited to competitive games that have two or more agents competing to win. This topic is covered extensively in Russell and Norvig (2020). Adversarial search is often applied in two person games where players make discrete moves in an alternating fashion. Under this approach the game can be modeled as a tree, with each branch representing a ply, or a move by a single player. A simple example is shown in Figure 2.

Figure 2- Tree Diagram



This graph shows two plys of a two-player game where each player has five possible moves, generally referred to as the branching factor (b). This simple tree has 25 terminal nodes, but an actual tree will be much larger, based on the average depth (d) of the game. The number of terminal nodes is approximately

$$T \approx b^d \tag{1}$$

Conclusively determining the winner of the game generally requires searching to the end of the game. But for most games of any complexity the number of nodes is far too large for an exhaustive search. Various techniques can be employed to limit this search by reducing the breadth of the search, the depth of the search, or both. In some situations, Monte-Carlo techniques are used to randomly search a portion of the tree.

Artificial Neural Networks and Deep Learning

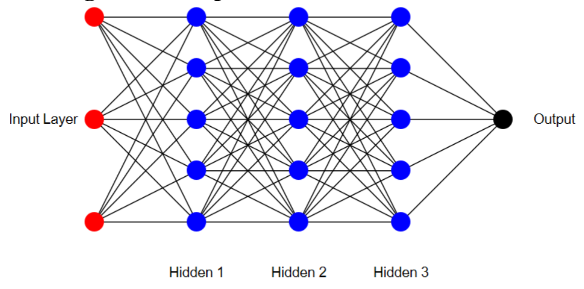
Artificial Neural Networks (ANNs), a computer model architecture based loosely on the structure of the human nervous system, were first proposed in the 1940s (McCulloch and Pitts 1943). An early implementation of an ANN was Rosenblatt's Perceptron, which was implemented on an IBM 704 at Cornell in 1957 (Rosenblatt 1958). Despite significant early enthusiasm for ANNs, interest in the approach has ebbed and flowed over the years. While some researchers successfully used shallow networks, it wasn't until major technical issues were overcome in the 2010s that deep learning emerged as a dominant force in AI. A 2006 paper introduced a method for successfully training deep networks (Hinton, Osindero, and Teh 2006) and the 2012 AlexNet paper (Krizhevsky, Sutskever, and Hinton 2012) introduced innovations that showed ANNs could significantly outperform other approached for image recognition. A rapid succession of other key papers demonstrated excellent performance on a number of machine learning tasks (Hinton and Salakhutdinov 2006; Graves, Mohamed, and Hinton 2013; Sutskever, Vinyals, and Le 2014;

Vinyals et al. 2014; He et al. 2015) and since then, ANN based architectures have been the dominant approach in AI.

ANNs are built by combining individual neurons. A neuron accepts multiple inputs, performs a linear combination, and then applies a non-linear transformation called an activation function. Common activation functions include the Rectified Linear Unit (ReLU), a piecewise linear function defined as *max(x,0);* the Sigmoid function, an S-shaped curve that maps inputs to the range [0,1], often used to model probabilities; and the Softmax function, which converts a vector of raw scores into a discrete probability distribution.

Deep Learning generally refers to architectures with multiple layers, where the output of each node in one layer is an input to every node in the next layer. Such networks are said to be fully connected. Figure 3 illustrates a very simple deep network with three inputs, three fully connected hidden layers, and a single output node. The single output is applicable to regression or binary classification models. In a multi-nominal classification model, the output layer would have one Softmax node for each potential class.

Figure 3- Deep Artificial Neural Network



Modern deep networks are generally much larger than the simple example previously discussed, with more layers and many more nodes. Since each node has a parameter (or weight) associated with every input, as well as a bias term, deep networks can easily have hundreds of thousands, or even millions of parameters that must be set during the training process. The benefit of a deep network is its ability to learn very complex transformations of the input data to the output data. Hornik, Stinchcombe, and White (1989) proved that deep networks with enough layers can approximate any continuous function on a compact subset of $\mathbb{R}^n$. Deep networks are thus very effective in identifying patterns in input data.

Fully connected networks remain extremely popular, but they have also been extended with other architectures. Convolutional Neural Networks (CNNs) implement a form of spatial invariance, making them popular in image processing applications. A CNN forms a filter that effectively scans an image, looking for patterns that may appear anywhere. Recurrent Neural Networks (RNNs) implement a form of short-term memory and are therefore popular for data with sequential dependencies, such as text or time series data. A Long Short-Term Memory (LSTM) network is a special type of RNN that can learn long-term dependencies within data. These architectures are often implemented along with a fully connected network. In image processing, for example, a series of CNNs may be used to identify features in the data that are then fed into a fully connected network for classification.

The newer ANN architecture is the Transformer. First introduced by Vaswani et al. (2017) as a sequence-to-sequence model for machine translation. The Transformer utilizes a mechanism known as self-attention that allows it to handle sequential data more effectively and efficiently. The Transformer has become the core of popular generative models such as ChatGPT and has led to significant improvement in many aspects of natural language process. For more details on the technical and mathematical foundations of Deep Learning see Goodfellow, Bengio, and Courville (2016). Chollet (2021) provides a practical introduction to implementing deep learning using

Python and the popular Keras library. Stevens, Antiga, and Viehmann (2020)is a similar guide based on the increasingly popular PyTorch library.

Reinforcement Learning

Reinforcement Learning (RL) is a set of machine learning techniques used to create intelligent agents that learn by interacting with their environment. RL stands as a third form of machine learning, distinct from supervised and unsupervised learning. It has played an important role in the development of AI in general and a particularly significant role in game-playing AIs. Sutton and Barto (2018) is a popular and comprehensive text that covers all aspects of RL.

Reinforcement learning can be described as a system where an agent interacts with its environment by taking actions and receiving rewards. The agent learns to make decisions to maximize its long-term reward. A state vector S_t, represents the environment at each time *t*. The primary elements of the RL system are

- Policy: the agent's action for a given state of the game, denote as π.
- Action: the move made by the agent, denoted as a_t.
- Reward Signal: the value achieved from taking an action, denoted r_t.

In a board game system, the agent interacts with the environment in discrete moves, based on the action a_t selected at time *t*, the state of the system, which represents the position of pieces on the board and potentially other critical information, transitions from s_t to s_{t+1}. Based on the change in state the agent receives a reward r_t. The reward indicates the immediate reward, a value function specifies the long-term reward. Roughly speaking the value of a state is the expected reward the agent will accumulate from that state in the future. Rewards are revealed directly by the environment, whereas values are estimated from the sequence of observations made by the agent overtime. The actions made by the agent seek to maximize the value function. The action-value function, *Q(s,a)*, represents the expected return of taking action *a* from state *s* and following a certain policy π thereafter. The function can be defined as

$$Q_\pi(s,a) = E_\pi\left[R_t \mid S_t = s, A_t = a\right] \qquad (2)$$

Where R_t is the cumulative sum of rewards from *t* onward. The optimal action-value function $Q^*(s,a)$ gives the maximum expected return achievable from a state *s* by talking action *a*, and following policy π.

RL systems can also be classified as value-based or policy-based. In a value-based system the agent attempts to learn the action-value function and derives its policy indirectly. For example, if an agent has learned the action value function, it may implement a greedy policy, choosing the action that leads to the highest Q-value. Conversely, in a policy-based system the agent learns the policy that maps states to actions directly without first learning a value function.

RL can also be categorized as on-policy or off-policy. On-policy methods refine the policy while following it, learning and improving the policy as the agent interacts with the environment. Off-policy methods use a different policy during the learning process. Off policy methods are often used in conjunction with value-based methods. The agent may use an exploratory process to visit a variety of states as it attempts to learn an accurate value function.

Finally, an RL system can either be model-based or model-free. In model-based RL, the agent has a model of the environment, allowing it to make inferences and predictions about future states and rewards based on its actions. In contrast, model-free RL does not have an internal model of the environment; instead, the agent learns about the environment through trial and error, directly associating actions with rewards without predicting future states.

Deep reinforcement learning is an approach that combines RL and deep learning. Deep RL is particularly useful in systems with complex, high-dimensional input data. The deep neural network may be used to implement the value function, the action-value function, or the policy function. Deep RL has become increasingly popular with the recent resurgence in deep learning.

Game Theory

Game theory is the study of strategic interactions among rational decision-makers, where the outcomes for each participant depend on the choice of all involved. Game theory has a long history in the study of economics dating back to the book *Theory of Games and Economic Behavior* (Von Neumann and Morgenstern 1947). John Nash would win a Nobel Prize for developing the concept of the Nash equilibrium in non-cooperative games (Nash 1950). Many texts have been written on game theory, notably Fudenberg and Tirole (1991). While games theory is often applied in economic and political decision making, it can also be applied to games such as chess, go, or poker.

In game theory players make decisions that result in outcomes with different payoffs. Players seek to maximize their payoffs. Players implement strategies, that is a complete plan of action for every possible situation in the game. In a pure a player always makes the same decision for a given situation. A mixed strategy is a probability distribution over a set of possible actions. A key concept in game theory is the Nash equilibrium, a set of strategies where no player benefits by unilaterally changing their strategy if other players do not change their strategy. At an equilibrium condition each player's strategy is optimal given the other player's strategies.

The quintessential example of game theory is the prisoner's dilemma. Here we outline the version presented in Fudenberg and Tirole (1991) and present the game in normal form in Figure 4.

