Evaluating Game Difficulty in Tetris Block Puzzle

Chun-Jui Wang, Jian-Ting Guo, [1] Hung Guei, Chung-Chin Shih, [2]
Ti-Rong Wu, [2,a)] I-Chen Wu[1,2]

Abstract:
Tetris Block Puzzle is a single-player stochastic puzzle in which a player places blocks on an 8×8 grid to complete lines; its popular variants have amassed tens of millions of downloads. Despite this reach, there is little principled assessment of which rule sets are more difficult. Inspired by prior work that uses AlphaZero as a strong evaluator for chess variants, we study difficulty in this domain using *Stochastic Gumbel AlphaZero (SGAZ)*, a budget-aware planning agent for stochastic environments. We evaluate rule changes—including holding block h, preview holding block p, and additional Tetris block variants —using metrics such as training reward and convergence iterations. Empirically, increasing h and p reduces difficulty (higher reward and faster convergence), while adding more Tetris block variants increases difficulty, with the T-pentomino producing the largest slowdown. Through analysis, SGAZ delivers strong play under small simulation budgets, enabling efficient, reproducible comparisons across rule sets and providing a reference for future design in stochastic puzzle games.

Keywords: Puzzle game, Game rule variants, Stochastic gumbel alpha zero, Game difficulty

1. Introduction

Reinforcement learning (RL) has achieved significant success across a wide range of games. Notably, AlphaGo [10] employed deep neural networks trained on human game records to achieve superhuman performance in Go. Its successor, AlphaGo Zero [12], removed the need for human data by learning entirely through self-play, surpassing the original AlphaGo. Finally, AlphaZero [11] generalized this approach beyond a single game, achieving state-of-the-art (SOTA) results in many board games, such as Go, chess, and shogi.

After the success of AlphaZero, several extensions have been developed in different directions. For example, Stochastic AlphaZero [1] was introduced to handle stochastic environments where randomness affects state transitions and outcomes. Meanwhile, because the standard AlphaZero framework provides no guarantee of policy improvement, Gumbel AlphaZero [3] was proposed to address this limitation. It ensures policy improvement, allowing effective training even under a very small number of simulation budgets.

Furthermore, as AlphaZero-based algorithms have achieved superhuman performance, they have also been used to explore novel rule sets [15] and to control game difficulty [4], [5]. Following this approach, we adopt a similar approach in the *Tetris Block Puzzle* [14], a single-player stochastic puzzle game, with Stochastic Gumbel AlphaZero to investigate new gameplay variants and analyze their difficulty.

We investigate the difficulty under different rule variants: the number of holding blocks (h), the number of preview holding blocks (p), and the inclusion of additional Tetris block types. We evaluate these variants using two quantitative metrics, including training reward and convergence iterations, to measure how each rule change affects gameplay difficulty. We observe that larger h and p consistently lower difficulty, yielding higher rewards and faster convergence speed for the strong AI agent. Conversely, adding new Tetris block types increases difficulty, most notably the T-pentomino, which produces the largest slowdown in convergence, thereby offering rule configurations that players can select to match their desired challenge.

2. Background

2.1 AlphaZero

AlphaZero [11] is a zero-knowledge learning method that achieves superhuman performance in multiple board games without human knowledge. Training alternates between (i) self-play and (ii) optimization. In self-play, each move is selected by running many simulations of Monte Carlo Tree Search (MCTS) [2] guided by a neural network that outputs a policy and a value.

Each MCTS simulation performs three phases: selection, expansion, and backpropagation. In the selection phase, the search traverses the tree from the root toward a leaf node using a PUCT formula [9]. In the expansion phase, the reached leaf is expanded to the tree, and all legal child nodes are expanded. The neural network then evaluates the state to provide action priors for those children and a value estimate for the leaf. During backpropagation, the value is propagated from the leaf back to the root along the selected path. After the simulation budget is reached, an action is sampled from

[1] National Yang Ming Chiao Tung University, Hsinchu, Taiwan
[2] Academia Sinica, Taipei, Taiwan
[a)] tirongwu@iis.sinica.edu.tw

the root's action distribution, where the probability of each action is proportional to its number of visits during search. The finished self-play games are stored in a replay buffer for optimization. In the optimization phase, a self-play game is sampled randomly from the replay buffer to update the network.

