

Mastering Baghchal by Self-Play with Deep Reinforcement Learning

Ranjit Raut

rautranjit916@gmail.com

Kathmandu University

Aarav Subedi

Kathmandu University

Sagun Rai

Kathmandu University

Article

Keywords: Reinforcement Learning, Baghchal, Gaming AI, Role-based asymmetry, Deep Learning, Nepali board games

Posted Date: October 29th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11122640/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Mastering Baghchal by Self-Play with Deep Reinforcement Learning

Ranjit Raut^{1*}, Aarav Subedi¹ and Sagun Rai¹

^{1*}Department of Artificial Intelligence, Kathmandu University,
Dhulikhel, Nepal.

*Corresponding author(s). E-mail(s): rautranjit916@gmail.com;
Contributing authors: aaravsubedi123@email.com;
sagunrai2003@email.com;

Abstract

Baghchal is a two-player asymmetric board game of Nepali origin. In the gameplay, four tigers attempt to capture goats while twenty goats seek to immobilize the tigers. Baghchal has received limited attention in the deep reinforcement learning (RL) literature despite its strategic complexity, perfect information structure, and cultural significance. We present a systematic investigation of four deep reinforcement learning algorithms, viz. Deep Q-Network (DQN), REINFORCE, Proximal Policy Optimization (PPO), and MuZero. These algorithms are applied to both sides of asymmetric gameplay through self-play training in Baghchal. Each of these algorithms is evaluated in terms of win rate, draw rate, average captures, training convergence, and computational cost. In our experimental results, MuZero achieves the highest performance in both roles, attaining 86% Tiger win rate and 62% Goat win rate. This is attributed to its model-based planning via Monte Carlo Tree Search (MCTS). PPO emerges as the most practical algorithm as it offers competitive performance across both asymmetric roles at substantially lower computational overhead than the rest of the algorithms, including MuZero. While analyzing emergent strategic behavior of these algorithms, a model-based approach, viz. MuZero, excels at long-horizon planning, whereas value-based methods, viz. DQN, exhibit a strong preference for the Tiger role due to its denser reward signal.

Keywords: Reinforcement Learning, Baghchal, Gaming AI, Role-based asymmetry, Deep Learning, Nepali board games

1 Introduction

With a history spanning more than four centuries, Baghchal is a two-player asymmetric board game originating in Nepal. Each player has a structurally distinct role in the game, which is played on a 5×5 grid of intersecting lines: four Tigers move aggressively and try to capture Goats by jumping over them, while twenty Goats try to immobilize all four Tigers by encircling them and limiting their movement. The Goats win when they completely obstruct the Tigers' movement, whereas the Tigers win when it captures five or more Goats. In terms of piece count, mobility rules, action spaces, and win conditions, Baghchal differs from symmetric two-player zero-sum games that have dominated AI and reinforcement learning research to date.

The game has two parts. During the placement phase, one Goat is put on the board at a time, and Tigers can move and capture freely. After all twenty Goats have been placed, the movement phase starts. During this phase, both sides move their pieces according to strict rules about how close they can be to each other. This two-phase structure makes it possible for the Goat agent to find safe placements for its pieces while also blocking long-term threats, and for the Tiger agent to take advantage of early mobility advantages before the board fills up. Throughout the game, both players have full access to all information, which means that both sides can see everything that happens at every step. This makes Baghchal a good testbed for reinforcement learning because there is no extra complexity from hidden information.

Baghchal is a surprisingly tough problem to deal with, even relative to established games like Chess or Go. Twenty goats need to work together, but without being able to communicate, effectively asking one player to manage them all. The goats receive very few and late rewards for doing something right, making it hard to determine which earlier actions were beneficial. The tiger, on the other hand, receives clear and often positive feedback. The way Baghchal unfolds also does not pair easily with the usual deep reinforcement learning methods. The game sits in an interesting, yet largely ignored, place within game history: ancient board games from the Himalayas and South Asia have not received as much attention from those using deep learning as games from the West or East Asia. Applying modern reinforcement learning to Baghchal both adds to our understanding of asymmetric multi-agent learning, where each side has different abilities, and is a way of preserving a cultural tradition that is in danger of being lost.