Figure 4- Prisoner's Dilemma

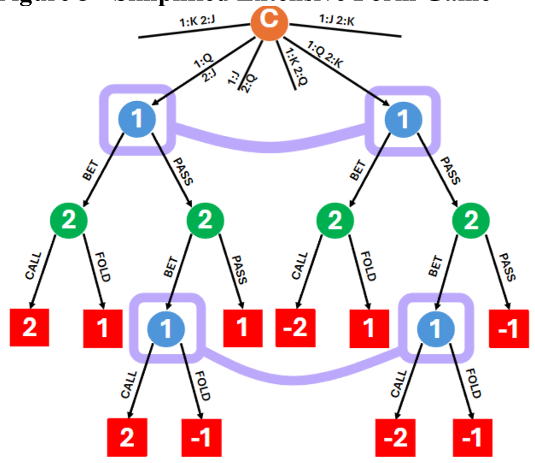
		Player 2	
		C	D
Player 1	C	1,1	-1,2
	D	2,-1	0,0

In this game two players have been arrested for jointly committing a crime. Two different strategies are available to each player, cooperate and keep quiet, or defect and implicate the other player. The matrix shows the 4 possible results based on each player's strategy, and the resulting payoffs for player 1 and 2. The police offer each criminal a deal, if they defect and implicate the other, they go free and get a reward provided the other player does not. If they both cooperate (refuse to testify) they both go free. If they both defect, both go to jail. The maximum collective reward occurs if the players cooperate and refuse to testify. However, each player can improve their individual outcome by defecting regardless of what the other player does, so the Nash equilibrium of the game is for both players to defect. In the prisoner's dilemma a Nash equilibrium is achieved with a pure (deterministic) strategy. In other games a Nash Equilibrium is only achieved with a mixed (probabilistic) strategy. The Nash equilibrium for the game rock – paper – scissors has each player select one of the three moves with equal probability.

Figure 4 shows a game represented in normal form. Games that involve sequential decisions are more commonly represented in extensive form, which is similar to the game tree format shown in Figure 2. The extensive form illustrates the moves by each player and includes chance nodes for games with stochastic components, such as the result of a die roll or a card draw.

An important concept in games of imperfect information is the information set. An information set is a collection of nodes that represent the player's knowledge at a certain point in the game. If a player is at an information set, they know they are at one of the nodes in that set, but they do not know which specific one. For example, in a card game, a player's information set includes their own cards but represents several possible nodes based on the opponents' cards. Extensive form game trees may connect the nodes in an information set with dashed lines. However, in practice, these trees are often too large to view in anything but a simplified form. Figure 5 shows the extensive form for a greatly simplified card game reproduced from Bowling et al. (2015).

Figure 5 - Simplified Extensive Form Game



In this game there are only three cards, a Jack, a Queen, and a King. The first node is a chance node, indicated by a C. The tree illustrates branches where player 1 is dealt a queen, and player 2 is dealt either the Jack or King. Player 1 has the first decision, indicated by the 1 in the node, and must either bet or pass. The game exists at one of the two nodes at the first level but player 1 does not know which, so these two nodes are in the same information set. If player 1 bets, player 2 can either call or fold. If player 1 passes, player 2 can either bet or pass. Only if player 2 bets does player 1 face a second decision, and again both those nodes are in the same information set. The terminal nodes, squares, represent the final outcomes from player 1's perspective. Player 1 will lose if he folds, or if player 1 has the king and calls. Player 1's outcome can range from a gain of 2 units, to a loss of 1 unit and depends both on the outcome of the chance node and the decision of player 2.

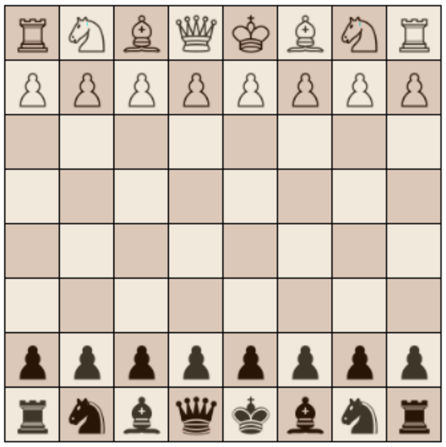
AIs that play games of incomplete information typically seek to determine an approximate Nash equilibrium of the extensive form game. This is defined as a strategy where neither player can improve their expected outcome by unilaterally changing their strategy. The most common approach for achieving this goal is based on a variation of the counterfactual regret minimization algorithm (Zinkevich et al. 2007). This iterative algorithm seeks to minimize the regret each player faces for masking a specific move compared to the counterfactual outcome of making a different move at each information set.

These simple examples show how a player must consider what decision his opponent must make when crafting his own strategy. Game theoretic concepts can be applied to a wide range of game scenarios, but they are particularly useful in games with incomplete information such as Poker, Stratego, and Diplomacy.

4 CORE CONCEPTS – CHECKERS, CHESS, AND OTHELLO

Chess is a deterministic, two player game of perfect information. It is played on an 8x8 grid of alternating colors. Each player has 16 pieces of 6 different types; king, queen, rook, bishop, knight and pawn, initially positioned as in Figure 6 ('Rules of Chess' 2025). Each piece type has a unique set of allowable moves. Players take turns moving and may capture opponent pieces by occupying the same square as the opponent piece. The objective of the game is to attack the opponent's king, placing them in a condition from which there is no escape, known as a checkmate.

Figure 6- Initial Chess Layout



Chess is one of the most widely studied problems in early AI. John McCarthy contributed a chapter titled *Chess as the Drosophila of AI*, in the book *Computers, Chess, and Cognition* (Marsland and Schaeffer 1990). This analogy likens chess to the fruit fly (Drosophila) in genetic research, highlighting its role as an ideal experimental platform.

The foundational concepts for chess playing programs was laid out in Shannon (1950). Written in the late 1940s, shortly after the development of the first electronic programmable computers, Shannon did not develop an actual program for playing chess, but his paper did define the key components that would be included in future programs, not just for chess, but for games in general. Shannon defined the *position*, or state of a chess game as:

1. The position of all pieces
2. The side with the next move
3. An indicator if castling is still available
4. The previous move
5. The number of moves made since the last pawn move or capture.

Most of these items are obvious, but some of them relate to the more esoteric rules of chess, such as the en passant capture or the 50-move draw rule.

Given that chess is a deterministic game of perfect information, under the assumption of perfect play any given game state must be either a win for white, a draw, or a win for black. In theory an evaluation function *f(P)* maps the current position to the final outcome. It is possible in principle to construct a program that would execute perfect play, selecting a move that leads to the best possible outcome for the player from any given state. A tree can be constructed that includes every possible move to the conclusion of the game. The outcome can be folded back, assuming optimal decision making by each player to determine the best move. However, given that there are on average about 30 legal moves and a game lasts about 40 moves, the tree has approximately 10^{120} branches making this mapping unworkable. Shannon points out that a machine executing one move per microsecond would require 10^{90} years per move. Modern computers are several orders of magnitude faster, but the calculation is still impractical.

Shannon then outlines an alternative programming approach, keeping a dictionary of all possible game states. The dictionary would define the perfect play move from that position. In addition to the impracticality of defining the perfect play move as defined above, the number of possible board states, approximately 10^{43}, makes even a dictionary of near optimal moves impractical. Shannon concludes that developing a program to play perfect chess is not practical, and developing a program to play legal chess is trivial and pointless, as random moves lead to extremely poor outcomes. He then frames the problem as defining a playing strategy, choosing a move from a given position, that can play comparable to a good human. It would take nearly 50 years, but programs would eventually be developed that would exceed the capabilities of any human player. Shannon outlined the basic components of this strategy, a minimax search through a partial game tree using an approximate evaluation function that would eventually lead to superhuman performance.

Partial Evaluation Function

Expanding the tree to termination results in a tree that is too large to evaluate, but determining the final outcome of the game requires full enumeration. The alternative outlined is to develop an evaluation function $f^*(P)$ that estimates the value of any arbitrary position on the board. The program can then generate a partial tree by searching forward a few moves and evaluate the resulting position at that point. Those values can then be folded back to determine the best move at the current position. This provides a heuristic assessment of the best move to make at any given point. The quality of the heuristic depends on two features, the depth of the search and the accuracy of the evaluation functions.

Shannon asserts that "*Most of the maxims and principles of correct play are really assertions about evaluating positions."* He articulates several evaluation heuristics including the relative value of pieces, the notion that rooks are most valuable on open files and that an exposed king is a weakness. Each of these factors can be given a numeric score that is combined in a weighted fashion to give an overall positional evaluation. The exact composition of these rules, and the method for combining them, based on human expertise would be a foundational concept in chess programs for the next 60 years. Deep Blue, the IBM program that eventually defeated Gary Kasparov, would invest heavily in developing an accurate and efficient evaluation function.

Minimax Search with Pruning

For any given state of the game, the program's task is to select the next move. This can be accomplished by searching through the game tree and assuming optimal play by each player. Each position on the tree is given an evaluation score based on the evaluation function. Assume each score is relative to White, with a higher value associated with a superior position for White. (Black's relative score is the negative of white's).

White will should pick the move that maximizes the evaluation score and assume that Black will respond by making a selection that minimizes white's score. The process can be illustrated through the simplified tree shown previously shown in Figure 2 In this tree we project forward 2 plys (one for white, and one for black) under the assumption that only 4 moves are available at each ply. Evaluation scores, relative to white, are calculated for the 16 leaf nodes. Black's move for each set of leaf nodes is selected to minimize white's score. White picks from the four possible moves to maximize its score. White picks the third move and assumes black will pick the third response.

Shannon defines a Type A strategy as one which considers all possible moves from each position for a fixed number of moves. Recognizing the weakness of this approach, he proposes an alternative Strategy B that conducts a variable search. The search is variable in depth as it will continue to progress deeper if the position is unstable, for example if it ends with a piece about to be captured. The search is also variable in terms of breadth, implementing some process to ignore inferior branches, a form of what would come to be known as pruning, though the details of how to achieve that pruning are limited.

Games Stages

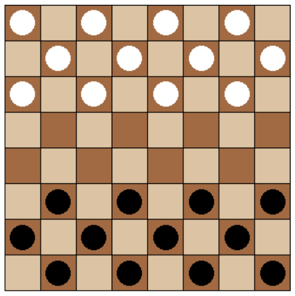
Chess theory has generally divided the game into three distinct stages, the opening, the middle game, and the end game. According to Shannon, the opening stage generally lasts about 10 moves. Skilled chess players generally implement a pre-defined opening book picking from a large set of widely analyzed openings. The end game refers to the portion of the game where only a few pieces are left on the board. Like openings, end games have been widely studied and documented with explicit strategies defined based on the set of pieces remaining. Shannon advocates using different tactics, and indeed different programs, during each stage of the game. Shannon's work is focused squarely on the middle game, the most complex portion of the game, where positional understanding is most critical. The segregation of the game into three distinct phases is a concept that would be firmly embedded into virtually all chess programs up to and including Deep Blue, the program that defeated Kasparov.