2.2 Gumbel AlphaZero

Although AlphaZero has achieved superhuman performance, it does not guarantee policy improvement when the simulation budget is small. The Gumbel Zero framework [3] was introduced to address this limitation. It modifies the root selection to improve sample efficiency and guarantee policy improvement. At the root, it first performs without replacement candidate selection using the *Gumbel-Top-k trick* [8] on the policy logits, then allocates simulations via sequential halving [7] to progressively prune weaker candidates until a single action remains; that surviving action is taken in the environment deterministically rather than sampling from visit counts. For training targets, since only a limited number of root children are explored, visit-count distributions are not used. Instead, Q-based targets are constructed by using backed-up Q estimates for visited actions and the value network to estimate the values of unvisited ones. These modifications preserve the overall AlphaZero training loop while improving efficiency under small simulation budgets and yielding stronger early training performance in practice.

2.3 Stochastic AlphaZero

To extend AlphaZero to stochastic environments, *Stochastic AlphaZero* [1] introduces afterstates [13] and incorporates a learned model into the planning process. The key idea is to separate the effect of the agent's action from the randomness of the environment. From a given state, applying an action deterministically yields a unique afterstate. With the environment's randomness, this afterstate then produces one of several possible successor states according to a probability distribution.

The learned model contains two additional components. The afterstate dynamic function ϕ predicts the next afterstate as^k given the current state s^{k-1} and chosen action a^k. The afterstate prediction function ψ estimates, for a given afterstate as^k, the set of possible chance outcomes along with their associated probabilities σ^k and the afterstate value Q^k. Then, the dynamic function g maps a sampled chance outcome c^{k+1}, together with the afterstate as^k, to the next state.

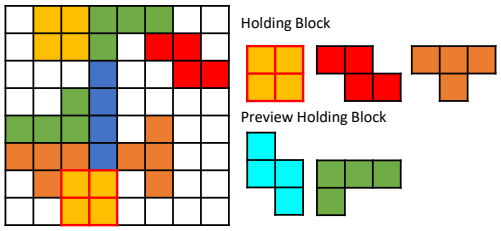
The search in Stochastic AlphaZero constructs a tree that alternates between decision nodes and chance nodes, corresponding to states and afterstates, respectively. During selection at a decision node, the algorithm applies a variant of the PUCT formula to choose an action. At a chance node, it samples an outcome according to the predicted chance probabilities.

Stochastic Gumbel AlphaZero (SGAZ) [6] combines the algorithms of Gumbel AlphaZero and Stochastic AlphaZero to improve training efficiency in stochastic environments. The effectiveness of the integrated method was demonstrated in [6], where Stochastic Gumbel MuZero, an extension of SGAZ, was successfully applied to the popular puzzle game 2048.

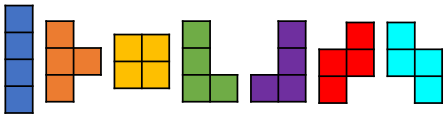
2.4 Evaluating Game Difficulty

Recent studies leverage AI both to regulate challenges and to assess game variants. One line of work is dynamic difficulty adjustment (DDA), where the agent's strength is adapted online to track the player's skill. For example, AlphaDDA uses the standard AlphaZero architecture but adjusts the agent's strength based on the state value in real-time, for example, by changing the number of simulations or increasing dropout to make the agent weaker when needed. Similarly, ROSAS and POSAS [4] both modify the MCTS action-selection policy to match the opponent's skill level. In addition, a related study [15] to our work uses AlphaZero to analyze chess rule variants and quantify balance and decisiveness through expected scores and draw rates, as well as measures of opening diversity and approximate piece values. This paper follows the same *assessment via a strong agent* paradigm in the Tetris Block Puzzle environment.

