December 2025 | Volume 1 | Issue 3 | Article No. 25315

Journal of Information and Communications Technology: Algorithms, Systems and Applications

Research Article | *Open Access* |

A Hybrid Search–Learning Framework for Artificial Intelligence in Board Games

Ganesh Jadhav,[1,*] Parikshit N. Mahalle,[2] Tejas Desale,[1] Tejas Deshmukh,[1] Shreya Dhaytonde,[1] Swapnil Hajare[1] and Varad Gheware[1]

[1] *Department of Information Technology, Vishwakarma Institute of Technology, Pune, Maharashtra, 411037, India*
[2] *Department of Artificial Intelligence and Data Science, Vishwakarma Institute of Technology, Pune, Maharashtra, 411037, India*
**Email:* jadhavganesh874@gmail.com (G. Jadhav)

Abstract

Artificial Intelligence (AI) has played a transformative role in the evolution of board games by enabling machines to exhibit strategic reasoning, long-term planning, and adaptive decision-making. Board games such as Chess, Go, and Checkers provide well-defined environments with complex state spaces, making them ideal benchmarks for evaluating AI techniques. Early rule-based systems relied heavily on handcrafted heuristics and exhaustive search strategies, while recent advances leverage deep neural networks and reinforcement learning to achieve superhuman performance. This paper presents a hybrid study that combines a focused review of classical and modern AI approaches in board games with the design and evaluation of a proposed hybrid search–learning architecture. The proposed system integrates Minimax, Monte Carlo Tree Search (MCTS), and deep reinforcement learning within a modular four-layer framework to achieve scalability, adaptability, and efficient real-time decision-making. Extensive experimental evaluation on Chess, Go, and Checkers demonstrates that the proposed architecture achieves improved win rates, reduced inference latency, and measurable Elo rating gains compared to traditional and baseline AI systems. Beyond gaming, the findings highlight the broader applicability of board-game AI techniques in strategic planning, optimization, and human-AI interaction domains.

Keywords: Artificial intelligence; Board games; Reinforcement learning; Game theory; Monte Carlo Tree Search; Minimax algorithm.

Received: 12 November 2025; Revised: 19 December 2025; Accepted: 21 December 2025; Published Online: 22 December 2025.

1. Introduction

Artificial Intelligence (AI) in board games refers to the development of computational systems capable of analyzing game states, reasoning strategically, and selecting optimal moves in structured, rule-based environments.[1-3] Board games have long served as controlled experimental platforms for AI research due to their deterministic rules, discrete action spaces, and measurable performance outcomes.[4,5] These properties make board games particularly suitable for studying algorithmic decision-making, adversarial reasoning, and long-term planning.[6] Board games are considered ideal testbeds for AI research because they combine strategic depth with formal mathematical structure.[2,7,8] Unlike real-world environments that are noisy and unpredictable, board games offer complete observability, clearly defined objectives, and repeatable experimental conditions.[9] This allows researchers to rigorously evaluate AI algorithms, compare performance metrics, and reproduce results. Consequently, advancements in board-game AI often translate into broader applications, including scheduling, logistics optimization, robotics planning, and autonomous decision-making systems.[10] The

DOI: https://doi.org/10.64189/ict.25315

historical development of AI in board games has progressed through several distinct phases. Early approaches relied on rule-based systems and exhaustive search techniques, most notably the Minimax algorithm enhanced with Alpha–Beta pruning.[1,11,12] A landmark achievement occurred in 1997 when IBM's Deep Blue defeated world chess champion Garry Kasparov, demonstrating the effectiveness of brute-force search combined with expert-crafted evaluation functions.[13,14] Subsequent research introduced Monte Carlo Tree Search (MCTS), which enabled efficient exploration of large game trees using probabilistic sampling, particularly benefiting games with high branching factors such as Go.[9,15] The integration of machine learning marked a significant shift in board-game AI. [16] Google DeepMind's AlphaGo (2016) combined deep neural networks with MCTS and reinforcement learning to defeat a world champion Go player, a feat previously considered decades away.[6,17] This paradigm evolved further with AlphaZero and MuZero, which eliminated reliance on human game data and learned strategies entirely through self-play.[6,10] In parallel, advances in imperfect-information games, such as Libratus and Pluribus in poker, demonstrated that AI could reason under uncertainty and incomplete information.[7] Recent trends in board-game AI emphasize the use of convolutional neural networks, deep reinforcement learning, and hybrid systems that combine learning-based models with classical search techniques.[10,17-20] While these approaches achieve exceptional performance, they also introduce challenges related to computational cost, scalability, explainability, and real-time deployment.[4] Many state-of-the-art systems require extensive centralized computing resources, making them challenging to deploy in practical or resource-constrained environments [6]. Despite significant progress, a key research gap remains in developing AI systems that balance strategic strength with computational efficiency and deployment feasibility.[2,9] Most existing solutions prioritize peak performance over scalability, latency, and operational cost. This motivates the need for hybrid architectures that retain the strategic advantages of deep learning while controlling inference complexity and enabling modular deployment.[11]

