АҚПАРАТТЫҚ-КОММУНИКАЦИЯЛЫҚ ТЕХНОЛОГИЯЛАР
ИНФОРМАЦИОННО-КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ
INFORMATION AND COMMUNICATION TECHNOLOGIES

ЖАСАНДЫ ИНТЕЛЛЕКТ
ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ
ARTIFICIAL INTELLIGENCE

DOI 10.51885/1561-4212_2025_1_187
IRSTI 28.23.17

M. Nurtay[1], A. Akhmetov[2], A. Tau[3], G. Alina[4], N. Mutovina[5]
Abylkas Saginov Karagandy Technical University, Karagandy, Kazakhstan
[1]*E-mail: solano.lifan2@bk.ru**
[2]*E-mail: furybam@gmail.com*
[3]*E-mail: ardak.tau@mail.ru*
[4]*E-mail: alinagaukhar@gmail.com*
[5]*E-mail: mutovina_natalya@mail.ru*

APPLICATION OF REINFORCEMENT LEARNING TO DEVELOP AN AI MODEL FOR GAME "TOGYZ QUMALAQ"

«ТОҒЫЗ ҚҰМАЛАҚ» ОЙЫНЫНА АРНАЛҒАН ЖАСАНДЫ ИНТЕЛЛЕКТ МОДЕЛІН ӘЗІРЛЕУ КЕЗІНДЕ НЫҒАЙТЫП ОҚЫТУДЫ ҚОЛДАНУ

ПРИМЕНЕНИЕ ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ ПРИ РАЗРАБОТКЕ МОДЕЛИ ИИ ДЛЯ ИГРЫ «ТОГЫЗ КУМАЛАК»

Abstract. *Logical games often require players to solve various puzzles and strategic challenges. Thanks to the active implementation of Artificial Intelligence, it has possible to use Deep Learning models in such games, which has led to a significant breakthrough in solving other related tasks in the field. This paper presents a study on the development and training of two Reinforcement Learning algorithms, Q-learning and Deep Q-Network (DQN) for playing the game Togyz qumalaq. Both models were trained and evaluated in a game against MiniMax, which was also implemented by the authors of this research. The experiments were conducted with MiniMax recursion depths of 2, 3, and 4, respectively. The article presents the training parameters of models based on Q-learning and DQN, which achieved the best results. For each of the models, reward and training episode graphs are provided. The paper also includes the architecture of DQN, which demonstrated promising results. The results show that DQN has achieved significant success in solving this task, demonstrating notable performance. A comparative analysis also revealed that, unlike DQN, Q-learning requires more significant computational and memory resources for training.*

Keywords: *Togyz qumalaq, Reinforcement Learning, Q-learning, Deep Q-Network (DQN), MiniMax, Algorithms, Game AI, Agent performance.*

Аңдатпа. *Логикалық ойындар көбінесе ойыншылардан әртүрлі басқатырғыштар (головоломки) мен стратегиялық міндеттерді шешуді талап етеді. Мұндай ойындарда жасанды интеллектті белсенді енгізу арқылы терең оқыту модельдерін қолдану мүмкіндігі пайда болды, бұл осы саладағы басқа да байланысты мәселелерді шешуде айтарлықтай жетістіктерге әкелді. Бұл мақалада тоғыз құмалақ ойынына арналған екі күшейтілген оқыту алгоритмін: Q-learning және Deep Q-Network (DQN) әзірлеу және оқыту бойынша зерттеу ұсынылған. Екі модель де MiniMax-қа қарсы ойында оқытылды және бағаланды, оны да осы зерттеудің авторлары жүзеге асырды. Эксперименттер сәйкесінше MiniMax 2, 3 және 4 рекурсия тереңдігімен жүргізілді. Мақалада ең жақсы нәтижелерге қол жеткізген Q-learning және DQN негізіндегі модельдерді оқыту параметрлері берілген. Модельдердің әрқайсысы үшін марапаттау кестелері мен оқу эпизодтары келтірілген. Құжат сонымен қатар перспективалы нәтижелер көрсеткен DQN архитектурасын қамтиды. Нәтижелер DQN бұл мәселені шешуде айтарлықтай жетістіктерге жеткенін және*

елеулі өнімділікті көрсеткенін білдіреді. Сонымен қатар, салыстырмалы талдау Q-оқытудың DQN-нен айырмашылығы, оқыту үшін едәуір есептеу ресурстары мен жад ресурстарын қажет ететіндігін көрсетті.