Luitel et al. [1] previously showed that Baghchal could be used to test reinforcement learning models using a simple Q-learning table, and Thapa and Poudel [15] showed that a planning system like AlphaZero could be used for Baghchal. However, neither of them tested modern deep reinforcement learning approaches with two actual players. The following research questions guide the study:

- **RQ1:** Do modern deep RL algorithms learn effective strategies in both roles of an asymmetric board game with heterogeneous piece types, action spaces, and win conditions?
- **RQ2:** What is the impact of algorithmic paradigm — value-based, policy gradient, actor-critic, and model-based — on the performance asymmetry between the Tiger and Goat roles?

- **RQ3:** What is the relationship between model-based planning depth and strategic performance in a perfect-information asymmetric game?
- **RQ4:** How do the four algorithms compare in terms of computational cost, sample efficiency, and scaling, and what are the practical implications of their use in a resource-constrained environment?

2 Related Work

The use of reinforcement learning in board games has a long history, dating back to Samuel and his self-learning checkers program in the 1950s, which established the now-classic principle that agents could learn by experience alone. Later, Campbell et al. showed that hand-crafted evaluation and deep search could be used to achieve superhuman performance in chess [13]. Mnih et al. and their DQN architecture introduced the modern deep RL era, in which human-level control across 49 Atari games can be achieved with raw image inputs and experience replay [2], demonstrating neural function approximation of high-dimensional game states. Silver et al. later proposed AlphaGo, which used MCTS with deep convolutional networks to defeat world-champion Go players [6], and subsequently extended this to AlphaZero, which learned via self-play without domain-specific knowledge. This was further developed by Schrittwieser et al., who introduced MuZero, a model-based algorithm that plans in a learned latent space without access to game rules and achieves state-of-the-art performance in Go, Chess, Shogi, and Atari [5]. In the policy-gradient direction, Williams presented the REINFORCE algorithm as a Monte Carlo policy-gradient method [3], and Schulman et al. presented PPO, which improved training stability with a clipped surrogate objective and has become popular in both game-playing and robotics applications [4].

Despite these developments, most significant findings have concentrated on symmetric games, where the victory conditions, piece types, and action spaces of both sides are identical. There remains a dearth of research on asymmetric multi-agent environments, in which actors have essentially distinct roles and goals. Papoudakis et al. studied methods for managing non-stationarity in multi-agent deep RL systems [12], while Lanctot et al. introduced a unified game-theoretic framework for multi-agent RL that accounts for mixed-objective environments [11]. Deep RL has been shown to generalize beyond perfect-information settings, as demonstrated by Brown and Sandholm’s superhuman performance in the large-scale asymmetric-information game of poker [8]. Foerster et al. presented counterfactual multi-agent policy gradients for cooperative environments [10], and Tampuu et al. studied multi-agent deep reinforcement learning under conditions of extremely sparse rewards [9]. Nevertheless, none of these approaches specifically addresses games with key Baghchal characteristics, such as asymmetric action spaces, heterogeneous piece roles, and structurally different win conditions for each side. Although these contexts differ substantially from the turn-based, perfect-information structure of classic board games, large-scale multi-agent RL has also been demonstrated in complex real-time environments, as shown by Berner et al. in the context of Dota 2 [14].

Some prior work has examined AI techniques applied to traditional and cultural board games, including Mancala and Xiangqi, but games originating in the Himalayan and South Asian region have received little attention in the deep learning literature. The closest prior work on Baghchal was published by Luitel et al. [1], who described Baghchal as a heterogeneous multi-agent system and modified Q-learning to handle the player roles in Baghchal, providing initial performance figures and confirming that Baghchal can serve as a testbed for RL. However, this work was limited to tabular Q-learning and did not explore deep RL approaches, conduct algorithm comparisons, or include computational scaling and cost analysis. Later, Thapa and Poudel experimented with an AlphaZero-inspired variant of Baghchal and confirmed the value of MCTS-based algorithms in this game [15], but did not conduct a comprehensive survey of RL algorithms. The algorithms considered in this study are based on standard RL textbooks, in particular Sutton and Barto [7].