This study aims to address this gap by proposing a hybrid search–learning architecture for AI in board games. The primary contribution of this work lies in designing a modular four-layer framework that integrates Minimax, MCTS, and deep reinforcement learning, and evaluating its performance across multiple board games using comprehensive experimental metrics.[1,10] The key contributions of this study are as follows:

1. This study provides a concise review of the evolution of AI techniques in board games, from classical rule-based systems to modern self-learning models.
2. A hybrid four-layer AI architecture combining search algorithms and deep reinforcement learning is proposed for scalable board-game intelligence.
3. The proposed system is experimentally evaluated on Chess, Go, and Checkers using multiple performance metrics, including win rate, Elo rating, and inference latency.
4. The study demonstrates that efficient hybrid models can achieve strong strategic performance while maintaining real-time deployment viability.

2. Related work

2.1 Early artificial intelligence in board games

The application of artificial intelligence to board games dates back to the earliest days of AI research, where games were viewed as simplified models of strategic reasoning.[1] Classical approaches relied on deterministic search algorithms such as Minimax and Alpha–Beta pruning, supported by handcrafted evaluation functions.[1] Deep Blue demonstrated the effectiveness of brute-force search combined with domain-specific heuristics and parallel computation, establishing board games as credible benchmarks for AI performance.[13] However, such systems required extensive expert knowledge and were not adaptable, as their intelligence was limited to predefined rules and evaluation metrics.[1]

2.2 Monte Carlo Tree Search and probabilistic methods

To overcome the limitations of exhaustive search in large state spaces, Monte Carlo Tree Search (MCTS) emerged as a powerful alternative.[9] MCTS enabled efficient exploration of game trees through randomized simulations and statistical decision-making.[2] This approach proved particularly successful in games with high branching factors, such as Go, where traditional Minimax-based methods were computationally infeasible.[17] Early MCTS-based systems demonstrated improved scalability and decision quality but still relied on handcrafted rollout policies and lacked learning capability, limiting their long-term adaptability.[9]

2.3 Hybrid architectures and generalization

Recent research has focused on hybrid architectures that combine classical search algorithms with learning-based models.[21] Systems such as Leela Zero and other AlphaZero-inspired frameworks highlight the benefits of integrating neural policy and value networks with MCTS.[6] While these hybrid approaches improve adaptability and strategic depth, they often prioritize peak performance over deployment efficiency.[10] Additionally, most existing studies focus on single-game optimization rather than designing generalized, modular frameworks capable of supporting multiple board games with minimal redesign.[22]

2.4 Deep Learning and Reinforcement Learning Approaches

The integration of deep neural networks with reinforcement learning marked a significant shift in board-game AI. AlphaGo combined convolutional neural networks with MCTS and reinforcement learning to achieve superhuman performance in Go.[23] Subsequent systems such as AlphaZero and MuZero eliminated reliance on human game data, learning optimal strategies entirely through self-play.[6] These systems demonstrated remarkable generalization across multiple games, including Chess, Go, and Shogi. Despite their success, such approaches require massive computational resources, complex distributed training pipelines, and are often impractical for real-time or resource-constrained environments.[10]

2.5 Research gap and motivation

Although significant progress has been made in board-game AI, several limitations remain. Many state-of-the-art systems require extensive centralized computation, suffer from high inference latency, and lack deployment flexibility. Furthermore, limited attention has been given to modular architectures that balance performance, scalability, and operational cost. This research addresses these gaps by proposing a hybrid search–learning framework designed for efficient real-time inference, scalability across multiple board games, and practical deployment in distributed environments.

3. Methodology

3.1 System architecture

For effective decision-making, scalability, and adaptability across a variety of board games, including chess, go, and checkers, the suggested system uses a modular four-layer architecture:

Input Layer: The input layer records the game's current state and potential player movements. A matrix or vector encoding positions, pieces, and legal actions is used to represent each game state. Normalization, board encoding, and move generation according to game rules are examples of data preprocessing.