Түйін сөздер: *Тоғыз құмалақ, күшейтілген оқыту, Q-оқыту, терең Q-желі (DQN), минимакс, алгоритмдер, ойын AI, агент өнімділігі.*

Аннотация. *Логические игры часто требуют от игроков решения различных головоломок и стратегических задач. Благодаря активному внедрению искусственного интеллекта в таких играх появилась возможность использовать модели глубокого обучения, что привело к значительному прорыву в решении других смежных задач в этой области. В этой статье представлено исследование по разработке и обучению двух алгоритмов обучения с подкреплением: Q-learning и Deep Q-Network (DQN) для игры Тоғыз кумалак. Обе модели были обучены и оценены в игре против MiniMax, которую также реализовали авторы данного исследования. Эксперименты проводились с глубиной рекурсии MiniMax 2, 3 и 4 соответственно. В статье представлены параметры обучения моделей на основе Q-learning и DQN, которые достигли наилучших результатов. Для каждой из моделей представлены графики вознаграждений и эпизодов обучения. В документ также включена архитектура DQN, которая продемонстрировала многообещающие результаты. Результаты показывают, что DQN добилась значительных успехов в решении этой задачи, продемонстрировав заметную производительность. Сравнительный анализ также показал, что, в отличие от DQN, Q-обучение требует для обучения более значительных вычислительных ресурсов и ресурсов памяти.*

Ключевые слова: *Тоғыз кумалак, обучение с подкреплением, Q-обучение, Глубокая Q-сеть (DQN), минимакс, алгоритмы, игровой ИИ, производительность агента.*

Introduction. **Artificial intelligence (AI) and Machine Learning (ML) are among the most promising and rapidly evolving fields in technology today. Moreover, the field of reinforcement learning (RL) is garnering increasing attention and finding broad applications across various domains. One of the intriguing tasks in which RL demonstrates its power is the creation and training of artificial intelligences capable of competing in traditional and culturally significant national games.**

The evolution of Reinforcement Learning (RL) in the context of games such as Go, chess, checkers and strategy games is a fascinating and important part of Artificial Intelligence. First of all, this study will introduce into the history and point out the key points, including the application of RL to the AlphaGo and AlphaZero models.

The first steps in the development of game algorithms in the 1950s and 60s were the Machine "Shannon's Type B" (Zhumanov A. et al, 2021) and the computer program "Samuel's Checkers Player" for checkers.

In the 1990s and 2000s, computer programs for chess (Deep Thought (Turgumbaev T. et al., 2019) Deep Blue became stronger due to the use of traditional methods such as MiniMax and alpha-beta clipping.

AlphaGo developed by DeepMind, became the first significant step in applying RL to Go, one of the most complex strategic games.

AlphaGo stunned the world in 2016 by defeating Go world champion Li Sedol. The model uses a combination of Deep Neural Networks and Reinforcement Learning methods to improve a game.

After the success of AlphaGo in 2017, DeepMind released a version of AlphaGo Zero, which was trained from scratch using only the most basic information about the rules of the game and self-playing metrics, and achieved even more impressive results (Vasiliev A. et al., 2020).

In 2017, DeepMind advanced its research with the creation of the AlphaZero (Vasiliev A. et al, 2020) model, capable of learning not only Go, but also chess and checkers with amazing performance.

AlphaZero was trained using a single common architecture for different games and

without any prior information about the rules of the game. This model surpassed the best chess and checkers programs and even improved AlphaGo's results (Vasiliev A. et al., 2020).