Shannon's paper merely outlined the concepts behind a theoretical chess program. Given the limitations of computation of the time he did not actually develop a program. The first known chess programs are described in Kister et al. (1957) and Bernstein et al. (1958). Those 2 programs are compared and contrasted in another paper co-authored by two the original Dartmouth conference attendees, Allen Newell and Herbert Simon (Newell, Shaw, and Simon 1958).

Checkers

Given the computational challenges associated with chess, progress on a more human-level competitive solution began with checkers. Like chess, checkers is played on an 8x8 grid. Each player has 12 pieces initially arranged as shown in Figure 7. All 12 pieces are identical and can only move forward diagonally until they are kinged; i.e., they reach the last row. Once kinged a piece can move forward or backward. Players capture opposing pieces by jumping them. A player wins all his opponent's pieces have been captured or have no legal moves ('Checkers' 2025). With a branching factor of approximately 8 to 10, and average game length of about 50 moves, checkers has a game tree complexity of approximately 10^{40}, still very large, but much smaller than chess.

Figure 7 - Checkers Board

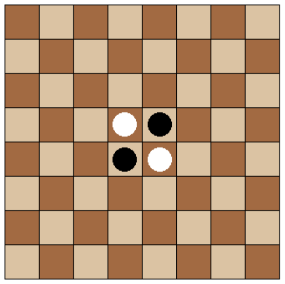


IBM researcher Arthur Samuel developed the first AI capable of playing a complex game with human level performance (Samuel 1959, 1967). His program was limited by the relative capabilities of computers in the 1960s, but he developed an innovative program that introduced several key techniques, including alpha-beta pruning, self-play and learning. The program executed a variable depth search, searching deeper where moves left the board in an unstable position, perhaps where a jump is possible. This technique would later come to be referred to as a quiescence search. An evaluation function would score the value of each board position based on hand-crafted features such as piece count, piece position, etc. More than 20 factors were combined using a weighted average. The program used two methods of learning. In rote learning it stored board positions and their outcome in a lookup table, in generalization it modified the evaluation function weights based on the eventual results of the game. While Samuel's checkers program never achieved world-class performance, it operated as a strong amateur, it did establish a foundation for AI applications in games, and demonstrated the potential of AI in general.

Othello

Another board game used for AI research during this period was Othello, also known as Reversi. Othello is played on the same 8x8 grid as checkers, and players have 64 identical pieces, white on one side black on the other. The game begins with 4 pieces on the board with the pattern shown in Figure 8. Players alternate turns, adding pieces to the board with their color facing up. When a newly placed piece outflanks an opponent's piece by forming a straight line with another piece of the same color, all opponent's pieces in between are flipped to the player's color. When all squares are filled, or no legal move remains, the player with the most pieces wins ('Reversi' 2025). Like chess or checkers, Othello is a deterministic game of perfect information. The branching factor varies significantly over the course of the game, but the game length is fixed at 60 moves or less. Given that, the complexity of the game is approximately 10^{58}.

Figure 8- Othello Board

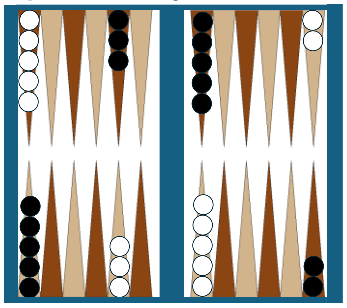


Paul Rosenbloom of Carnegie Mellon developed an Othello program named IAGO that defeated 19 other Othello programs in a 1981 tournament (Rosenbloom 1982). IAGO implemented a knowledge heavy AI approach, with human expertise explicitly encoded into the program. Because of the explicit time constraints of the game, and the computational limits of the time, IAGO develops a time budget for each move. It used an α-β search algorithm and a hand-crafted evaluation function. By carefully selecting the order of branches to explore, IAGO achieved an average search depth of 6.3 moves. IAGO also explicitly modeled the opening, middle and end games and modified its strategies based on the phase of the game. IAGO defeated other Othello programs with a record of 10-0. Unable to find a willing champion level human opponent, the program was evaluated against championship level human games move by move, making the same move as the human 59% of the time.

5 BASIC REINFORCEMENT LEARNING – BACKGAMMON

Backgammon is another popular two player board game. Backgammon is played on a board with four quadrants, each of which has six points. Each player has 15 pieces arranged as shown in Figure 9. Players move in opposite direction, in this example black moves clockwise and white moves counterclockwise. Players move their pieces based on the roll of two dice, where each die authorizes a specific move. So, for example, if a player rolls a five and a two, they may move one move of five spots and another move of two spots using the same piece or a different piece. Any point occupied by two pieces is blocked from the other player. A point with only one piece is vulnerable. If an opposing piece lands on a vulnerable piece that piece is hit and removed from the board. It must re-enter the board from the beginning. Players win by clearing all their pieces from the board first. Backgammon has a unique feature known as a doubling cube. At any time, a player may double the stake of the game by invoking the cube. The other player must either accept the new stakes or concede the game at the original stakes. The doubling player cannot double again, but the other player may choose to double again later in the game. There is no limit on the number of doubles that can be made ('Backgammon' 2025). Backgammon is a game of perfect information, but due to the dice it is stochastic. The stochastic nature of the game greatly expands the complexity tree. Since the order of the dice does not matter, there are 21 unique rolls to consider in backgammon. The dice adds 21 branches at each ply creating a very large complexity tree of about 10^{263}.

Figure 9- Backgammon Board



In the late 1980s Gerald Tesauro, a researcher at IBM, created a series of backgammon programs that would eventually achieve world-class level play. His first program Neurogammon introduced a neural network architecture. The program won a computer backgammon tournament in 1989. He would later develop a program called TD-Gammon that would achieve world-class level play (Tesauro 1992, 1994, 1995, 2002). TD Gammon is explicitly based on a form of reinforcement learning known as temporal difference learning, specifically the TD(λ) version of that algorithm. Temporal difference learning was first introduced in Sutton (1988). TD Gammon used self-play and RL to learn a value function that maps game state to expected outcome based on the difference in value achieved with each move. The program then selects the best move based on what move leads to the highest value state based on a 2 ply search. TD Gammon achieved master level human capability, losing to top level human players in very competitive matches.

6 DEEP BLUE - WORLD CHAMPION CHESS

While chess was a topic of great interest in the early days of AI, it was not until the 1990s that AI conquered chess. In 1997 an AI known as Deep Blue defeated world champion Gary Kasparov in a widely covered match that captured public attention and was a major turning point in the public perception of AI.

Deep Blue was the latest in a series of chess programs; Chip Test, Deep Thought and finally Deep Blue, developed by a team that started work at Carnegie Mellon and later moved to the research lab at IBM. The development effort is chronicled in the academic press in Campbell, Hoane, and Hsu (2002) and the popular press in Hsu (2002). Deep Blue implemented a game tree search algorithm not dissimilar to that described by Shannon in 1950. But Deep Blue was backed by massive computational resources, well beyond anything conceivable in 1950.

Deep Blue consisted of a 30-processor cluster supported by 480 custom designed chess chips. The custom chess chips could perform search and evaluation operations in hardware. They are described in detail in Hsu (1999). Using a combination of software and hardware-based searching, the program could evaluate about 100 million nodes on the search tree. The search was highly non-uniform, using advanced pruning techniques to limit the portions of the tree explored, as well as quiescence and other search extension techniques to explore key portions of the tree more deeply. With these modifications Deep Blue could reach search depths of 30-40 plys.

In addition to the ability to search massive portions of the game tree, Deep Blue strength came from its ability to encode substantial human expertise. The hardware-based evaluation function recognized over 8,000 patterns and assigned each position a score on each one. Overall position strength is a weighted combination of these features. Deep Blue also divided its game into opening, middle and closing stages. The opening book was manually crafted by human grandmasters and covered 4,000 positions. Deep Blue also created an extended book for the opening phase that utilized a database of 700,000 games played by grandmasters. The use of this database is further discussed in Campbell (1999). The extended book would bias moves toward those used by grandmasters. Deep Blue also included a database of all endgame positions with five or fewer pieces, and selected positions with six pieces.

Overall Deep Blue's triumph over Kasparov was a watershed moment in the history of game playing AI, and AI in general. It was the first demonstration that an AI could play an intellectually challenging game at a level higher than any human. Kasparov discusses his experience playing Deep Blue on (Fridman and Kasparov 2019). From a technical perspective, Deep Blue was a stellar example of good old-fashioned AI. The program used massive computational resources to implement human specified skills. Despite the qualifier Deep in its name, Deep Blue did not include a deep learning architecture and in fact the program did learn at all. It only got better in between games as its creators fine-tuned the algorithm.

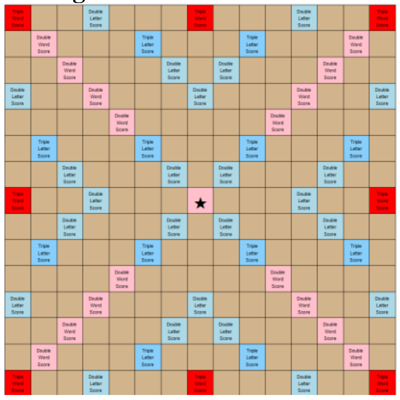
7 NLP – CROSSWORDS, SCRABBLE AND JEOPARDY

Up through the 1990s AI made significant progress in developing skills in increasingly complex board games, culminating in the defeat of the world champion chess master. The early 2000s saw less activity in game playing AI overall, and a switch from classic board games to word games. In this time period AI demonstrated human and super-human level skills in crossword puzzles, scrabble, and most notably the popular question and answer game show Jeopardy. Together these applications integrated information retrieval and Natural Language Processing (NLP) concepts into game playing AIs.