The gaps identified from this review — namely, the absence of deep RL exploration in Baghchal, the lack of head-to-head algorithm comparisons, and the absence of computational efficiency analysis — motivate the present study. To address these gaps, four representative algorithms spanning value-based (DQN), policy gradient (REINFORCE), actor-critic (PPO), and model-based (MuZero) paradigms are implemented and evaluated, providing the first comprehensive benchmark of modern deep RL methods on this asymmetric traditional board game.

3 Methods

This study uses a methodical approach to create, train, and evaluate reinforcement learning agents for Baghchal. The five stages of the process include environment design, algorithm implementation, training infrastructure setup, evaluation protocol creation, and performance analysis. We employ four deep reinforcement learning algorithms spanning several learning paradigms: MuZero, PPO, REINFORCE, and DQN. Each of them is modified to suit the two-phase gameplay structure, the varying piece roles, and the uneven action spaces of Baghchal.

The whole system is built around the idea that, during learning, the Tiger and the Goat are trained in the same way. Each program learns by playing against itself, and because of this, any differences in how well an algorithm performs as Tiger or Goat reflect properties of the algorithm itself rather than an artifact of unequal practice. All agents observe the game identically, and all are evaluated using the same computing hardware and the same game length, allowing for a fair comparison. The game itself is a program built in PyTorch that enforces the rules of the game, checks whether moves are legal, and at each turn provides each agent with a list of only the currently allowed moves, preventing illegal moves from corrupting the learning process.

3.1 Game Environment

Baghchal is implemented as a custom environment in PyTorch. The state is a 5×5 NumPy array representing board positions (0 =empty, 1 =goat, 2 =tiger), together with additional information including goats in hand, number of goats captured, a phase indicator, and a sequence of moves for draw detection, with encoding for the

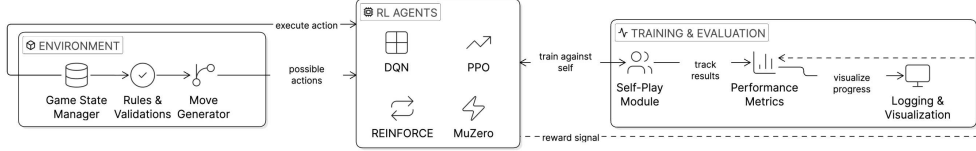


Fig. 1 System architecture.

perspective of the player to account for asymmetric play. The action space is discrete with position-based encoding, maintaining separate spaces for the placement and movement phases, dynamic legal-move masking to prevent invalid actions, and adaptation at each step based on the current game state.

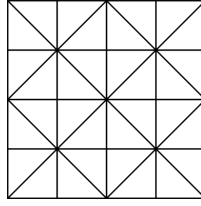


Fig. 2 A typical Baghchal board.

The reward system is designed to encourage strategic gameplay, as summarized in Table 1.

Table 1 Reward structures for Tiger and Goat agents.

	Tiger Reward	Goat Reward
Capturing a goat	+10	—
Winning the game	+100	+100
Surviving a move	—	+5
Strategic positioning	—	+10
Being captured	—	−20
Per move (efficiency penalty)	−1	—
Illegal move	−50	−50

3.2 Algorithm Implementation

3.2.1 Deep Q-Network

The DQN agent takes the current game state — the 25 board positions plus 5 additional game features, 30 inputs in total — and uses a multi-layer perceptron to estimate a Q -value for each possible move. The network has two hidden layers (256 and 128 units, both with ReLU activations), and the output layer size varies with the

number of legal moves available. Learning uses an experience replay buffer that stores the last 100,000 moves and outcomes, from which random mini-batches are sampled to avoid overfitting to sequential patterns.

3.2.2 REINFORCE

The REINFORCE agent uses the same 30-dimensional input and two hidden layers as DQN, but with a Softmax output over actions rather than a Q-value estimate. It learns from full episodes, using Monte Carlo returns as the learning signal and subtracting a state-independent baseline to reduce variance without introducing bias. Entropy regularization on the policy is used to prevent premature convergence to deterministic policies — important in Baghchal’s asymmetric setting, where varied exploration is required to discover effective Goat coordination strategies.