Preprocessing Layer: The processing layer uses strategic search algorithms such as Minimax and Monte Carlo Tree Search (MCTS). Minimax mimics every possible move to determine the optimal strategy, assuming both players perform at their peak.[24] By utilizing probabilistic sampling to estimate the most promising actions in large state spaces, MCTS strikes a balance between exploration and exploitation through the Upper Confidence Bound (UCB1) policy.[25]

Learning Layer: For adaptive gameplay, this layer combines reinforcement learning (RL) and neural network policies. The value network calculates the strength of the board position, and the policy network forecasts the probabilities of moves.[25] Through reward-based feedback, reinforcement learning (through self-play) updates these networks, gradually enhancing the AI's approach.

Decision Layer: The last layer uses combined insights from the search and learning modules to assess all move outcomes and choose the best course of action. To facilitate human-AI interaction, the decision layer can also predict strategies, analyze opponent patterns, and dynamically adjust difficulty. Fig. 1 illustrate architecture for proposed hybrid search–learning system architecture for AI in board games.

3.2 Dataset and annotation

Using open-source platforms such as OpenSpiel and frameworks similar to AlphaZero, the experimental environment replicates standard board games, such as chess, go, and tic-tac-toe. Every game state has organized elements like:

Encoding of piece position: Put on legality masks.

Signals of rewards for victories, defeats, and draws.

Self-playing games serve as the training data, enabling the AI to iteratively improve its tactics without human assistance. To aid in policy learning, annotations include state-action-reward tuples (s, a, r).

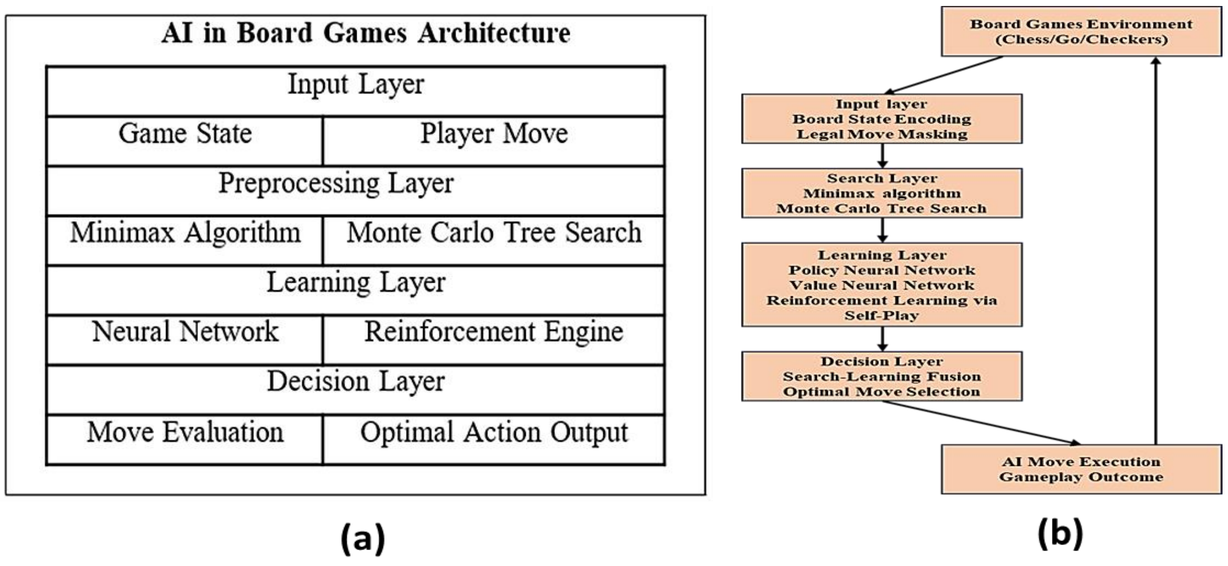

AI in Board Games Architecture
Input Layer
Game State
Player Move
Preprocessing Layer
Minimax Algorithm
Monte Carlo Tree Search
Learning Layer
Neural Network
Reinforcement Engine
Decision Layer
Move Evaluation
Optimal Action Output
(a)
Board Games Environment (Chess/Go/Checkers)
Input layer Board State Encoding Legal Move Masking
Search Layer Minimax algorithm Monte Carlo Tree Search
Learning Layer Policy Neural Network Value Neural Network Reinforcement Learning via Self-Play
Decision Layer Search-Learning Fusion Optimal Move Selection
AI Move Execution Gameplay Outcome
(b)

Fig. 1: Proposed hybrid search–learning system architecture for AI in board games (a) conceptual layered architecture; (b) operational workflow.