3. Tetris Block Puzzle



(a) An example of the Tetris Block Puzzle game.



(b) The standard tetromino blocks.

Fig. 1: The gameplay of Tetris Block Puzzle. (a) The player chooses an O-tetromino from three holding blocks and places it (highlighted with a red border), completing a vertical line. This clears the line and earns the player 1 point. After the O-tetromino is taken, the first preview holding block, an S-tetromino, becomes a new holding block for the next turn. Then, a new preview holding block is randomly selected from the standard blocks shown in (b) and may be rotated.

Tetris Block Puzzle [14] is a single-player stochastic puzzle game where the player arranges blocks on an 8×8 grid to perfectly complete lines. As illustrated in Figure 1a, tetromino blocks randomly appear during gameplay, and the player fits them into empty slots on the grid. Once a line is fully placed, its blocks are cleared, and the player earns 1 point. In order to analyze the difficulty of the Tetris Block Puzzle

under different rule settings, we propose to modify the game rules in three specific ways as follows.

Holding Block Rules. The *holding blocks* refer to h candidate blocks, from which the player chooses one to place in each turn. When placing the block, it should be placed in the provided shape without any rotation. After placing, a replacement block is automatically added to the set of holding blocks for the next turn.

Preview Holding Block Rules. The *preview holding blocks*, a feature not present in the classic game rules, are p replacement blocks that will appear in the subsequent turns. Specifically, the first preview holding block becomes a holding block after the player takes one step, and then a new preview holding block is randomly created at the end of the preview holding block sequence.

Tetris Block Variants. In addition to the standard tetromino blocks shown in Figure 1b, other types of Tetris blocks may be included to extend the challenge of the gameplay, e.g., pentomino blocks. In this work, we examine this perspective by introducing the *U-pentomino*, *V-pentomino*, *X-pentomino*, and *T-pentomino* blocks, whose shapes are depicted in Figure 8.

Note that the rules of classic Tetris Block Puzzle are the configuration with $h = 3, p = 0$ and without any additional block.

4. Experiments

This subsection introduces the training process and then analyzes the modified game rules step by step. To assess the difficulties of various Tetris Block Puzzle variants, we implement an extended game environment with adjustable rules using the MiniZero framework [16]. In addition, to evaluate the difficulty of these game variants, we will use two metrics: *Training Rewards* and *Convergence Iterations*, as illustrated below.

Training Rewards. Represented by the average total rewards over the last 50 iterations before training is complete.

Convergence Iterations. Represented by the number of iterations required for the agent to consistently reach the maximum total reward over three consecutive iterations.

These two metrics allow us to evaluate the impact of various game rule changes on the difficulty and will be used for all of the remaining experiments. In addition, all the experiments were run on a machine with four 1080Ti GPUs.

4.1 Training classic Tetris Block Puzzle

First, we train the model under the classic rules setting with $h = 3$ and $p = 0$, using SGAZ for 500 iterations to demonstrate that Tetris Block Puzzle can be effectively trained by SGAZ. The average rewards per iteration are presented in Figure 2. The results show that SGAZ successfully learns the game, reaching an average total reward of 6544, which is very close to the maximum value set in the environment[*1].

[*1] To prevent a game from becoming endless, we set a maximum reward limit of 6750 points.

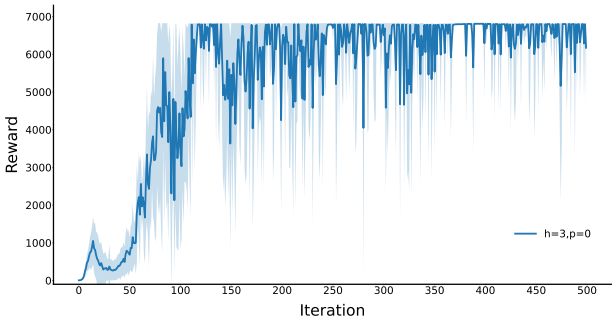


Fig. 2: The training curve for the classic game with $h = 3$ and $p = 0$.