Some of the earliest successes in word-based games came in the form of crossword puzzles. A crossword puzzle is a word game where players fill in a grid of interlocking blank squares with words based on clues provided. A team of researchers from Duke, that later moved to industry, developed Proverb, an AI that could solve crossword puzzles (Keim et al. 1999; Littman, Keim, and Shazeer 2002). Proverb effectively uses a two-step process to solve a crossword puzzle. A series of *Expert Modules* develop and rank candidate solutions to the clue, then a solver takes these weighted lists and iteratively searches for the best solution to satisfy the grid constraints of the puzzle. Candidate generation is an information retrieval process; some 30 independent modules generate candidate solutions and assign each a probability score. Experts implement a wide range of techniques, including generating lists of words with the correct length from a dictionary, searching a database of 350,000 solved crossword clues, searching online data sets like the Internet Movie Database (IMDB), and using linguistic approaches such as latent semantic indexing. Proverb was able to achieve high level human, but not superhuman, performance. Over a test set of 350 puzzles, it achieved word accuracy of 95%, letter accuracy of 98%, and solved 46% of the puzzles perfectly. But in a competitive tournament, Proverb placed 190 out of 251.

A further refinement of word-based game AI involved the board game Scrabble. Scrabble is a word game in which two to four players compete to create words in a crossword puzzle like fashion on a 15x15 board as shown in Figure 10. Players maintain a *rack* of seven tiles each, drawn from a bag of 100 tiles. Each tile has a single letter and an associated point value. Players score by placing tiles on the board to spell acceptable words, from an agreed upon list or a dictionary. The initial word must cover the star at the center of the board, and each subsequent word must connect to at least one word already on the board. The points earned are the sum of the scores for each tile used, factored by any of the multipliers shown on the board. Multipliers include double or triple letter score, and double or triple word score ('Scrabble' 2025). Because players are unaware of the tiles in each opponent's rack, Scrabble is a game of imperfect information.

Figure 10- Scrabble Board

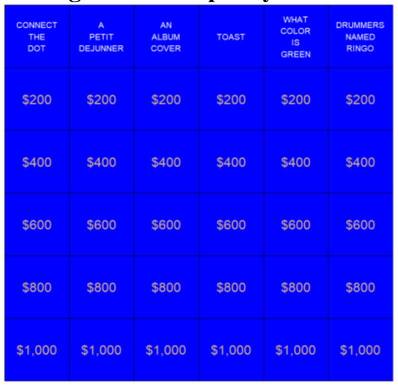


An AI named Maven was able to achieve superhuman performance in Scrabble (Sheppard 2002). Maven includes a move generation component that begins with a dictionary to determine all legal moves for a given board and rack. Each move is processed by an evaluation function that considers not only points scored, but the quality of the remaining rack. The evaluation function was learned through simulated play. As in other games such as chess, Maven divides the game into three phases, the normal game, pre-endgame, and endgame, and uses a different search approach in each phase. The pre-endgame begins when the game can end within the two-ply search horizon. The endgame begins when all tiles have been removed from the bag, at which point Scrabble becomes a game of perfect information; the opponent's rack can be inferred from the tiles on the board and in Maven's rack. End game play is important as going out first, emptying your rack, has significant scoring advantages including an extra turn and a tile penalty for the tiles left in the opponent's rack. In the endgame Maven switches to the B* algorithm, an enhanced form of alpha-beta pruning.

Maven was able to perform at a level equal to or better than top-level humans. Maven averages about 35.0 points per move, vs, about 33.0 points for top humans. Maven maintained a 35-15 record in tournament play against top humans, including victories over the reigning US and world champions in 1998.

While Proverb and Maven demonstrated the ability of AI to excel in NLP, the most notable achievement of AI in word-based games involved the game of Jeopardy. Jeopardy is a popular television game show that originally aired in 1964. It changed formats several times, and was off-air for some time, but the current version has aired continuously since 1984. Three contestants participate in a question-and-answer game, picking topics from the game board, an example of which is shown in Figure 11. There are five questions in six categories with each question having a different dollar value. The example shown is for the first round. The second *Double Jeopardy* round has double the value for each question. The categories change from round to round and game to game. A player selects a square and is given an answer, to which they must respond in the form of a question that elicits the answer shown. Players may buzz in only after the question has been fully read, an early buzz locks them out for a short period of time. The player who buzzes in first can answer. A correct answer earns the dollar value of the square, an incorrect answer results in a deduction of the same amount. One square in the first round and two squares in the second round are daily doubles. The person selecting the daily double is given the exclusive opportunity to answer the question. Unlike regular questions the dollar value of a daily double is a wager from the player before the answer is revealed. Players may bet up to the full amount of their bankroll, or a minimum by round if their bankroll is small. Because of the high dollar value potential of the Daily Double these questions often play an outsized role in the competition. The game concludes with a Final Jeopardy round in which all players wager on the same question. At the time of the wager players know the category, but not the specific question. ('Jeopardy!' 2025).

Figure 11- Jeopardy Board



CONNECT THE DOT	A PETIT DEJUNNER	AN ALBUM COVER	TOAST	WHAT COLOR IS GREEN	DRUMMERS NAMED RINGO
$200	$200	$200	$200	$200	$200
$400	$400	$400	$400	$400	$400
$600	$600	$600	$600	$600	$600
$800	$800	$800	$800	$800	$800
$1,000	$1,000	$1,000	$1,000	$1,000	$1,000

The player with the highest score at the end of the game is the winner. Winners return for the next episode. Skilled players can go on long winning streaks and earn substantial prizes. Current host Ken Jennings began as a player and set the record with 74 consecutive victories and over $2.5 million in winnings. Jennings would later be part of the live competition against the AI system Watson developed by IBM.

Jennings winning streak is said to be the inspiration for IBM to take on the challenge of developing an AI program to play Jeopardy. Looking to build on the publicity IBM had earned from Deep Blue, they began development of Watson in 2007 and the competition against Jennings and another champion took place in 2011. The full history of the effort is provided in a non-technical fashion in the book *Final Jeopardy* (Baker 2011) and the IBM project leader discusses the project in detail on Fridman and Ferrucci (2019).

Watson is a highly sophisticated and modular set of computer programs, capable of playing Jeopardy in real time at a world class level. The components of Watson fall into two broad categories, question answering and game strategy. Question answering involves NLP based tasks to parse the question, determine what it is asking, find a set of candidate answers, and assign probabilities to the candidate answers. Game strategy involves a range of decisions such as when to buzz in, what square to pick when controlling the board, and how to wager on Daily Double and Final Jeopardy questions. The question answering components of Watson are described in Ferrucci et al. (2010). Gene Tesauro, the developer of the TD Gammon application, was a team member on the Watson team and he co-authored two papers describing the game strategy of Watson (Tesauro et al. 2013; Tesauro et al. 2012).

The question answering component of Watson is referred to as DeepQA (question-answer). DeepQA was a sophisticated information retrieval and natural language processing application. Like its predecessor Deep Blue, Deep QA was not a deep learning-based system. It was developed to operate on a massive corpus of data download and pre-processed for easy access. Watson did not have access to the internet in competition. DeepQA began by parsing the question, which was provided as a direct text stream. The first challenge was to identify the lexical answer type (LAT), i.e. the specific type of answer expected for a given question, which given the unusual and complex phrasing of clues is especially difficult in Jeopardy. The system used many expert modules to generate a series of candidate answers, a process referred to as hypotheses generation. Candidate answers pass through a soft filtering process that reduces the list to about the top 100 answers. Those answers then are processed through an evidence scoring process to assess the validity of the potential answer. Answers are merged and processed through a model trained on a data set of known questions and answers to assign a confidence level to each potential answer. In the final competition Watson would display the top answers along with the confidence levels. Early versions of Watson would take up to two hours to answer a single question, but with tuning and massive parallelization Watson could develop its ranked answer list in 3-5 seconds.

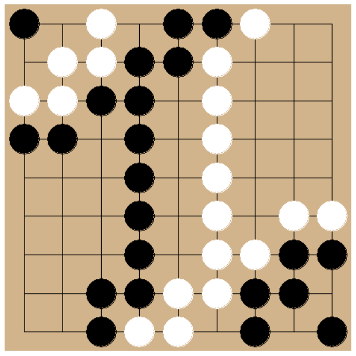
Given the answer list the first strategic decision is when to buzz in, a decision Watson made based on confidence level thresholds. Other strategic decisions Watson needs to make include what square to select next when controlling the board, how to wager on Daily Doubles, and how to wager in Final Jeopardy. Strategy development was facilitated by extensive statistical analysis of previous Jeopardy games and detailed game simulations. Game analysis revealed both the criticality of the Daily Double in the final outcome of the game, and the fact that Daily Double squares are not distributed randomly, they are most common in the bottom half of the board, are more common in some columns than others, and the two second round doubles are never in the same column. Like many top human competitors, Watson prioritizes double hunting in its cell selection strategy, but unlike humans it can use more than simple heuristics and estimate detailed probabilities. Much of Watson’s strategy was refined through detailed simulation. Interestingly, Watson kept question answering out of the strategic simulations, it simply assigned probabilities to correct answers. Watson based its wagering strategy in part on its confidence in the category, based on previous questions answered in that category, as well an assessment of the category title, and the relative ratios of player scores at the time of the wager. The IBM team implemented a deep reinforcement learning model to determine daily double bets. The model, based on Tesauro’s backgammon work, estimated the win probability from the game state. Watson could determine an optimal bet based on its confidence in the category and the probability of correctly or incorrectly answering the Daily Double question. Interestingly, the team found the risk level associated with optimal bets to be quite high and implemented a modification to make a sub-optimal but lower risk level bet. For Final Jeopardy wagering Watson would estimate its confidence in the category based on the category title and use that to develop a game theoretic best response strategy based on its database of (suboptimal) human betting.