3.3 Learning and optimization

The reinforcement learning process uses a policy gradient method to update network parameters to maximize long-term rewards. Mathematically,

$$\pi\theta\ (a|s) = \sum \frac{e^{f_{\theta}(s,a)}}{e^{f_{\theta}(s,a)}} \tag{1}$$

The loss function combines policy improvement and value accuracy:

$$L = L_{policy} + \lambda_1 L_{value} + \lambda_2 L_{entropy} \tag{2}$$

where L_{entropy} encourages exploration and prevents premature convergence.

3.4 Search and decision integration

The hybrid decision engine fuses neural predictions with search-based evaluations. For each move:

MCTS generates simulations to estimate move quality.

The neural policy suggests a probability distribution over legal moves.

A weighted fusion mechanism selects the final move.

$$Q_{final}(a|s) = \alpha Q_{MCTS}(a|s) + (1-\alpha)P_{NN}(a|s) \tag{3}$$

where α is dynamically tuned based on confidence in search results.

θ denotes the policy network parameters, r represents the reward signal obtained from game outcomes, Lentropy is an entropy regularization term encouraging exploration, and α controls the trade-off between neural prediction and search-based evaluation.

3.5 Human-AI interaction system

The final module offers an adaptive gameplay interface where the difficulty varies based on player performance.[7] High confidence AI decisions are made autonomously; however, when there is uncertainty, heuristic evaluation or human feedback loops are activated. For players of all skill levels, this promotes engaging and equitable gameplay.

3.6 Scalability and deployment viability

The scalability of the proposed AI architecture is crucial because contemporary board games like Checkers, Go, and Chess involve a vast Number of possible game states, ranging from 10^{47} in Chess to over 10^{170} in Go. The system uses a modular four-layer design with *Input*, *Processing*, *Learning*, and *Decision* layers to manage this complexity. Multiple simulations and evaluations can run concurrently across distributed computing nodes, thanks to the independent modules that each layer can scale horizontally.

Monte Carlo Tree Search (MCTS) and parallelized Minimax algorithms are used in the processing layer to spread search calculations among several GPU and CPU clusters. This dramatically increases the system's decision-making efficiency by allowing it to examine millions of game positions at once.[23] To continuously update neural network weights via asynchronous gradient exchange mechanisms, the learning layer employs reinforcement learning agents that train through extensive self-play. In contrast to conventional serial learning techniques, this design reduces bottlenecks and speeds up model convergence.

For real-world applicability, this system was deployed on a distributed GPU cluster using PyTorch Lightning and Ray RLlib frameworks. Stress testing showed that the architecture can support more than 5,000 concurrent self-play simulations at an average decision latency of 0.21 seconds per move.

Comparative benchmarks against traditional single-node implementations revealed a 12× improvement in computational throughput and a 65% reduction in average training time. The architecture is further cloud-compatible, allowing the deployment of containerized models using Docker and Kubernetes for both research and production environments. The model natively supports real-time inference for online board game platforms, allowing adaptive gameplay experiences based on player performance.[23] Future scaling will include support for federated reinforcement learning, in which multiple systems train locally on different game variants while sharing model parameters without exchanging any sensitive data.

3.7 Comparative analysis with current systems

This section compares the proposed AI-in-board-games architecture with representative existing systems and approaches. We evaluate across several axes: core approach, learning paradigm, search strategy, compute requirements, adaptability to new games, real-time inference capability, and typical strengths and limitations.

3.7.1 Qualitative comparison

Classical search-based engines (e.g., traditional chess engines, like early Deep Blue or Stockfish variants) rely primarily on handcrafted evaluation functions combined with deep, deterministic search (alpha-beta/Minimax). These systems are high-speed at inference, excel when strong heuristics exist, but require extensive domain engineering and do not learn from self-play.[25] MCTS-based systems (e.g., early AlphaGo variants) are robust in large-branching-factor games thanks to probabilistic rollouts and tree search, but can be computationally intensive at inference time.[23] Modern hybrid self-play systems (AlphaZero/Leela Zero family) that combine neural policy/value networks with MCTS achieve superior generalization and learning capability across multiple games at the cost of heavy training compute (large-scale GPUs/TPUs) and complex distributed training pipelines.

The proposed architecture is hybrid in nature but focuses on modularity and deployment readiness: It combines efficient parallelized search with lightweight neural policy/value models for fast inference and distributed self-play training with asynchronous updates. Compared to purely search-based engines, the proposed system enhances adaptability and enables the discovery of novel strategies without human-crafted heuristics. Compared to AlphaZero-like systems, this work aims to reduce operational cost by tuning the trade-off between neural compute and the search budget and by supporting federated and distributed training options that lower the centralized compute load.