4.2 Training Tetris Block Puzzle variants

Next, following the same training setup, we provide game variants by modifying the number of holding blocks h, preview holding blocks p, and adding additional block types.

4.2.1 Analyzing Holding Block Rules

We evaluate game variants with varying only the number of holding blocks h, while keeping the number of preview holding blocks p fixed at zero.

Training Rewards. The results in Figure 3 and Table 1 show the reward during training. For the setting of $h = 1$, the agent performs extremely poorly, indicating that this game rule is not suitable (too difficult), as SGAZ cannot play effectively. However, for other settings, the agent performs well, and even for the setting of $h = 3$, SGAZ can achieve the maximum total reward.

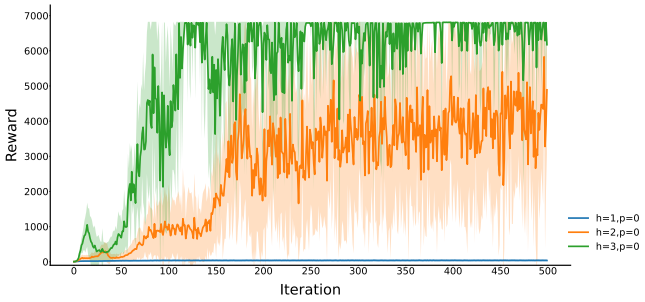


Fig. 3: The training curve for $p = 0$ with different h.

Table 1: The training rewards of the different numbers of holding blocks.

h	1	2	3
Reward	39.0	4126.1	6544.0

Convergence Iterations. Table 2 reports the convergence iterations. The results show that convergence iterations decrease as the number of holding blocks h increases. This means that with a larger h, the agent converges more quickly, and obviously, the game becomes easier.

Table 2: Convergence iterations for different numbers of holding blocks. The "-" mark indicates that the agent did not converge within training iterations.

h	1	2	3
Convergence speed	-	160	61

4.2.2 Analyzing Preview Holding Block Rules

Furthermore, we analyze the variants of fixing the number of holding blocks h and varying only the number of preview holding blocks p to assess their impact on the game.

Training Rewards. The results, summarized in Figure 4, 5, and 6, show that the training rewards increase as the number of preview holding blocks p grows. In addition, the results indicate that changing the number of preview holding blocks p is less significant compared to changing the number of holding blocks h. Specifically, with $h = 1$ the training rewards increase slowly and converge at about 5000, while for other values of h it can easily exceed this score.

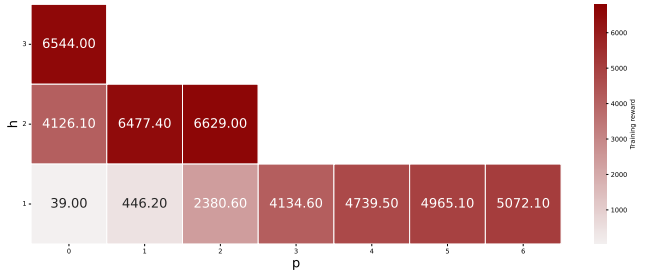


Fig. 4: The training rewards impact of different numbers of preview holding blocks. The blank areas are expected to be optimal, as their preceding settings are already optimal.

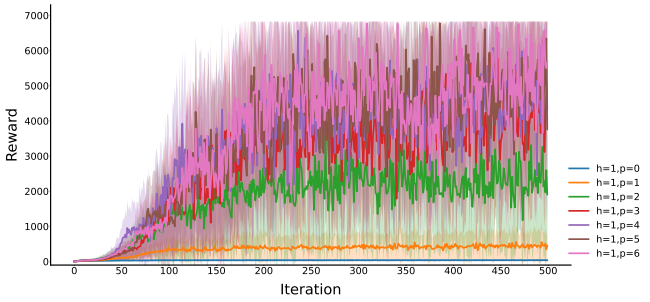


Fig. 5: The training curve of different numbers of preview holding blocks under a fixed setting of holding blocks $h = 1$.