Reinforcement Learning in the context of games has become a medium for demonstrating the application of RL to complex strategic planning and learning problems.

Thus, AlphaGo and AlphaZero have shown that RL with Deep Learning can learn strategies that surpass human abilities in games. These technologies have also found practical application in decision optimization and robotics.

Advances in gaming have shown the potential of RL to solve a wider range of tasks, such as automatic parameter tuning, robot control, and autonomous systems.

Traditional board games such as Togyz qumalaq, Mankala and Kalah can be improved and diversified using Artificial Intelligence (Dalieva, A. et al., 2021).

Mancala is a traditional African game where players move and collect stones in holes on a game board. The goal is to collect more stones than the opponent.

The board game Kalah (Pekař L. et al., 2020) **is popular in different regions of Africa and on the Arabian peninsula. Players collect and move stones in an effort to score more points and capture the opponent's stones.**

Togyz qumalaq is a strategy game popular in Central Asia where players move stones between holes on the game board in an effort to grab more stones from the opponent (Dalieva, A. et al., 2021). The main goal of the game is to accumulate more stones.

Here is a general description of these games and how AI can make changes to them:

– AI can be used to develop optimal strategies in games. It can analyze the current state of the game and suggest the best moves for the players.

– AI can serve to train beginners in the game by providing recommendations and explaining the rules.

– AI can serve to create more intelligent virtual rivals. These bots can adapt their strategies to each player's style of play.

– AI can create a variety of game scenarios and complicate tasks for players, enriching the gaming experience.

– These games have their own unique rules and traditions in different cultures. AI can help players from different parts of the world understand and enjoy these games by providing multilingual translation and context.

It is possible to improve and enrich traditional board games, making them more interesting and accessible to a diverse audience using Artificial Intelligence.

Togyz qumalaq is a traditional board game widely popular among the peoples of Central Asia, including Kazakhstan. It is characterized by intricate rules, strategic decision-making, and a rich history, making it an excellent subject for research in the field of machine learning. This article examines how RL agents can be trained to play "Togyz qumalaq" and how this can contribute to the preservation and promotion of this culturally significant national game (Zhumanov A., et al., 2021).

Literature review. The research (Shakya A. **et al.,** 2023) is dedicated to the latest advancements in the field of reinforcement learning (RL) and contemporary gaming applications. By analyzing the literature on deep learning (DL) and reinforcement learning, as well as the extent to which scientific research is based on games such as ATARI, chess, and Go, the authors have established a unified framework and trends for the present and future of this industry (RL in gaming). Through their research, the authors conclude that deep RL constitutes approximately 25.1% of the DL literature, with a significant portion of this literature focused on RL applications in the gaming industry, paving the way for new and more complex algorithms capable of surpassing human performance.

In article (Wu K. et al., 2023), the application of the DQN algorithm for training a neural network to play the games Pong and Ms. Pac-Man is explored. The obtained data demonstrates a high level of performance of DQN in the simple Atari Pong game, but it encounters challenges when learning the more complex game Ms. Pac-Man. The authors attribute this lower performance to time and computational resource limitations. Additionally, the authors conclude that with sufficient training time and exploration, the model will eventually converge once the optimal combination of hyperparameters is determined.

In reference (Pekař L. et al., 2020), the history of the game Kalah is examined, along with research evaluating efficient tree search solutions. The authors propose the development of computationally simpler gaming algorithms and strategies. The article introduces an original heuristic algorithm based on the analysis of game rules, as well as standard and modified mini-max tree search algorithms. Experiments using a simple C++ application with the Qt framework revealed that the proposed heuristic algorithm performs comparably to an average-experience player and can outperform tree search algorithms with depths up to 2 nodes. The heuristic algorithm involves the use of self-learning strategies during gameplay to achieve better performance.