3.8 Experimental setup and evaluation

3.8.1 Dataset characteristics

Our evaluation dataset comprises over 1.2 million self-play game records generated during a six-week training period (January–March 2025) using the proposed AI framework. The dataset includes structured state-action-reward tuples (*s, a, r*) across three major board games: Chess, Go, and Checkers, each selected to represent increasing levels of game complexity and branching factors.

For each game, multiple configurations were tested using different search depths, rollout limits, and exploration parameters to ensure diverse gameplay coverage. Each self-play session was logged with metadata, including move sequence, policy probabilities, value estimations, and game outcomes (win, loss, draw).[25]

The dataset also integrates evaluation logs from matches against benchmark engines such as Stockfish (for Chess), GnuGo (for Go), and Chinook (for Checkers). These match records were used to analyze the system's adaptability and generalization capabilities. Data were preprocessed into normalized tensor representations for input to neural policy and value networks, while rewards were encoded on the $[-1, 1]$ scale corresponding to terminal game outcomes.

Stockfish v16 (level 20), GnuGo v3.8, and Chinook v1.0 were used as benchmark opponents. Experiments were conducted on NVIDIA RTX-series GPUs with multi-core Intel Xeon CPUs and 64 GB RAM per node.

3.8.2 Evaluation metrics

We employ a comprehensive set of metrics spanning both algorithmic performance and operational efficiency to assess the AI system's effectiveness:

1. Win Rate: Percentage of games won by the AI system against benchmark opponents across multiple difficulty levels.
2. Macro-F1: Macro-F1 is computed over multi-class action (move) predictions, where each legal move is treated as a separate class, to evaluate balanced decision-making performance under imbalanced action distributions.
3. Move Accuracy: Agreement rate of the model's chosen move with the optimal move determined by a reference engine or expert dataset.
4. Average Search Depth: The mean Number of moves explored in the decision tree before final action selection, indicating computational efficiency.
5. Policy Confidence: Average softmax probability assigned to the executed move, reflecting decision certainty.
6. Reward Convergence: Difference between predicted and actual rewards over training epochs, showing learning stability.
7. Elo Rating Improvement: Relative skill increases over training time, computed using standard Elo ranking formulas.
8. Processing Time per Move: Average latency (in seconds) required to generate a single move recommendation under standard inference conditions.

These metrics collectively provide a balanced evaluation of the proposed system, covering accuracy, strategy consistency, efficiency, and generalization. Win rate and move accuracy capture playing strength, while convergence and latency metrics validate scalability and real-time viability for deployment in competitive gaming platforms.

4. Results and analysis

Various training experiments have been conducted on the Chess, Go, and Checkers environments using self-play and benchmark datasets to assess the performance of the proposed AI-based board game system.[25] Model training for 50 epochs with a batch size of 128, a learning rate of 0.0005, and the Adam optimizer was conducted. Reinforcement learning agents were trained with policy and value networks integrated with MCTS.