Watson was ultimately tested in competition against humans. While only its final two game match against world champions was televised, Watson sparred experienced humans in 130 games. In the first round Watson played in 73 games against average players, winning 64.4% of the games. In the second round it played 55 games against former Tournament of Champions players, much stronger competition than the first round. Having improved its capabilities from the first round, Watson won 70.9% of the games. In the final competition against two of the most successful players ever, in a two-game match, Watson won by a large margin; $77,147 to Jennings second place score of $24,000. As was the case with the Deep Blue chess victory, Watson generated major publicity for AI. Like Deep Blue, Watson was enabled by massive compute and systems highly engineered with human expertise. With minor exceptions, for example the Daily Doubling system in Watson, these systems were non-learning systems in the good old-fashioned AI (GOFAI) framework. In the immediate aftermath of Watson, AI progress in games would dramatically accelerate. In a little more than 10 years AI would achieve superhuman performance in a wide range of more complex games primarily relying on newer technical approaches including deep reinforcement learning.

8 ADVANCED LEARNING – GO AND OTHER GAMES

After Deep Blue proved that computers could play chess at world champion levels, attention turned to the game of Go. Go is an extremely popular game with an estimated 20 million players, mostly in Asia. Go is similar to Othello, players alternate turns placing pieces, called stones, on a board attempting to out-position their opponent’s pieces and capture territory. Go is played on a 19x19 grid, but unlike Othello the pieces are placed not in the squares but at one of the 361 intersections. Figure 12 shows a simplified Go board with a 9x9 grid, a version of the game played by beginners. Players take turns placing stones on the board and capture opponent’s stones by surrounding them on all sides ('Rules of Go' 2025).

Figure 12- Simplified Go Board



Go is a deterministic game of perfect information, but because of the size of the board and duration of the game it has an enormous search tree. The branching factor for Go begins at 361, the size of the board, and decreases from there. With an average game length of about 250 moves the complexity tree for Go is estimated at more than 10^{720}. In addition, it is significantly more difficult to assess the strength of a position in Go compared to chess. Combined, these factors make Go a significantly more difficult challenge than chess for a computer program.

A large team at Google's DeepMind would take on the Go challenge. In the process their program AlphaGo would defeat Lee Sedol, one of the top Go players in the world, in a televised match with an estimated audience of over 100 million viewers as documented in the film AlphaGo (Kohs 2017). Shortly after defeating Sedol, the DeepMind team released new versions of the program, AlphaGoZero and AlphaZero, that were far better players than the original, achieving superhuman level performance. These programs represent a major milestone in the history of AI games, and AI in general. David Silver, a key figure on the DeepMind Go team discusses the effort in detail on (Fridman and Silver 2020).

Given the massive size of the search tree, alpha-beta search is of limited effectiveness in Go. Much attention has been paid to a stochastic search approach known as Monte Carlo Tree Search (MCTS). David Silver, one of the lead Go researchers at Deep Mind, co-authored a paper that demonstrated some extensions to MTCS in the context of Go in 2011 (Gelly and Silver 2011). Their program *MoGo* was able to achieve master level performance in the simplified 9x9 version of Go using these techniques. But the major breakthrough was AlphaGo presented in Silver et al. (2016). AlphaGo continued the use of MCTS, but added in several deep neural networks, and unlike the approach in previous AIs such as Deep Blue, it included very limited explicit human defined expertise in the form of rules or heuristics. AlphaGo learned how to play Go primarily by observing human play and then engaging in extensive self-play.

The first step in the training process for AlphaGo was a supervised learning (SL) task. The SL policy network $p_\sigma(a|s)$ that predicts the next action given the current state of the game. The network was a 13-layer deep convolutional neural network trained on 30 million positions from games recorded on the KGS Go Server. The convolutional architecture allowed the network to recognize patterns that might occur at different portions of the board. Based on this supervised training the network could correctly predict the next move 57% of the time. A second policy network p_p was then trained using policy gradient reinforcement learning. The network was initialized as the SL network, then refined through millions of games of self-play. The final training step focused on training a value network. This network was similar to the policy network except that instead of a softmax output of next move probabilities, it output a single predicted win probability. Once trained the networks were integrated with MCTS. The policy network provided a sub-set of high probability moves to be evaluated through a randomized search using yet another policy network called the rollout network. The rollout network implements a fast rollout policy to guide the random search process. The rollout network is a relatively simple network, but it is the one case where human expertise and heuristics are explicitly incorporated. AlphaGo assessed moves by taking a linear combination of the value network prediction and the result of the fast rollout, selecting the move with the highest value.

The end result was a system capable of playing the full version of Go at a world class level. The 2017 paper documents competition against other go programs where AlphaGo won 494 out of 495 games. It also documents a match against Fan Hui, a multi-time European champion in which AlphaGo won 5-0. After the paper was published a slightly modified version of the program defeated former world champion Lee Sedol four games to one in the highly publicized competition. Another modified version, referred to as AlphaGo Master played against top level players in 2017 winning all 60 games. AlphaGo played Go at a world-class level but would quickly be surpassed by significantly enhanced versions of the program. The next major enhancement, referred to as AlphaGo Zero, is documented in a 2017 paper (Silver, Schrittwieser, et al. 2017).

Several changes to the AlphaGo system were made for AlphaGo Zero. The policy and value networks were combined into a single integrated network with dual outputs, a next move probability and an overall win probability. The supervised learning stage was eliminated. AlphaGo Zero was trained exclusively through self-play, beginning with randomized play. It continued to use MCTS as a component of the evaluation process but no longer used a separate rollout network. AlphaGo Zero had no access to human expertise, either through heuristics or even observation, as the authors state it learned *tabula rasa* through reinforcement learning. It was only provided with the rule set for Go. Even with these limitations, AlphaGo Zero achieved an Elo rating superior to AlphaGo Master, and defeated that version 89 to 11 in head-to-head play.

The next major iteration of the program goes by the name of AlphaZero and is also documented in a 2017 paper (Silver, Hubert, et al. 2017). AlphaZero represented a major generalization of the AlphaGo Zero model. Like its predecessor it was provided with no human expertise and learned tabula rasa through self-play. It achieved superhuman performance in Go, but also in chess and shogi. Shogi, sometimes referred to as Japanese chess, is a modified and more complex version of chess played on a larger board where captured pieces change sides and maybe placed anywhere on the board ('Shogi' 2024.)

The core of AlphaZero is similar to its predecessor. It utilizes a deep neural network that takes board position as an input, and outputs move and win probabilities. Trained through reinforcement learning, AlphaZero also implemented an MCTS. AlphaZero differed from AlphaGo Zero in several respects; it implements an outcome that includes draws, and no longer exploits the symmetrical properties of the Go board, because unlike Go, Chess, and Shogi boards are not invariant to rotation and reflection. AlphaZero also streamlined and simplified the training process. While a separate instance was used for each game, each instance utilized a common architecture and hyper parameter set. Utilizing only self-play AlphaZero achieved performance superior to AlphaGo Zero in Go and other leading AI in chess and Shogi after training for less than 24 hours. The rapid learning is impressive, but AlphaZero did have access to very large-scale computational resources; 500 1st generation TPU's and 64 2nd generation TPU's, roughly equivalent to over 500,000 high end desktop CPUs.

AlphaZero represents a major milestone in not only AI applied to games but for AI in general. Not only could it develop superhuman performance across multiple games, but it also demonstrated the remarkable ability to mimic human knowledge discovery. On its own AlphaZero independently learned the 12 most common openings played by human chess players.

9 CARD GAMES – TEXAS HOLD 'EM

The games reviewed thus far have been games of perfect information, where all players know the full history and current state-examples include Checkers, Chess, and Go. In contrast, imperfect information games hide key elements of the state; for instance, a poker player cannot see opponents' cards. These games are significantly harder to solve: the state space is larger, backward induction becomes impractical, and players must reason not only about the game state but also about their opponents' beliefs. As a result, imperfect information games require game-theoretic approaches to model and solve strategic uncertainty. Major developments in game theoretic AI occurred in the context of the game Texas Hold'em.

Texas Hold'em is one of the most popular versions of the card game poker. In this game players aim to make the best five-card hand from a total of seven cards: two private cards and five community cards. Players are first dealt two cards face down unseen by other players. Then three cards are dealt face up, known and available to each player, in what is known as the flop. In the next round, known as the turn, a fourth community card is dealt face up, and in the final round, the river, a fifth face up community card is dealt. After each set of cards are dealt a round of beating occurs. Players may fold (exit the game and forfeit any bets already made), bet (initiate a new bet), call (match an existing bet), or raise (increase the current bet). Heads Up games are limited to two players, while multi-player games can have up to 10 players. In the no-limit version there is no limit on the number of raises that can be made so each round of betting has no fixed upper limit on the number of bets ('Texas hold 'em' 2025). The complex interplay of strategy, probability and psychology makes Texas Hold'em a very difficult game for humans, and an intriguing subject for AI research.

Over a relatively short time period research teams at the University of Alberta and Carnegie Mellon developed a series of increasingly sophisticated poker programs playing increasingly complex versions of the game. The Alberta team developed a program named *Cephus* that solved the Heads-Up Limit version of the game. With only two-players and bets limited to fixed increments, the game has a smaller size than other versions though it still has over 10^{14} decision points (Bowling et al. 2015, 2017).

Cephus is said to have essentially weakly solved the game. Weakly implies solution from the initial position, while essentially implies that a human lifetime of play is not sufficient to establish a statistically significant difference from an exact solution. In this context a solution is defined as a Nash equilibrium strategy for each player, meaning no player can increase their expected utility by unilaterally choosing a different strategy. A strategy for a player specifies for each information set a probability distribution over the available actions. In the case of Texas Hold'em those actions are call, fold, bet, pass, or raise. In general, the Nash equilibrium strategy is a mixed strategy, i.e. a probability distribution across these available actions.

Cephus, and several of the programs before and after, use a variation of Counterfactual Regret Minimization (CFR). Counterfactual Regret is the regret, the difference between the payoff actually received and what could have been achieved with a different action, for not having taken a particular action when it was available. It is a measure of how much better a player could have done by using a different strategy. The game is represented as a tree and the program iterates through a large number of simulated games iteratively updating the regret calculations and modifying the strategy. With enough computation the strategy evolves toward a Nash equilibrium. The team solved the game using a very substantial level of computational resources, 200 nodes deployed for 68.5 days, utilizing 900 core years of computation and 10.9TB of storage.