Article (Vasiliev A., et al., 2020) provides a review of the field of Reinforcement Learning (RL), including key algorithms and applications. It examines fundamental RL algorithms and Deep Reinforcement Learning (DRL) methods for solving complex problems with continuous actions and state spaces. Special attention is given to the applicability of RL in various interdisciplinary domains. The article also discusses model-based approaches and multi-agent RL. In conclusion, it emphasizes the prospects for future research in the field of RL.

In work (Buchanan B. G., 2021), the authors introduce a novel approach to playing the game of Go, utilizing deep neural networks to evaluate board positions and make moves. These networks are trained using a combination of supervised learning with expert data and reinforcement learning. Without prior search, they achieve the level of modern Monte Carlo tree search programs and introduce a new search algorithm that combines Monte Carlo simulations with value and policy networks. Using this algorithm, the AlphaGo program defeats other Go programs and secures a 5:0 victory over the European Go champion, marking the first time a computer program has defeated a professional human player in a full-sized game of Go, previously considered unattainable until a decade ago.

The work (Diddigi R., et al., 2022) presents the AlphaGo program, which demonstrated the level of artificial intelligence capable of competing and winning in one of the most complex strategic games - Go. AlphaGo was part of a research project that allowed DeepMind to conduct research in the areas of boosted learning and reinforcement learning, which has significant implications for the development of artificial intelligence and other fields. AlphaGo's success isn't just limited to games; Technologies developed for AlphaGo can be applied in various fields, including medicine, business and scientific research. AlphaGo has achieved many mile-stones in the development of artificial intelligence, including beating professional Go players and developing new learning methods. These advantages helped establish AlphaGo as a major advance in artificial intelligence and stimulate further research and development in the field.

The research (Shevtekar M., et al., 2022) presents the main advantages of the generalized MiniMax Q-learning algorithm for stochastic games, which can solve complex zero-sum games where determining optimal strategies can be challenging, including economic simulations and strategic scenarios. The considered algorithm is able to adapt to changing conditions and strategies of other players, which makes it useful in environments with dynamic variables, and can also be used for research and development of new methods in the field of artificial intelligence and reinforcement learning.

The article (Zhu, W., et al., 2019) discusses the MiniMax method for which the AlphaBeta

Pruning pruning method is used. AlphaBeta Pruning is an optimization technique for tree search algorithms such as game trees. The basic idea of this technique is that when certain conditions are detected, certain subtrees can be excluded from further analysis, resulting in a significant reduction in the computational load. In the case of MiniMax, this allows you to exclude nodes of the game tree where the probability of finding a more profitable move is low. This reduces the number of calculations and allows you to evaluate prospects in the game more efficiently.

The authors' use in article (Zhou, X., et al., 2017) of the DQN (Deep Q-Network) algorithm and its theoretical analysis can be used in the field of artificial intelligence and reinforcement learning in gaming applications and the development and improvement of artificial intelligence for games, including classic video games and board games, in robotics, algorithms are used to train robots to make decisions and control in various environments, to control systems and processes in the field of autonomous cars DQN can help develop control and decision-making systems. Deep learning algorithms, including DQN, are used to analyze medical data and diagnose various diseases.

In (Ms.S.Manju, et al., 2011) improved strategies for the classical Q-learning algorithm using various reward methods are considered. In the relative method, immediate rewards are selected. This allows the agent to respond more quickly to positive rewards, reducing the number of iterations before obtaining positive results. The continuous action method uses discounted rewards. This allows the long-term consequences of an agent's actions to be taken into account, helping the agent make more informed decisions. The Hyper-Q method uses mixed strategies with cumulative rewards. This allows the agent to adapt to different scenarios and choose the best actions in different situations. These advanced strategies enable agents to learn faster and more efficiently, reducing the number of iterations and improving the ability to adapt to a variety of scenarios.

The AIM and objectives of the study. The aim of the research is to develop and train a model using Reinforcement Learning methods that can play the traditional Kazakh game "Togyz qumalaq".