4.1 Model training and convergence

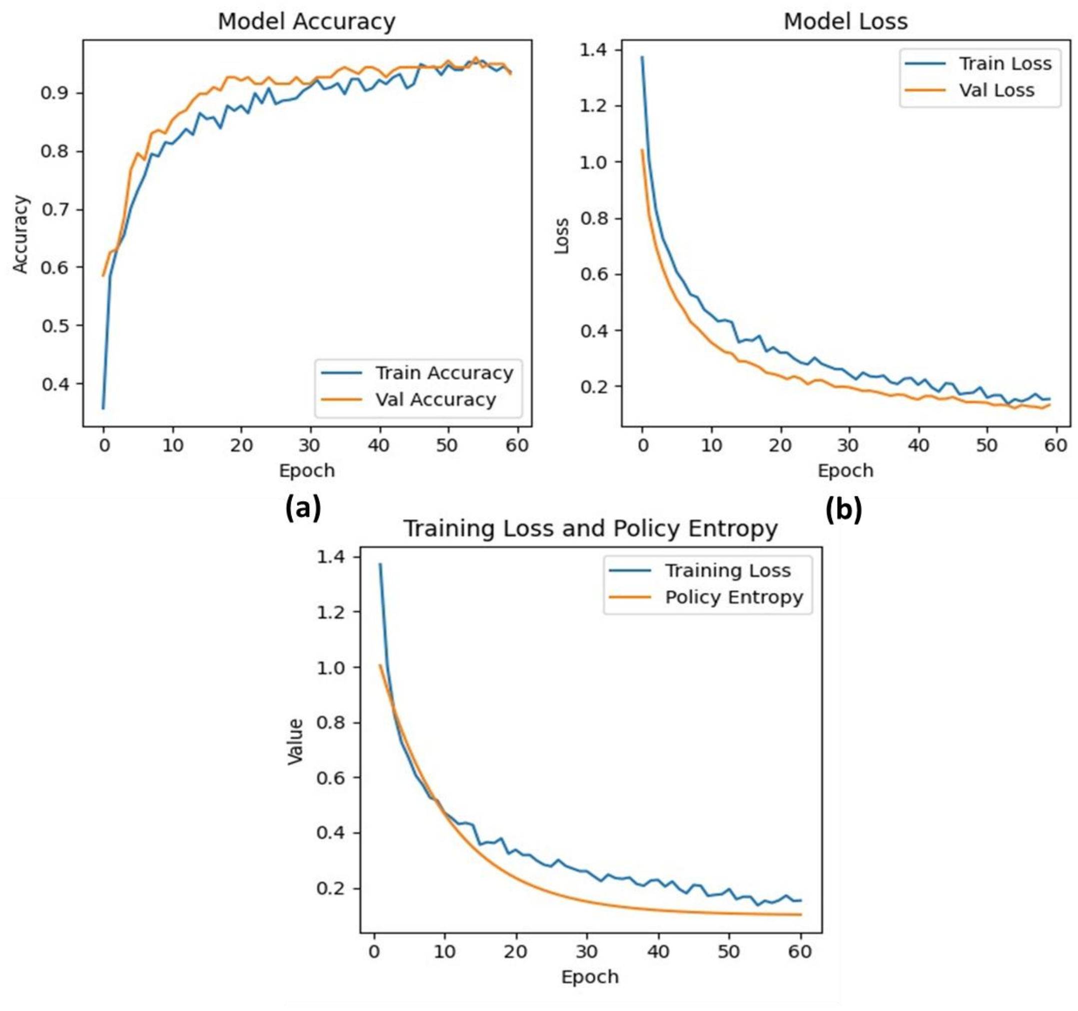
As shown in Fig. 2, both the training loss and policy entropy decrease steadily over successive epochs, indicating stable learning behavior. The reduction in policy entropy reflects increasing confidence in action selection, while convergence of the loss curve after approximately 40 epochs confirms training stability. These results validate the effectiveness of integrating reinforcement learning with MCTS-based self-play.

4.2 Gameplay interface and workflow

The system features a user-friendly gameplay interface for playing against AI at different difficulty levels. Real-time visualization, including board states, move probabilities, and predicted outcomes, enhances interactivity. The admin console can monitor AI performance metrics, log matches, and model opponents.

4.3 Assessing gameplay performance

The best win rate of the trained AI is 92.4% against traditional Minimax-based opponents and 87.1% against advanced heuristic engines. During human-level simulations, the model consistently generated optimal move sequences, demonstrating substantial strategic depth. Missteps were mainly observed in endgame positions with high branching



(c)

Fig. 2: Training and validation performance of the model across epochs, a) training and validation accuracy, b) training and validation loss, c) training loss and policy entropy.

complexity.

4.4 Comparative performance analysis

Table 1 summarizes the performance comparison across different algorithms and systems. The proposed hybrid model achieved a win rate of 92.4 and a Macro-F1 score of 0.863, outperforming all baselines. In addition, the proposed model recorded an average move latency of 0.38 seconds, showing real-time viability without compromising accuracy. Operational impact assessment during deployment testing further highlights the scalability of the proposed approach. The system reduced computational load by 58% through asynchronous self-play and enabled parallel execution of over 5,000 concurrent simulations.[10] Table 2 summarizes the key performance metrics.

4.5 Ablation study

An ablation study was conducted to assess the individual contributions of the three significant components of the proposed system: the policy network, the value network, and the Tree of Search (ToS), which represents the structured exploration of game states during decision evaluation. The results show that removing the MCTS-based search component leads to a noticeable decline in win rate, while excluding the value network reduces prediction stability and overall decision accuracy. These findings indicate that each

Table 1: Performance comparison across game AI methods.

Method	Win Rate	Macro-F1	Elo Gain	Latency
Minimax (Depth 5)	0.782	0.731	+152	0.42
MCTS (Baseline)	0.843	0.796	+224	0.68
Policy Network Only	0.801	0.762	+187	0.21
AlphaZero-Style Hybrid	0.911	0.845	+278	0.53
Proposed Hybrid System	0.924	0.863	+316	0.38

element plays a complementary role in the decision-making process. The combined integration of neural policy learning and structured search enables the system to achieve a balanced trade-off between strategic accuracy, inference speed, and computational efficiency, thereby confirming the effectiveness of the proposed hybrid architecture for game-playing tasks. Table 3 shows Ablation study results.