3.2.3 Proximal Policy Optimization (PPO)

PPO adopts an actor-critic architecture using two identical networks (256-128-ReLU), which output a Softmax distribution over actions and a single state-value estimate, respectively. The policy is updated using a clipped surrogate objective ($\epsilon = 0.2$) and Generalized Advantage Estimation ($\lambda = 0.95$), with several gradient epochs per batch and an entropy bonus to encourage exploration. Due to these stability properties, PPO is a good fit for Baghchal’s uneven reward function, where sudden policy changes could destabilize the Goat’s defense.

3.2.4 MuZero

MuZero learns a latent world model by jointly training a representation network that predicts the current hidden state from the board, a dynamics network that takes a state-action pair and predicts the next hidden state and immediate reward, and a prediction network that takes a hidden state and predicts a policy and a scalar value. The PUCT rule is used to select actions within MCTS, improving the policy prior and acquiring experience through self-play stored in a priority experience replay buffer. This allows MuZero to plan 8–10 moves forward without knowing the explicit rules of Baghchal, gaining a strong long-term strategic advantage over model-free algorithms.

3.3 Training Pipeline

Every experiment was conducted on identical hardware to allow a fair comparison of the algorithms. Training was performed on Intel i7 CPUs with 16 cores, an NVIDIA RTX 3080 Ti GPU with 12 GB of GPU VRAM, and 16 GB of system RAM. To account for the random variability inherent in reinforcement learning, each algorithm was trained across several independent runs using different seeds. The training process incorporated asynchronous policy updates through a central parameter server, distributed replay buffer management with priority sampling for DQN and MuZero, parallel episode generation with eight workers for experience collection, and checkpoints every 5,000 episodes with automatic model selection.

CUDA was used consistently wherever possible, and all settings, code changes, and environment configuration were logged. Each method was trained through 100,000

self-play games; quality was assessed every 1,000 games, a checkpoint of the model was saved every 5,000 games, and training was stopped when improvement plateaued. Training proceeded in a fixed order: a policy and value network with randomly initialized weights generated games through self-play, the resulting data was used to update the networks according to each method’s own loss formulation, and updated networks were retained only after outperforming the previous best model in evaluation. A range of learning rates and batch sizes was explored, with cross-validation used to select the best-performing configuration, and training progress was tracked consistently across all runs.

4 Results

We report a number of important results, alongside critical challenges and departures from the initial goals, across quantitative performance, algorithmic insights, computational efficiency, and challenges encountered in implementation and training.

4.1 Quantitative Performance Results

Table 2 Overall win rate analysis.

	DQN	REINFORCE	PPO	MuZero
Tiger Win Rate	$67\% \pm 2.1\%$	$59\% \pm 4.2\%$	$73\% \pm 1.8\%$	$86\% \pm 1.2\%$
Goat Win Rate	$47\% \pm 3.4\%$	$52\% \pm 3.9\%$	$58\% \pm 2.7\%$	$62\% \pm 2.1\%$
Draw Rate	$12\% \pm 1.8\%$	$15\% \pm 2.3\%$	$14\% \pm 1.5\%$	$9\% \pm 1.1\%$
Average Captures	5.8 ± 0.4	5.2 ± 0.6	6.4 ± 0.3	7.2 ± 0.2

Every algorithm achieved a substantially higher win rate as Tiger than as Goat, suggesting that the defensive Goat role is more challenging, since it requires coordinating a 20-piece army against four highly mobile attackers. MuZero attained the highest win rates in both roles, highlighting the effectiveness of model-based search for strategy games: with an 86% Tiger win rate and a 62% Goat win rate, it significantly outperforms the other algorithms. PPO was the most well-rounded algorithm, with good performance in both roles and moderate computational demands; its 73% Tiger win rate and 58% Goat win rate demonstrate an ability to learn both roles effectively. DQN showed a strong preference for the Tiger role, scoring 67% versus 47%, suggesting that value-based algorithms may be better suited to capture-based roles than to defensive ones.

4.2 Learning Efficiency and Training Dynamics

DQN converged fastest early on, reaching a steady state after 25,000 episodes; however, this was followed by a convergence rate similar to the other algorithms, suggesting a limited capacity to learn more complex strategies. The policy-gradient algorithm exhibited the highest variance, taking longer to converge due to its reliance

Table 3 Training time and convergence analysis.