Convergence Iterations. Figure 7 reports the convergence iterations. The results show the same situation as Figure 4, revealing that increasing p slightly decreases the game difficulties.

4.2.3 Analyzing Tetris Block Variants

Finally, we explore adjusting the difficulty by introducing additional blocks. Using the U-pentomino, V-pentomino, X-pentomino, and T-pentomino blocks illustrated in Figure 8, we assess the training performance of adding them under moderate settings of $h = 2$ and $p \in \{0, 1, 2\}$. For the additional blocks setting, we run experiments by adding either one or two blocks during training. As shown in Figure 9, cells on the main diagonal (e.g., (U, U) or (V, V)) correspond to runs with a single added block, while off-diagonal cells (e.g., (U, V) or (U, X)) correspond to runs with two added blocks.

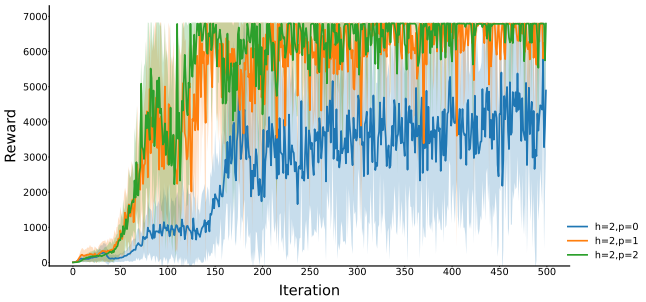


Fig. 6: The training curve of different numbers of preview holding blocks under a fixed setting of holding blocks $h = 2$.

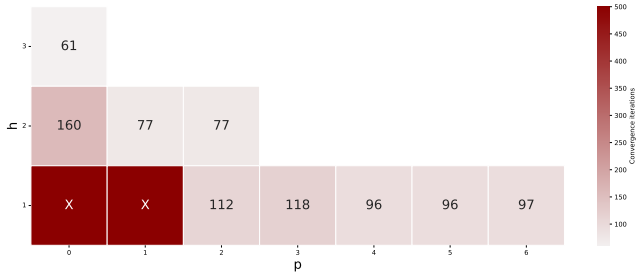


Fig. 7: Impact of different numbers of preview holding blocks on convergence iterations. The mark of "X" area indicates that the agent did not converge within training iterations.

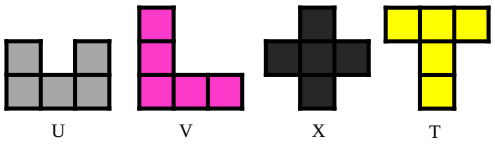


Fig. 8: The U,V,X,T-pentomino blocks.

Training Rewards. The results in Figure 9 show that the training rewards decrease when any additional block is added, compared with Figure 4. In addition, we can see that adding the T-pentomino has the largest impact, as the training rewards decrease more with the T-pentomino than with the other blocks. This indicates that adding additional blocks would increase the game's difficulty.

Convergence Iterations. Figure 10 reports the convergence iterations. Based on the result, adding an additional block will increase the game's difficulty. We can observe that for the experiment setting with $h = 2$ and $p = 0$, all experiments involving the addition of two blocks did not converge during training. This is a clear difference compared to the results shown in Figure 7, where convergence was easily reached. In addition, the result still shows that the T-pentomino has the largest impact on the agent, as in every experiment where a single block is added, the T-pentomino always causes the slowest convergence speed.