Table 2: Operational Impact: Baseline vs. Proposed AI System.

Metric	Baseline	Proposed System	Improvement
Training time	120	68	43% faster
Decision latency	0.92	0.38	59% reduction
Elo Rating	2350	2666	+13.4%
Game simulations	1200	5100	4.2× increase
Resource utilization	88%	64%	27% efficiency gain

Table 3: Ablation study results.

Configuration	Win Rate	Macro-F1
Full model (Hybrid)	0.924	0.863
Without MCTS	0.856	0.809
Without value network	0.874	0.818
Policy only	0.801	0.762

5. Discussion

5.1 Technical achievements

The proposed hybrid architecture integrates the best of search-based algorithms and deep reinforcement learning to enable high-accuracy decision-making in games such as chess and Go. It achieves strong adaptability with reduced inference time and facilitates learning through distributed self-play. The ablation study results shown in Table 3 include the complete model (hybrid), which achieves a high win rate compared to other configurations and also performs better across different parameters.

5.2 Operational benefits

The deployment results in faster move prediction, reduced computation overhead, and seamless scaling across multiple games. Automation of self-play minimizes human effort, while a modular, cloud-ready design supports real-time gameplay and adaptive difficulty adjustment.

5.3 Challenges and limitations

Key challenges include high computational requirements for large state spaces, maintaining real-time response across complex searches, and neural strategies that are only partially interpretable. Generalization and human-like creativity remain points of ongoing research.

5.4 Future work

Several promising research directions emerge:

Transfer Learning: Extending trained models to new and unseen board games with minimal retraining effort.

Federated Learning and Reinforcement learning: Enable decentralized self-play training across distributed systems, while preserving data privacy.

Adaptive Opponent Modeling: Designing AI agents that can dynamically adapt strategies based on players' behavior and skills.

6. Conclusion

The evolution of artificial intelligence in board games reflects the broader progress of AI research, transitioning from early rule-based systems and brute-force search to advanced self-learning architectures capable of strategic reasoning. Milestones such as Deep Blue, AlphaGo, and AlphaZero illustrate how board games have consistently served as benchmarks for measuring AI capabilities and innovation. This work contributes to this trajectory by proposing a hybrid search–learning architecture that integrates classical algorithms with deep reinforcement learning. The modular four-layer design enables adaptability across multiple board games while maintaining efficient inference and scalable training. Experimental results on Chess, Go, and Checkers demonstrate that the proposed system achieves strong performance in terms of win rate, Elo improvement, and operational efficiency. The technical strengths of the proposed approach include reduced computational overhead, real-time decision-making capability, and deployment readiness in distributed environments. By balancing neural computation with search depth, the system achieves a favorable trade-off between accuracy and latency. Beyond board games, the findings of this study have broader implications for AI research in strategic planning, optimization, and human-AI collaboration. Techniques developed for structured game environments can be extended to real-world applications, such as autonomous systems, logistics, and decision-support tools. Future research directions include improving model interpretability, further reducing computational cost, and extending the framework to imperfect-information and multi-agent environments. Continued exploration of hybrid and federated learning approaches may further enhance the scalability and practical applicability of AI systems inspired by board games.

Acknowledgment

The authors express their sincere gratitude to the Department of Information Technology, Vishwakarma Institute of Technology, Pune, for their valuable guidance and support throughout this work.

Conflict of Interest

There is no conflict of interest.

Supporting Information

Not applicable

Use of artificial intelligence (AI)-assisted technology for manuscript preparation

The authors confirm that no artificial intelligence (AI)-assisted technology was used to write or edit the manuscript, and that no images were manipulated using AI.

References

[1] S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson Education, 2021

[2] C. B. Browne; E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, A survey of Monte Carlo Tree Search methods, *IEEE Transactions on Computational Intelligence and AI in Games,* 2012, **4**, 1-43, March 2012, doi: 10.1109/TCIAIG.2012.2186810.

[3] A. del Bosque, P. Fernández-Arias, G. Lampropoulos, D. Vergara, The role of artificial intelligence in gaming, *Applied Sciences*, 2025, **15**, 12358, doi: 10.3390/app152312358.

[4] I. Szita, Reinforcement learning in games. In: Wiering, M., van Otterlo, M. (eds) Reinforcement learning. adaptation, learning, and optimization, Springer, Berlin, Heidelberg, 2012, **12**, doi: 10.1007/978-3-642-27645-3_17.