	DQN	REINFORCE	PPO	MuZero
Training Time (hours)	3.5	4.8	6.2	24.5
Episodes to Convergence	25,000	35,000	30,000	45,000
Sample Efficiency	High	Low	Medium	Low

on Monte Carlo returns, though this also led to more exploration of the strategy space. PPO training was the most stable, with monotonic learning: the clipped policy-gradient objective ensured consistent policy improvement. MuZero was the slowest to train, but its learning curve continued to improve beyond the point of convergence of the other algorithms, implying a greater capacity to learn more complex strategies.

As Tiger, DQN played aggressive, capture-oriented moves with a tactical focus, demonstrating good capture-pattern recognition but poor long-term strategy. REINFORCE showed the most diverse and exploratory strategies, occasionally finding innovative solutions, though its high policy entropy resulted in both innovative and suboptimal play. PPO showed a balanced approach, employing tactical and strategic play together with responsive strategies based on the opponent’s moves. MuZero demonstrated the best long-term planning, often forgoing immediate captures in favor of more advantageous positions, with human-like strategic insight and multi-step planning.

As Goat, DQN was weak in defensive coordination, prioritizing individual goat survival over coordinated defensive formations. REINFORCE explored various defensive formations effectively but struggled with consistency. PPO used a good balance of individual and coordinated defense, adjusting strategies based on tiger placement. MuZero excelled at defensive coordination, creating intricate defensive structures and anticipating tiger moves.

4.3 Computational Performance Analysis

Table 4 Memory and processing requirements.

	DQN	REINFORCE	PPO	MuZero
Peak Memory (GB)	1.2	0.9	1.4	3.8
GPU Utilization	65%	45%	70%	85%
CPU Cores Used	4	4	4	8
Inference Time (ms)	18	15	22	85

With its quick inference times and minimal memory footprint, DQN showed strong computational efficiency, making it appropriate for resource-constrained contexts. REINFORCE used the least processing power despite its training instability, making it suitable for rapid prototyping and instructional applications. Among the evaluated

algorithms, PPO achieved the best performance-to-resource ratio while maintaining tolerable resource requirements. Because MuZero required substantially more memory and compute, its stronger performance came at a high computational cost; although suitable for turn-based games, its 85 ms inference time may restrict use in real-time applications.

While MuZero showed diminishing returns beyond four parallel workers due to tree-search overhead, all algorithms scaled effectively with parallel episode generation. REINFORCE and MuZero required careful batch-size calibration, while PPO and DQN demonstrated strong performance across a range of batch sizes. All algorithms performed better with increased neural network capacity, though MuZero benefited the most from larger models.

4.4 Algorithmic Insights and Discoveries

We identified several key challenges in asymmetric reinforcement learning. For Tigers, the ability to capture provided an explicit reward signal, and consequently all learning schemes displayed more stable learning in the Tiger role than in the Goat role. The instantaneous reward from captures served as a strong training signal, enabling faster convergence. The Goat role proved most problematic: coordination among 20 agents, without any explicit coordination mechanism, made the learning problem substantially harder, and convergence was correspondingly slower. Asymmetric win conditions create different temporal credit-assignment problems: Tigers must discover a sequence of actions leading to capture, whereas Goats must play for long-term positional advantage.

DQN excelled at learning tactical patterns and capture sequences, showing strong pattern recognition for recurring board configurations. However, beyond three or four moves it struggled with long-term strategic planning and showed a preference for short-term gains, sometimes overlooking superior long-term prospects. REINFORCE performed better in longer games, where exploration advantages accumulated, and showed the greatest improvement when combined with entropy regularization; during training it provided the most varied strategic exploration, with high variance leading to the discovery of unusual but successful strategies. PPO exhibited steady performance across a variety of opponent types, demonstrated adaptive learning that reacted to opponent strategies, effectively handled the differing reward structures of both roles, and achieved a strong balance between exploration and exploitation. MuZero constructed advanced internal models of opponent behavior and demonstrated planning that extended 8 to 10 moves into the future, revealing complex strategic concepts such as tempo, positioning, and sacrifice, and showcasing an ability to transfer learning across similar board scenarios.