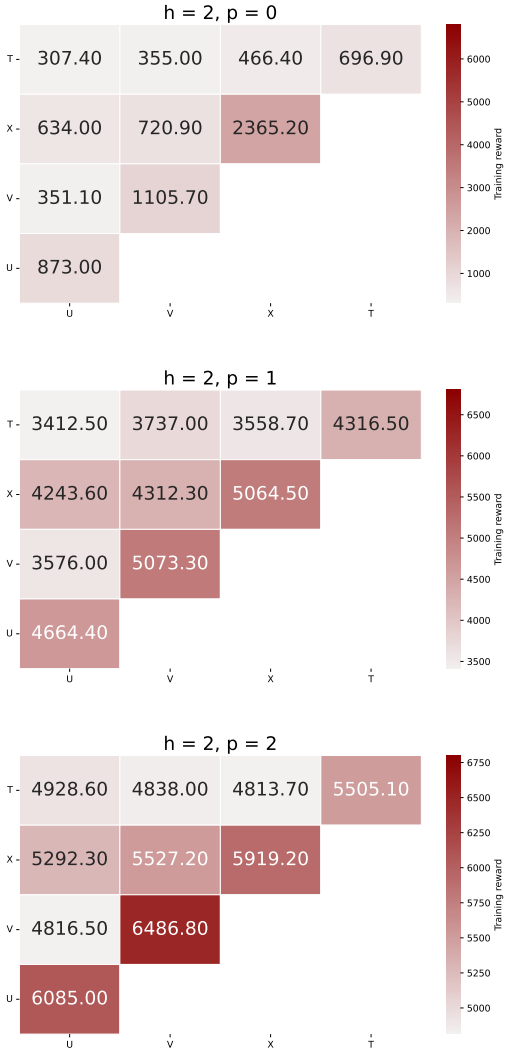


Fig. 9: The training rewards impact of adding additional blocks.

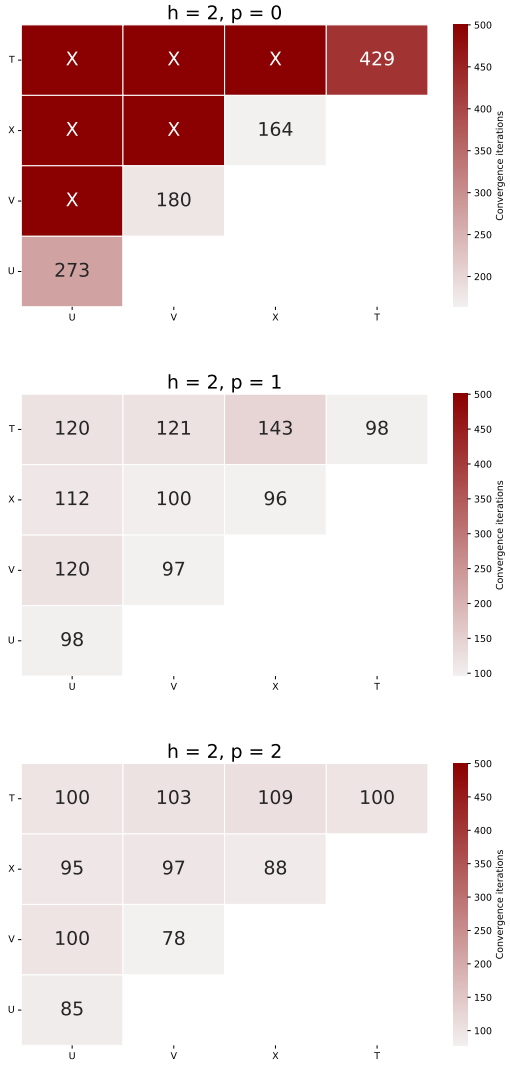


Fig. 10: The convergence iterations impact of adding additional blocks.

5. Discussion

In this paper, we assess the game difficulty of various Tetris Block Puzzle variants. First, we formulate game variants based on three key settings: the holding block rules, the preview holding block rules, and the Tetris block variants. We introduce two metrics to evaluate the difficulty of different game rules: Training rewards and Convergence iterations. Additionally, we demonstrate that Stochastic Gumbel AlphaZero efficiently achieves near-optimal performance under mild settings, making it ideal for difficulty assessment. Most importantly, our analysis indicates that increasing either the number of holding blocks or the number of preview holding blocks makes the game easier, though the former has a markedly stronger impact. However, adding new block types significantly makes the game harder, especially for the T-pentomino block. These insights refine our understanding of the game difficulty of Tetris Block Puzzle and provide a concrete reference for further research on stochastic puzzle games.