Following the Cephus project, the Alberta group, along with new collaborators from the Czech Republic developed DeepStack (Moravčík et al. 2017). DeepStack played the Head's-Up No-Limit version of the game. While still a two-player game, this version allows players to bet any amount which makes the game tree significantly more complex, expanding from roughly 10^{14} decision points, to about 10^{160}. Deep Stack uses deep neural networks, with 7 layers and 500 nodes each to estimate the value function for a depth-limited look ahead search. One network is trained on data available after the flop, a second on data available after the turn. The networks were trained on 1 million simulated hands for the flop network, and 10 million for the turn network with a data set that includes the pot size, the public cards and the two player ranges. The network outputs the counterfactual values, the expected value for each of the possible 1,326 hidden card hands based on the pot size, public state, and the opponent's range. Deep Stack was tested in competition with 33 professional poker players playing up to 3,000 hands per player for a total of 44,852 hands. Deep Stack did very well, defeating 10 of the 11 players who played the full 3,000 hands by a statistically significant margin. While Deep Stack demonstrated that AI could compete at an expert level in heads-up no limit, a team from Carnegie Mellon would soon develop an AI that could compete at a world-class level not only in heads-up, but also in the far more complex domain of multiplayer poker.

A team at Carnegie Mellon developed two AI programs to compete in Texas Hold'em. The program Libratus achieved world class performance in Head's-Up No-Limit (Brown and Sandholm 2018), and Pluribus achieved superhuman performance in multi-player (Brown and Sandholm 2019). These AIs are discussed in detail by the two authors separately on the Lex Fridman AI podcast (Fridman and Brown 2022; Fridman and Sandholm 2018).

Libratus was built on three main modules. The first module develops a so-called blueprint strategy based on an abstracted (simplified) version of the game, computing game theoretic solutions using the Monte Carlo counterfactual regret minimization approach. The abstraction simplified the game by discretizing bet sizes and grouping similar hand combinations. During game play the second module creates a finer grained abstraction and solves it in real time in a manner that is consistent with the blueprint strategy. The third module is a self-improver, which modifies the blueprint strategy as game play occurs utilizing the opponents play to determine what missing branches of the tree to fill out.

10 VIDEO GAMES – ATARI AND BEYOND

With major accomplishments made in both board and card games, game playing AI development began on new types of games. From 2015 through 2022 game playing AI achieved high level performance on a series of increasingly complex video games. From basic Atari games to complex e-sports games like Dota 2, Starcraft 2, and Gran Turismo, AI has achieved high levels of performance. In addressing these games, the focus shifted from discrete move games to real time video.

In 1972 Atari, Inc. released the video game Pong, one of the first ever video games. Pong, a simple two-dimensional computer version of ping-pong, became extremely popular. Atari would issue many other games including notable titles such as Space Invaders, Ms. Pac-Man and Asteroids ('Atari, Inc.' 2025). In 2015 researchers at Google Deep Mind released a study documenting an AI they developed that could play 49 different Atari 2600 games using a common architecture and hyperparameter set across all games (Mnih et al. 2015).

The program, generally referred to as the DQN Agent (Deep Q Network), used reinforcement learning and a deep neural network to learn the value function for each game. The Atari 2600 games are controlled by a joystick and a single button, used either individually or together. Depending on the game, the action space varied from 4 to 18 discrete actions. Unlike previous board game programs, where a numeric vector represented the state of the game, the DQN agent used only the pixels presented on the screen. Specifically, the program scaled the video from a 210 x 160 image with a 128-color palette to a monochrome image with varying luminance on a scale of 84 x 84 pixels. The program used a deep convolutional neural network to process these images and estimate the optimal action-value function. The $84 \times 84 \times 4$ image is processed through 3 convolutional layers and two fully connected layers. The output is the selected action, a combination joystick and button selections. The number of valid actions varied between 4 and 18 depending on the game. Using the same architecture and hyperparameter settings, the program was able to play at the level of a professional human tester across 49 games. In 2020 another team a Deep Mind developed an AI called MuZero, a reinforcement learning algorithm that combines model-based planning with learned environment models (Schrittwieser et al. 2020). MuZero achieved superhuman performance at Atari video games, but also in the traditional games of Go, chess, and shogi.

The Atari games are relatively simple by modern video game standards, but within a few years AI had been developed to play much more sophisticated games. StarCraft II is a real time strategy (RTS) game released in 2010. Players control one of three different factions with the objective of defeating the other factions by gathering resources, building bases and commanding armies. It is known as one of the most complex and skill intensive RTS games ever developed ('StarCraft II' 2025). While many AI programs were developed to compete in Starcraft II, the program AlphaStar, developed by a team at Deep Mind, was the first to achieve a Grandmaster rating. The program was ranked at the 99.8th percentile of officially ranked human players (Vinyals et al. 2019). Vinyals discussed the effort on (Fridman and Vinyals 2019). Using an overall process similar to AlphaGo, AlphaStar was trained using a combination of supervised learning, training to mimic human play, and reinforcement learning through self-play. Unlike the Atari program, AlphaStar did not use an image as input, instead it accessed unit level information via an API and implemented its actions through the same API. This API provided limited visibility information to the AI, the same information a human player could observe. While the difference in perception was criticized by some, the challenge for AlphaStar was not just to compete against humans, but to demonstrate an AI could perform the decision making required to succeed in a complex real-time game. To make those decisions, AlphaStar used a recurrent neural network as the core of its policy network, along with a separate value network—an overall configuration not dissimilar to the high-level architecture of AlphaGo.

Dota 2 is a team-based multi-player online battle game released in 2013. The game pits two 5-player teams against each other as they attempt to destroy the opponent's base. It is considered a game with a steep learning curve and high skill ceiling ('Dota 2' 2025). Dota 2 is popular as an esport with a large prize pool tournament, in excess of $35 million. In 2019 an AI named OpenAI Five, developed at OpenAI, defeated the world champion team in a direct competition (Berner, Brockman, and et.al. 2019). Open Five attacked this problem by applying reinforcement learning at a massive scale. Open AI deployed thousands of GPUs in training cycle that lasted 10 months. To support this massive training process, they developed a process they dubbed surgery, that allowed them to modify their model without having to restart training.

Like AlphaStar, OpenAI Five perceived the environment not by examining the screen image, but rather through an API that exposed about 16 thousand inputs which were designed to mimic information a human could see. The agent receives an update 7.5 times per second, returning a discrete action to the game engine. While faster than a typical human reaction rate, steps were taken to level make the agent operate at a human level, for example a 200ms observation delay.

OpenAI Five used Proximal Policy Optimization, an on policy, actor-critic based RL algorithm that had been previously developed by Open AI. The agent used a deep neural network as the actor, learning a policy that outputs a probability distribution over allowable actions for a given state, and a separate deep neural net as a critic, learning a value function from the current state. In OpenAI Five both these neural netes were recurrent neural nets, specifically long short-term memory (LSTM).

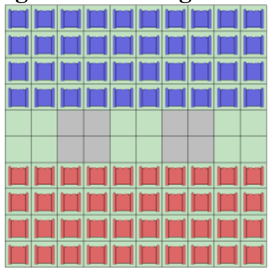
Gran Turismo is a highly realistic race car simulation and is the highest selling video game on the Sony PlayStation platform ('Gran Turismo (series)' 2025). In 2022 a team at Sony AI developed Quantile Regression Soft Actor-Critic, a custom deep RL algorithm that powered their AI GT Sophy, a model that competed against and ultimately defeated four of the world's best Gran Turismo divers in the world (Wurman et al. 2022) .

11 STRATEGO, DIPLOMACY, AND STUDENT OF GAMES

Board games like chess, Go, and shogi are difficult because of the large state spaces that need to be evaluated. But they are all games of perfect information. When imperfect information is introduced, even simple games generate much larger state spaces that must be evaluated in a game theoretic context. Stratego is a strategy game played on a 10x10 square board. Each player has 40 pieces representing different military ranks. The pieces come in 12 different designations including ranks, bombs, and a flag. Players take turns moving and higher ranked pieces can capture lower ranked pieces. The objective of the game is to capture the opponent’s flag ('Stratego' 2025).

The game is relatively simple to play and is designated for ages 8 and up. Each player deploys their pieces as shown in Figure 13 but with the ranks hidden from the opponent. This hidden information creates an enormous state space of 10^{565}.

Figure 13-Stratego Board



A team at Deep Mind developed an AI named Deep Nash that was able to learn to play Stratego via self-play without any human demonstration (Perolat et al. 2022). Deep Nash uses a game theoretical approach implemented with deep learning to learn the game end to end, from the initial deployment phase through the game playing phase using an algorithm they called Regularized Nash Dynamics (R-NaD). The algorithm uses a deep convolutional network, then processes the recommended moves through a fine-tuning module and a post processing module that filters out bad moves based on game specific knowledge. Deep Nash competed against eight other state of the art Stratego AIs in large scale repeated play winning over 97% of the games. It’s also competed against expert human level players achieving a number three ranking for 2022.

Diplomacy is another strategy game where two to seven players compete to control the most territory on a map of Europe. A unique and challenging aspect of Diplomacy is that players use bi-lateral communication to develop alliances. Effective communication and negotiation skills are required to successfully compete. The objective of the game is to control of majority of the territory on the map ('Diplomacy (game)' 2025).

A team at Meta developed the AI Cicero to compete in seven player Diplomacy (Meta et al. 2022). Cicero utilizes a pre-trained large language model (LLM) fine-tuned on a corpus of 12.9 million messages from archived games of Diplomacy. The output of the LLM is processed through a filtering system to eliminate non-sensical or inconsistent messages. The AI also uses deep reinforcement learning to implement a strategic reasoning module to select intents and actions. The module uses a policy and value network trained using self-play RL to maintain a human-compatible policy. Cicero was tested in 40 anonymous online Diplomacy games in 2022 against 82 distinct human players, none of whom knew they were playing an AI. Cicero achieved a score double the average human performing at a strong human level in a game that requires direct player to player communication, demonstrating for the first time that Large Language Models can be integrated as a component of a game playing AI.