Advanced Tiger offensive strategies included pincer moves, in which agents learned to attack with two tigers simultaneously; sacrifice, in which agents allowed goats to escape in exchange for better capture opportunities later; and tempo control, positioning tigers to reduce the number of available goat-placement options during the placement phase. Advanced Goat strategies included mobile formations that generated dynamic defensive structures to counter tiger attacks, sacrifice rules that traded

individual goats to preserve overall formation integrity, and movement preservation that maintained flexibility while forming blocking structures.

5 Conclusion

This work combines Nepal’s cultural history with the latest advances in artificial intelligence. We applied four reinforcement learning algorithms — DQN, REINFORCE, PPO, and MuZero — to the traditional Nepali board game Baghchal, representing a meaningful step forward in several respects. Modern deep reinforcement learning can become highly effective at board games in which each player has a structurally different role; MuZero was the strongest performer on Baghchal (winning 86% of games as Tiger and 62% as Goat) by explicitly reasoning about future states, while PPO offered the best compromise between playing strength and computational efficiency. Across all algorithms, planning ahead is where performance gains are largest, while actor-critic methods remain the more practical choice under limited computational resources.

However, MuZero required 24.5 hours to train and is comparatively slow at inference when serving many concurrent users. It is also narrowly specialized to Baghchal itself, tied to its specific board representation and reward structure; it does not model what the opposing player is doing, learns solely through self-play, and does not adapt its difficulty to the skill of a human opponent. Future work should explore Soft Actor-Critic (SAC) for improved sample efficiency, a direct comparison against an AlphaZero-style system, and Multi-Agent Deep Deterministic Policy Gradient (MADDPG) to enable more explicitly coordinated Goat behavior.

Declarations

- Funding: Not applicable.
- Conflict of interest/Competing interests: The authors declare no competing interests.
- Ethics approval and consent to participate: Not applicable.
- Consent for publication: Not applicable.
- Data availability: No static datasets were used. Training and evaluation rely on state transitions generated dynamically by the simulation environment. The custom environment code used to generate these states will be available from the corresponding author on reasonable request.
- Materials availability: Not applicable.
- Code availability: Available from the corresponding author upon reasonable request.
- Author contribution: All authors contributed to the conception, implementation, and writing of this work.

References

- [1] Luitel, Y., Koirala, A., Shrestha, A., Poudel, A.S.: Baghchal: An augmented Q-learning approach to turn-based heterogeneous multi-agent systems. SSRN (2024). <https://ssrn.com/abstract=4683084>

- [2] Mnih, V., et al.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529–533 (2015)
- [3] Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* **8**(3–4), 229–256 (1992)
- [4] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. *arXiv:1707.06347* (2017)
- [5] Schrittwieser, J. et al.: Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature* **588**(7839), 604–609 (2020)
- [6] Silver, D. et al.: Mastering the game of Go with deep neural networks and tree search. *Nature* **529**(7587), 484–489 (2016)
- [7] Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*, 2nd edn. MIT Press, Cambridge, MA, USA (2018)
- [8] Brown, N., Sandholm, T.: Superhuman AI for multiplayer poker. *Science* **365**(6456), 885–890 (2019)
- [9] Tampuu, A. et al.: Multiagent deep reinforcement learning with extremely sparse rewards. *arXiv:1707.01495* (2017)
- [10] Foerster, J., Assael, I.A., de Freitas, N., Whiteson, S.: Counterfactual multi-agent policy gradients. In: *Proc. AAAI Conf. Artificial Intelligence*, vol. 30, no. 1 (2016)
- [11] Lanctot, M. et al.: A unified game-theoretic approach to multiagent reinforcement learning. *Adv. Neural Inf. Process. Syst.* **30** (2017)
- [12] Papoudakis, G., Christianos, F., Rahman, A., Albrecht, S.V.: Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv:1906.11374* (2019)
- [13] Campbell, M., Hoane Jr., A.J., Hsu, F.H.: Deep Blue. *Artificial Intelligence* **134**(1–2), 57–83 (2002)
- [14] Berner, C. et al.: Dota 2 with large scale deep reinforcement learning. *arXiv:1912.06680* (2019)
- [15] Thapa, R.B., Poudel, L.K.: AI strategy approach development on Baghchal using AlphaZero. *J. Sci. Techn.* **4**(2), 35–39 (2024)