Looking ahead, we plan to explore more game variants, such as banning specific blocks from choosing, changing puzzle sizes, and introducing entirely new block types. We also intend to conduct studies to evaluate how these variations affect human players' perceived challenge and enjoyment in the future.

References

[1] Antonoglou, I., Schrittwieser, J., Ozair, S., Hubert, T. K. and Silver, D.: Planning in Stochastic Environments with a Learned Model, *International Conference on Learning Representations* (2021).

[2] Coulom, R.: Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search, *Computers and Games*, Lecture Notes in Computer Science, Berlin, Heidelberg, Springer, pp. 72–83 (2007).

[3] Danihelka, I., Guez, A., Schrittwieser, J. and Silver, D.: Policy Improvement by Planning with Gumbel, *International Conference on Learning Representations* (2022).

[4] Demediuk, S., Tamassia, M., Raffe, W., Zambetta, F., Li, X. and Mueller, F.: Monte Carlo Tree Search Based Algorithms for Dynamic Difficulty Adjustment, *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 53–59 (2017).

[5] Fujita, K.: AlphaDDA: Strategies for Adjusting the Playing Strength of a Fully Trained AlphaZero System to a Suitable Human Training Partner, *PeerJ Computer Science*, Vol. 8, p.

e1123 (2022).

[6] Kao, C.-Y., Guei, H., Wu, T.-R. and Wu, I.-C.: Gumbel MuZero for the Game of 2048, *2022 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, Tainan, Taiwan, IEEE, pp. 42–47 (2022).

[7] Karnin, Z., Koren, T. and Somekh, O.: Almost Optimal Exploration in Multi-Armed Bandits, *Proceedings of the 30th International Conference on Machine Learning*, PMLR, pp. 1238–1246 (2013).

[8] Kool, W., Hoof, H. V. and Welling, M.: Stochastic Beams and Where To Find Them: The Gumbel-Top-k Trick for Sampling Sequences Without Replacement, *Proceedings of the 36th International Conference on Machine Learning*, PMLR, pp. 3499–3508 (2019).

[9] Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T. and Silver, D.: Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model, *Nature*, Vol. 588, No. 7839, pp. 604–609 (2020).

[10] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T. and Hassabis, D.: Mastering the Game of Go with Deep Neural Networks and Tree Search, *Nature*, Vol. 529, No. 7587, pp. 484–489 (2016).

[11] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K. and Hassabis, D.: A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go through Self-Play, *Science*, Vol. 362, No. 6419, pp. 1140–1144 (2018).

[12] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T. and Hassabis, D.: Mastering the Game of Go without Human Knowledge, *Nature*, Vol. 550, No. 7676, pp. 354–359 (2017).

[13] Sutton, R. S. and Barto, A. G.: *Reinforcement Learning: An Introduction*, Adaptive Computation and Machine Learning Series, MIT Press, Cambridge, MA, USA, 2 edition (2018).

[14] Tetris Holding, LLC: Tetris® Block Puzzle, `https://tetris.com/products/video-game/tetris-block-puzzle` (2025). Accessed July 29, 2025.

[15] Tomašev, N., Paquet, U., Hassabis, D. and Kramnik, V.: Assessing Game Balance with AlphaZero: Exploring Alternative Rule Sets in Chess (2020).

[16] Wu, T.-R., Guei, H., Peng, P.-C., Huang, P.-W., Wei, T. H., Shih, C.-C. and Tsai, Y.-J.: MiniZero: Comparative Analysis of AlphaZero and MuZero on Go, Othello, and Atari Games, *IEEE Transactions on Games*, Vol. 17, No. 1, pp. 125–137 (2025).