The quest to build increasingly generalized game playing AIs, continued with the development of Student of Games by a cross institutional team of industry and academic researchers (Schmid et al. 2023). Student of Games can play both perfect information games (chess and Go) and imperfect information games (heads-up no-limit Texas hold’em and Scotland Yard). Scotland Yard is a board game where players control detectives trying to catch a criminal on the streets of London ('Scotland Yard (board game)' 2025). The AI combines concepts from AlphaZero, such as deep neural networks, self-play, and MCTS, and Deep Stack, game theoretic reasoning for imperfect information games. It implements rowing-Tree Counterfactual Regret Minimization (GT-CFR) for search and learns Counterfactual Value-and-Policy Networks (CVPNs) from self-play data. Student of Games achieves strong, expert-level performance in all tested games — chess, Go, poker, and Scotland Yard — but generally falls short of the very best domain-specialized agents when given the same compute budget. Student of Games sacrifices some efficiency to achieve greater generality.

12 SUMMARY AND CONCLUSIONS

Games have played a major role in the development of AI since the field was first conceived. Many of the most consequential innovations in AI were first developed in the game playing arena, and prior to the widespread adoption of generative AI models, game playing AIs where the face of AI.

Over the nearly sixty-year history of game playing AI, several key themes are apparent. The complexity of solvable games has expanded steadily and dramatically as AIs took on games with larger and larger search spaces, progressed from complete to incomplete information games, and advanced from one-on-one to multi-player games. The ability of AIs has increased from human to superhuman. A fundamental innovation is the progression from AIs where human knowledge is hard coded, as in Deep Blue, to imitated, as in Alpha Go, to learned *tabula rasa* via reinforcement learning, as in Alpha Zero. While early game AIs such as Deep Blue were narrowly focused on a single game, AIs have become vastly mode general with the same model able to excel at multiple games establishing the type of general reasoning capability that will enable narrow AI to generalize toward AGI. Modern game AIs are dominated by reinforcement learning, and the field of games was one of the primary domains for the development of RL. The technology behind game playing AIs has evolved to include deep learning, natural language processing, and large language models. One factor that has remained consistent is that most state-of-the-art game AIs required a massive level of compute, relative to the time.

The capabilities developed in game playing AIs have and will continue to be generalizable to non-game domains that are applicable to a range of business applications. The strategic decision making under uncertainty developed in AIs like AlphaZero has broad applicability in fields such as supply chain management. The game theoretic decision-making innovations in AIs that conquered poker can be applied to algorithmic trading. The capabilities demonstrated negotiating alliances in Diplomacy can be applied to large scale vendor negotiations in e-commerce. The innovations made in tree search for large solution space games can be applied to large scale optimization problems such as route optimization and scheduling. The ability to master video games via self-play can be applied to operational control of any system that can be modeled via stochastic simulation.

The business literature is being populated with applications of AI to supply chain (Brintrup et al. 2024; Burtea and Tsay 2024; Kassa et al. 2023; Rolf et al. 2023; Toorajipour et al. 2021; Yan et al. 2022; Zamani et al. 2023), operations (Gijsbrechts et al. 2022; Harsha et al. 2025; Kaynov et al. 2024; Küfner, Uhlemann, and Ziegler 2018), vehicle routing (Jiao et al. 2021), pricing (Deng et al. 2022; Mandania and Oliveira 2023; Zhu et al. 2024), healthcare operations (Wu et al. 2025) and general optimization (Balaji et al. 2019; Mazyavkina et al. 2021).

Academics researching quantitative applications in business must develop a foundational level of knowledge in artificial intelligence to remain relevant. Understanding the historical evolution of AI in domain of competitive games can play an important role in that process. This paper seeks to provide a road map to business researchers on an important stream of research that has largely been published outside of traditional business journals.

13 REFERENCES

'Atari, Inc.'. 2025. Accessed August 15, 2025. https://en.wikipedia.org/wiki/Atari,_Inc.

'Backgammon'. 2025. Accessed August 15, 2025. https://en.wikipedia.org/wiki/Backgammon.

Baker, Stephen. 2011. *Final Jeopardy: The Story of Watson, the Computer That Will Transform Our World* (Mariner Books).

Balaji, Bharathan, Jordan Bell-Masterson, Enes Bilgin, et al. 2019. 'ORL: Reinforcement Learning Benchmarks for Online Stochastic Optimization Problems'. https://arxiv.org/abs/1911.10641.

Berner, Christopher , Greg Brockman, and et.al. 2019. 'Dota 2 with Large Scale Deep Reinforcement Learning'. https://arxiv.org/abs/1912.06680.

Bernstein, A., T. Arbuckle, M. De V. Roberts, et al. 1958. "A chess playing program for the IBM 704." In *Proceedings of the May 6-8, 1958, western joint computer conference: contrasts in computers*, 157–59. Los Angeles, California: Association for Computing Machinery.

Bowling, Michael, Neil Burch, Michael Johanson, et al. 2015. 'Heads-up limit hold'em poker is solved', *Science*, 347: 145-49.

———. 2017. 'Heads-up limit hold'em poker is solved', *Commun. ACM*, 60: 81–88.

Brintrup, Alexandra, Edward Kosasih, Philipp Schaffer, et al. 2024. 'Digital supply chain surveillance using artificial intelligence: definitions, opportunities and risks', *International Journal of Production Research*, 62: 4674-95.

Brown, Noam, and Tuomas Sandholm. 2018. 'Superhuman AI for heads-up no-limit poker: Libratus beats top professionals', *Science*, 359: 418 - 24.

———. 2019. 'Superhuman AI for multiplayer poker', *Science*, 365: 885 - 90.

Burtea, Radu, and Calvin Tsay. 2024. 'Constrained continuous-action reinforcement learning for supply chain inventory management', *Computers & Chemical Engineering*, 181: 108518.

Campbell, Murray. 1999. 'Knowledge discovery in deep blue', *Commun. ACM*, 42: 65–67.

Campbell, Murray, A. Joseph Hoane, and Feng-hsiung Hsu. 2002. 'Deep Blue', *Artificial Intelligence*, 134: 57-83.

'Checkers'. 2025. Accessed August 15, 2025. https://en.wikipedia.org/wiki/Checkers.

Chollet, Francois. 2021. *Deep Learning with Python* (Manning Publications Co.).

Deng, Yipu, Jinyang Zheng, Liqiang Huang, et al. 2022. 'Let Artificial Intelligence Be Your Shelf Watchdog: The Impact of Intelligent Image Processing-Powered Shelf Monitoring on Product Sales', *MIS Q.*, 47: 1045-72.

'Diplomacy (game)'. 2025. Accessed August 15, 2025. https://en.wikipedia.org/wiki/Diplomacy_(game).

'Dota 2'. 2025. Accessed August 15, 2025. https://en.wikipedia.org/wiki/Dota_2.

Ferrucci, David A., Eric W. Brown, Jennifer Chu-Carroll, et al. 2010. 'Building Watson: An Overview of the DeepQA Project', *AI Mag.*, 31: 59-79.

Fridman, Lex, and Noam Brown. 2022. 'Noam Brown: AI for Poker, Diplomacy, and Multi-Agent Reasoning'. https://lexfridman.com/noam-brown/.

Fridman, Lex, and David Ferrucci. 2019. 'David Ferrucci: IBM Watson, Jeopardy & Deep Conversations with AI'. https://lexfridman.com/david-ferrucci/.

Fridman, Lex, and Garry Kasparov. 2019. 'Garry Kasparov: Chess, Deep Blue, AI, and Putin'. https://www.youtube.com/watch?v=8RVa0THWUWw.

Fridman, Lex, and Tuomas Sandholm. 2018. 'Tuomas Sandholm: Poker and Game Theory'. https://lexfridman.com/tuomas-sandholm/.

Fridman, Lex, and David Silver. 2020. 'David Silver: AlphaGo, AlphaZero, AlphaStar, and Deep Reinforcement Learning'. https://lexfridman.com/david-silver/.

Fridman, Lex, and Oriol Vinyals. 2019. 'Oriol Vinyals: AlphaStar and StarCraft II AI'. https://lexfridman.com/oriol-vinyals/.

Fudenberg, Drew, and Jean Tirole. 1991. *Game Theory* (MIT Press).

'Game complexity'. 2025. Accessed August 15, 2025. https://en.wikipedia.org/wiki/Game_complexity.

Gelly, Sylvain, and David Silver. 2011. 'Monte-Carlo tree search and rapid action value estimation in computer Go', *Artificial Intelligence*, 175: 1856-75.

Gijsbrechts, Joren, Robert N. Boute, Jan A. Van Mieghem, et al. 2022. 'Can Deep Reinforcement Learning Improve Inventory Management? Performance on Lost Sales, Dual-Sourcing, and Multi-Echelon Problems', *Manufacturing & Service Operations Management*, 24: 1349-68.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning* (MIT Press).

'Gran Turismo (series)'. 2025. Accessed August 15, 2025. https://en.wikipedia.org/wiki/Gran_Turismo_(series).

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey E. Hinton. 2013. 'Speech recognition with deep recurrent neural networks', *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*: 6645-49.

Harsha, Pavithra, Ashish Jagmohan, Jayant Kalagnanam, et al. 2025. 'Deep Policy Iteration with Integer Programming for Inventory Management', *Manufacturing & Service Operations Management*, 27: 369-88.

He, Kaiming, X. Zhang, Shaoqing Ren, et al. 2015. 'Deep Residual Learning for Image Recognition', *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*: 770-78.

Hinton, G. E., and R. R. Salakhutdinov. 2006. 'Reducing the Dimensionality of Data with Neural Networks', *Science*, 313: 504-07.

Hinton, Geoffrey E., Simon Osindero, and Yee Whye Teh. 2006. 'A Fast Learning Algorithm for Deep Belief Nets', *Neural Computation*, 18: 1527-54.

Hornik, Kurt, Maxwell B. Stinchcombe, and Halbert L. White. 1989. 'Multilayer feedforward networks are universal approximators', *Neural Networks*, 2: 359-66.

Hsu, Feng-Hsiung. 1999. 'IBM's Deep Blue Chess grandmaster chips', *IEEE Micro*, 19: 70-81.

———. 2002. *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion* (Princeton University Press).