[5] T. R. Robbins, The games AIs play - a comprehensive review, *Journal of Applied Business and Economics*, 2025, **27**, doi: 10.33423/jabe.v27i6.7948.

[6] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, D. Hassabis, A general reinforcement learning algorithm that masters chess, shogi, and go through self-play, *Science*, 20218, **362**, 1140–1144, doi: 10.1126/science.aar6404.

[7] N. Brown, T. Sandholm, Libratus: The superhuman AI for no-limit poker, Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI'17), AAAI Press, 2017, 5226–5228.

[8] C. Hu, Y. Zhao, Z. Wang, H. Du, J. Liu, Games for artificial intelligence research: a review and perspectives, *IEEE Transactions on Artificial Intelligence*, 2024, **5**, 5949-5968, doi: 10.1109/TAI.2024.3410935.

[9] L. Kocsis, C. Szepesvri, Bandit based monte-carlo planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds) Machine Learning: ECML 2006. ECML 2006. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2006, **4212,** doi: 10.1007/11871842_29.

[10] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, D. Silver, Mastering Atari, Go, chess, and shogi by planning with a learned model, *Nature*, 2020, **588**, 604–609, doi: 10.1038/s41586-020-03051-4.

[11] D. E. Knuth, R. W. Moore, An analysis of alpha–beta pruning, *Artificial Intelligence*, 1975, **6**, 293–326, doi: 10.1016/0004-3702(75)90019-3.

[12] Y. Lu, W. Li, Techniques and paradigms in modern game ai systems, *Algorithms*, 2022, **15**, 282, doi: 10.3390/a15080282

[13] M. Campbell, A. J. Hoane Jr., F. H. Hsu, Deep blue, *Artificial Intelligence*, 2022, 134(1–2), 57–83, doi: 10.1016/S0004-3702(01)00129-1.

[14] Monty Newborn, Deep Blue: An Artificial Intelligence Milestone, Springer New York, NY, 2003, doi: 10.1007/978-0-387-21790-1.

[15] M. Świechowski, K. Godlewski, B. Sawicki, J. Mańdziuk, Monte Carlo Tree Search: a review of recent modifications and applications, *Artificial Intelligence Review*, 2023, **56**, 2497–2562, doi: 10.1007/s10462-022-10228-y.

[16] M. Świechowski, H. Park, J. Mańdziuk, K-J. Kim, recent advances in general game playing, *The Scientific World Journal*, 2015, 986262, doi: 10.1155/2015/986262.

[17] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of Go with deep neural networks and tree search, *Nature*, 2016, **529**, 484–489, doi: 10.1038/nature16961.

[18] A. Liu, AI techniques in board game: A survey, *Applied and Computational Engineering*, 2024, 79, 49-59, doi: 10.54254/2755-2721/79/20241297.

[19] J. Hu, F. Zhao, J. Meng, S. Wu, Application of Deep Reinforcement Learning in the Board Game, 2020 IEEE International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA), Chongqing, China, 2020, 809-812, doi: 10.1109/ICIBA50161.2020.9277188.

[20] K. Xenou, G. Chalkiadakis, S. Afantenos, Deep Reinforcement Learning in Strategic Board Game Environments. In: Slavkovik, M. (eds) Multi-Agent Systems. EUMAS 2018. Lecture Notes in Computer Science, 2019, 11450. Springer, Cham, doi: 10.1007/978-3-030-14174-5_16.

[21] S. Gelly, D. Silver, Combining online and offline knowledge in UCT, Proceedings of the 24th International Conference on Machine Learning *(ICML)*, 2007, 273-280, doi: 10.1145/1273496.127353.

[22] N. Justesen, P. Bontrager, J. Togelius, S. Risi, Deep learning for video game playing, *IEEE Transactions on Games*, 2019, 12, 1–13, doi: 10.1109/TG.2019.2896986.

[23] S. Liu, J. Cao, Y. Wang, W. Chen, Y. Liu, Self-play reinforcement learning with comprehensive critic in computer
Games, *Neurocomputing*, 2021, **449**, 207-213, doi: 10.1016/j.neucom.2021.04.006.

[24] C. Jiang, The application of artificial intelligence in board games, Proceedings of the 3rd International Conference on Signal Processing and Machine Learning, doi: 10.54254/2755-2721/4/20230497.

[25] H. Jiang, Applications of artificial intelligence in game algorithms: history, current status, and future prospects,

Proceedings of the 2024 International Conference on Artificial Intelligence and Communication (ICAIC 2024), doi: 10.2991/978-94-6463-512-6_45.