To achieve this aim, the following objectives are accomplished:

Model architecture selection: choose and configure a Reinforcement Learning model architecture capable of adapting to the complex rules and strategies of the game "Togyz qumalaq"

RL agent training: train an RL agent based on collected data with the goal of enabling it to play "Togyz qumalaq" competitively.

Results evaluation: evaluate the performance of the fitted agent by comparing its gameplay to that of a programmed algorithm and assessing the effectiveness of its strategy.

Materials and methods. In the development of any RL model, it is necessary to gather data in order to describe the state of the environment and identify key factors.

Game state representation: the game "Togyz qumalaq" consists of a game board with 18 cells, each initially containing 9 stones and a variable number of stones as the game progresses. Each player has 9 cells at their disposal. The game state is represented as an array of 18 elements, where each element denotes the number of stones in a cell. The objective of each player is to collect the maximum number of stones by the end of the game. However, if a player runs out of cells to make a move, the game is also considered in favor of the opponent. There can be a draw when players accumulate an equal number of stones at the end of the game. A player removes all stones from one of the pits on their side. If the selected pit contained more than one stone, the first removed stone is placed back into the same pit from which it was taken. All subsequent removed stones are distributed counterclockwise (from left to right in their own pits and from right to left in the opponent's pits). One stone is placed in each pit.

If the selected pit contained only one stone, it is placed in the next pit counterclockwise.

If, while distributing the stones, a player places the last stone into an opponent's pit, thereby

making the total number of stones in that pit even, they capture all the stones from that pit (including their own) and place them in their own storage pit.

If, during the stone distribution, a player places the last stone in an opponent's pit, bringing the total number of stones in that pit to three, and if the following three conditions are met, then all three of those stones are moved to the player's "qazan" (a special collecting pit), and the pit is transformed into a "tuzdyq" (a "sacred place" in Kazakh). The three conditions are as follows:

– if the player already has a "tuzdyq";

– the last pit on the opponent's side (9th hole) cannot be turned into a "tuzdyq";

– a "tuzdyq" cannot be created symmetrically to an opponent's already created "tuzdyq" (for example, if the third pit on the player's side is an opponent's "tuzdyq," the player cannot transform the third pit on the opponent's side into their own "tuzdyq").

A move that violates any of these three conditions can be made, but it does not result in the creation of a "tuzdyq" and the three stones remain in the pit. Any stone that falls into a "tuzdyq" during distribution is taken by the owner of the "tuzdyq" and placed in their "qazan". If a player cannot make a move because all of their pits are empty, the second player transfers all the remaining stones in their pits to their "qazan" and the game ends. The player with the most stones in their "qazan" wins.

At the time of conducting the research, the authors did not have access to any historical data regarding games with optimal moves. However, they found a solution by creating agents in the form of random move generation and by using the classic recursive MiniMax algorithm. The MiniMax algorithm is presented in the listing below:

Algorithm 1 MiniMax class implementation

```
class MiniMax:
    function get_move(board, depth)
    parameters:
        •   instance of class Board
        •   depth of recursion
    returns:
        •   move
    begin function
        best_move = null
        best_eval = -inf
        best_moves = []
        possible_moves = board.get_possible_moves()

        foreach move in possible_moves
            board.make_move(move)
            eval = MiniMax(board, depth, -inf, +inf, false)
            board.undo_move()

            if eval is greater than best_eval
                best_eval = eval
                best_moves = [move]
            else if abs(eval - best_eval) is less or equal 0.01
                best_moves.add(move)
```

```
            endif
        end for

        move = random move from best_moves
        return move
    end function

    function MiniMax(board, depth, alpha, beta, is_maximizing)
    parameters:
        • instance of class Board
        • depth of recursion
        • alpha, beta
        • is_maximizing (boolean)
    returns:
        • eval
    begin function
        eval = 0
        possible_moves = board.get_possible_moves()

        if depth is equal 0 or board.check_winner() is not null
            eval = board.evaluate()
            return eval
        end if

        if is_maximizing
            max_eval = -inf
            foreach move in possible_moves
                board.make_move(move)
                eval = MiniMax(board, depth - 1, alpha, beta, false)
                board.undo_move()
                max_eval = max(max_eval, eval)
                alpha = max(alpha, eval)

                if beta is less or equal alpha
                    break
                end if
            end for
            return max_eval
        else
            min_eval = +inf
            foreach move in possible_moves
                board.make_move(move)
                eval = MiniMax(board, depth - 1, alpha, beta, true)
                board.undo_move()
                min_eval = min(min_eval, eval)
                beta = min(beta, eval)
```

```
                if beta is less or equal alpha
                    break
                end if
            end for
            return min_eval
        end if
    end function
end class
```