'Jeopardy!'. 2025. Accessed August 15, 2025. https://en.wikipedia.org/wiki/Jeopardy!

Jiao, Yan, Xiaocheng Tang, Zhiwei Qin, et al. 2021. 'Real-world ride-hailing vehicle repositioning using deep reinforcement learning', *Transportation Research Part C: Emerging Technologies*, 130: 103289.

Kassa, Adane, Daniel Kitaw, Ulrich Stache, et al. 2023. 'Artificial intelligence techniques for enhancing supply chain resilience: A systematic literature review, holistic framework, and future research', *Computers & Industrial Engineering*, 186: 109714.

Kaynov, Illya, Marijn van Knippenberg, Vlado Menkovski, et al. 2024. 'Deep Reinforcement Learning for One-Warehouse Multi-Retailer inventory management', *International Journal of Production Economics*, 267: 109088.

Keim, Greg A., Noam M. Shazeer, Michael L. Littman, et al. 1999. "PROVERB: The Probabilistic Cruciverbalist." In *AAAI/IAAI*.

Kister, J., P. Stein, S. Ulam, et al. 1957. 'Experiments in Chess', *J. ACM*, 4: 174–77.

Kohs, Greg. 2017. 'AlphaGo - The Movie '. https://www.youtube.com/watch?v=WXuK6gekU1Y.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. 2012. 'ImageNet classification with deep convolutional neural networks', *Communications of the ACM*, 60: 84 - 90.

Küfner, Thomas, Thomas H. J. Uhlemann, and Bastian Ziegler. 2018. 'Lean Data in Manufacturing Systems: Using Artificial Intelligence for Decentralized Data Reduction and Information Extraction', *Procedia CIRP*, 72: 219-24.

Littman, Michael L., Greg A. Keim, and Noam Shazeer. 2002. 'A probabilistic approach to solving crossword puzzles', *Artificial Intelligence*, 134: 23-55.

Mandania, Rupal, and Fernando S. Oliveira. 2023. 'Dynamic pricing of regulated field services using reinforcement learning', *IISE Transactions*, 55: 1022-34.

Marsland, T. Anthony, and J. Schaeffer. 1990. *Computers, Chess, and Cognition* (Springer-Verlag).

Mazyavkina, Nina, Sergey Sviridov, Sergei Ivanov, et al. 2021. 'Reinforcement learning for combinatorial optimization: A survey', *Computers & Operations Research*, 134: 105400.

McCulloch, Warren S., and Walter Pitts. 1943. 'A logical calculus of the ideas immanent in nervous activity', *The bulletin of mathematical biophysics*, 5: 115-33.

Meta, Anton Bakhtin, Noam Brown, et al. 2022. 'Human-level play in the game of Diplomacy by combining language models with strategic reasoning', *Science*, 378: 1067-74.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, et al. 2015. 'Human-level control through deep reinforcement learning', *Nature*, 518: 529-33.

Moravčík, Matej, Martin Schmid, Neil Burch, et al. 2017. 'DeepStack: Expert-level artificial intelligence in heads-up no-limit poker', *Science*, 356: 508-13.

Nash, John F. 1950. 'Equilibrium Points in N-Person Games', *Proceedings of the National Academy of Sciences of the United States of America*, 36 1: 48-9.

Newell, Allen, J. C. Shaw, and H. A. Simon. 1958. 'Chess-playing programs and the problem of complexity', *IBM J. Res. Dev.*, 2: 320–35.

Perolat, Julien, Bart De Vylder, Daniel Hennes, et al. 2022. 'Mastering the game of Stratego with model-free multiagent reinforcement learning', *Science*, 378: 990-96.

'Reversi'. 2025. Accessed August 15, 2025. https://en.wikipedia.org/wiki/Reversi.

Rolf, Benjamin, Ilya Jackson, Marcel Müller, et al. 2023. 'A review on reinforcement learning algorithms and applications in supply chain management', *International Journal of Production Research*, 61: 7151-79.

Rosenblatt, Frank. 1958. 'The perceptron: a probabilistic model for information storage and organization in the brain', *Psychological review*, 65 6: 386-408.

Rosenbloom, Paul S. 1982. 'A world-championship-level Othello program', *Artificial Intelligence*, 19: 279-320.

'Rules of Chess'. 2025. Accessed August 15, 2025. https://en.wikipedia.org/wiki/Rules_of_chess.

'Rules of Go'. 2025. Accessed August 15, 2025. https://en.wikipedia.org/wiki/Rules_of_Go.

Russell, Stuart J., and Peter Norvig. 2020. *Artificial intelligence : a modern approach* (Pearson: Boston).

Samuel, A. L. 1959. 'Some Studies in Machine Learning Using the Game of Checkers', *IBM Journal of Research and Development*, 3: 210-29.

———. 1967. 'Some Studies in Machine Learning Using the Game of Checkers. II—Recent Progress', *IBM Journal of Research and Development*, 11: 601-17.

Schmid, Martin, Matej Moravčík, Neil Burch, et al. 2023. 'Student of Games: A unified learning algorithm for both perfect and imperfect information games', *Science Advances*, 9.

Schrittwieser, Julian, Ioannis Antonoglou, Thomas Hubert, et al. 2020. 'Mastering Atari, Go, chess and shogi by planning with a learned model', *Nature*, 588: 604-09.

'Scotland Yard (board game)'. 2025. Accessed August 15, 2025. https://en.wikipedia.org/wiki/Scotland_Yard_(board_game).

'Scrabble'. 2025. Accessed August 15, 2025. https://en.wikipedia.org/wiki/Scrabble.

Shannon, Claude E. 1950. 'Programming a computer for playing chess', *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 41: 256-75.

Sheppard, Brian. 2002. 'World-championship-caliber Scrabble', *Artificial Intelligence*, 134: 241-75.

Silver, David, Aja Huang, Chris J. Maddison, et al. 2016. 'Mastering the game of Go with deep neural networks and tree search', *Nature*, 529: 484-89.

Silver, David, Thomas Hubert, Julian Schrittwieser, et al. 2017. 'Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm'. https://arxiv.org/abs/1712.01815.

Silver, David, Julian Schrittwieser, Karen Simonyan, et al. 2017. 'Mastering the game of Go without human knowledge', *Nature*, 550: 354-59.

'StarCraft II'. 2025. Accessed August 15, 2025. https://en.wikipedia.org/wiki/StarCraft_II.

Stevens, E., L. Antiga, and T. Viehmann. 2020. *Deep Learning with PyTorch: Build, train, and tune neural networks using Python tools* (Manning).

'Stratego'. 2025. Accessed August 15, 2025. https://en.wikipedia.org/wiki/Stratego.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. 2014. 'Sequence to Sequence Learning with Neural Networks'. https://arxiv.org/abs/1409.3215.

Sutton, Richard S. 1988. 'Learning to predict by the methods of temporal differences', *Machine Learning*, 3: 9-44.

Sutton, Richard S., and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (A Bradford Book).

Tesauro, G., D. C. Gondek, J. Lenchner, et al. 2012. 'Simulation, learning, and optimization techniques in Watson's game strategies', *IBM Journal of Research and Development*, 56: 16:1-16:11.

Tesauro, Gerald. 1992. 'Practical issues in temporal difference learning', *Machine Learning*, 8: 257-77.

———. 1994. 'TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play', *Neural Computation*, 6: 215-19.

———. 1995. 'Temporal difference learning and TD-Gammon', *Commun. ACM*, 38: 58-68.

———. 2002. 'Programming backgammon using self-teaching neural nets', *Artificial Intelligence*, 134: 181-99.

Tesauro, Gerald, David Gondek, Jonathan Lenchner, et al. 2013. 'Analysis of Watson's Strategies for Playing Jeopardy!', *J. Artif. Intell. Res.*, 47: 205-51.

'Texas hold 'em'. 2025. Accessed August 15, 2025. https://en.wikipedia.org/wiki/Texas_hold_%27em.

Toorajipour, Reza, Vahid Sohrabpour, Ali Nazarpour, et al. 2021. 'Artificial intelligence in supply chain management: A systematic literature review', *Journal of Business Research*, 122: 502-17.

Turing, A. M. 1950. 'Computing machinery and intelligence', *Mind*, 59: 433-60.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, et al. 2017. "Attention is all you need." In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6000–10. Long Beach, California, USA: Curran Associates Inc.

Vinyals, Oriol, Igor Babuschkin, Wojciech M. Czarnecki, et al. 2019. 'Grandmaster level in StarCraft II using multi-agent reinforcement learning', *Nature*, 575: 350 - 54.

Vinyals, Oriol, Alexander Toshev, Samy Bengio, et al. 2014. 'Show and tell: A neural image caption generator', *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*: 3156-64.

Von Neumann, John, and Oskar Morgenstern. 1947. *Theory of games and economic behavior, 2nd rev. ed* (Princeton University Press: Princeton, NJ, US).

Wu, Qihao, Jiangxue Han, Yimo Yan, et al. 2025. 'Reinforcement learning for healthcare operations management: methodological framework, recent developments, and future research directions', *Health Care Management Science*, 28: 298-333.

Wurman, Peter R., Samuel Barrett, Kenta Kawamoto, et al. 2022. 'Outracing champion Gran Turismo drivers with deep reinforcement learning', *Nature*, 602: 223-28.

Yan, Yimo, Andy H. F. Chow, Chin Pang Ho, et al. 2022. 'Reinforcement learning for logistics and supply chain management: Methodologies, state of the art, and future opportunities', *Transportation Research Part E: Logistics and Transportation Review*, 162: 102712.

Zamani, Efpraxia D., Conn Smyth, Samrat Gupta, et al. 2023. 'Artificial intelligence and big data analytics for supply chain resilience: a systematic literature review', *Annals of Operations Research*, 327: 605-32.

Zhu, Xinghui, Lulu Jian, Xin Chen, et al. 2024. 'Reinforcement learning for Multi-Flight Dynamic Pricing', *Computers & Industrial Engineering*, 193: 110302.

Zinkevich, Martin A., Michael Bradley Johanson, Michael Bowling, et al. 2007. "Regret Minimization in Games with Incomplete Information." In *Neural Information Processing Systems*.