The code for the MiniMax algorithm is interpreted as follows: the MiniMax class has a method `get_move()` which takes the current board state as an array and the depth of recursion as input. This method returns the optimal cell number from which the computer should make its move. The MiniMax(board, depth, alpha, beta, is_maximizing) function is the recursive core of the MiniMax algorithm. It calculates the evaluation score for the current board state. The `is_maximizing` flag indicates whether the computer is maximizing its score at the current recursion level.

As the first RL model, Q – learning was adopted. The Q – function formula (1) for the model looks as follows:

$$Q(s,a) = Q(s,a) + \alpha \times [R + \gamma \times \max\big(Q(s',a')\big) - Q(s,a)] \quad (1)$$

where Q(s, a) represents the Q – value for a state-action pair (s, a), α (alpha) is the learning rate, controlling how much the Q – values are updated in each learning step, R is the immediate reward received after taking action 'a' in state 's', γ is the discount factor, which balances the importance of immediate rewards versus future rewards, max(Q(s', a')) represents the maximum Q – value for the next state 's' among all possible actions 'a'.

The parameters of the Q – learning model were as follows:

Learning rate – 0.0001

Discount factor – 0.90

Epsilon – 0.5

The next implementation of the RL model was Deep Q Network (DQN). The DQN model was based on fully connected Dense layers with the following architecture:

The input layer has a size of 18 neurons (board), it is connected to the first Dense layer with 64 neurons (dense), after which the state of 2 players (tuzdyq) is supplied in parallel. These two parallel layers then pass through Dense layers of 64 and 32 neurons respectively, then concatenate layer into a layer of 96 neurons. In parallel there is a layer (move_owner) with 1 neuron, which serves to supply the model with information about the order of moves. This layer passes through Dense into 16 neurons and is concatenated into (concatenate_1). Next, 1 layer of Dense for 128, two layers for 64 neurons and one layer for 9 neurons are added on top. After this, it is necessary to take into account information about possible moves, which allows this to be done by a parallel input layer (possible_moves). The activation of the last layer occurs according to softmax, where the maximum element of the vector shows the most probable number of the cell from which a move must be made.

Adam was used as an optimizer with a learning rate of 1e-6, and the loss function was Categorical Crossentropy. The discount factor was taken as 0.95. Batch size was 32.

The structure of the DQN algorithm is shown in Figure 1.

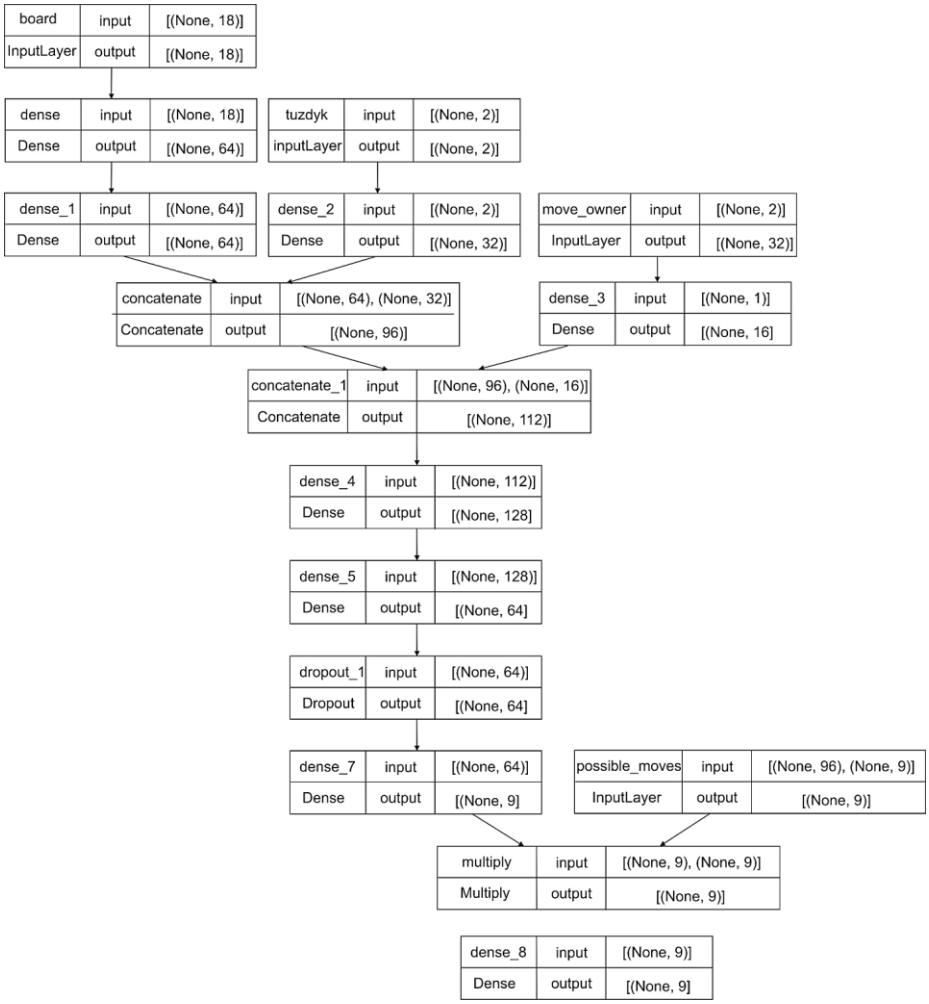

Figure 1. The structure of DQN model

Note – compiled by the author or compiled by the authors on the basis of (Nurtay M., 2022)

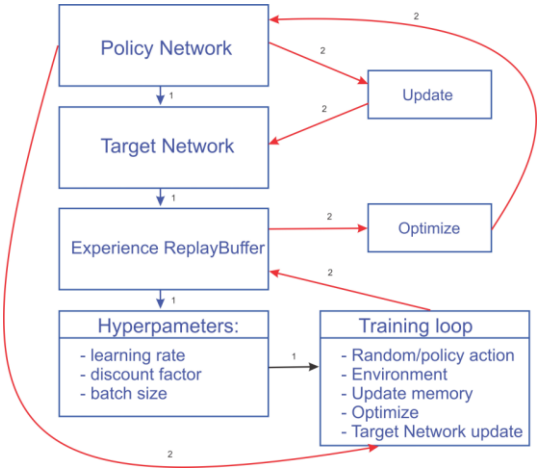

Figure 2. DQN algorithm's pipeline

Note – compiled by the author or compiled by the authors on the basis of (Nurtay M., 2022)

The Policy Network takes the state of the environment as input and predicts Q-values (expected future rewards) for all possible actions. Policy Network learns to predict optimal actions in different environmental states.

Target Network is a second neural network that also predicts Q-values, but it is used during the training process to stabilize the update of Q-values. During training, the Target Network is updated slower than the Policy Network and is used to calculate target Q-values.

The Experience Replay Buffer stores previous states, actions, rewards, and next states collected as the agent interacts with the environment. It is used to randomly sample learning experiences, which helps improve learning stability.

Training Loop is the core of the DQN algorithm, in which the agent interacts with the environment, collects experience and updates the Policy Network and Target Network. The training cycle includes sampling mini-batches from the experience buffer, calculating losses, and updating the network weights.

Hyperparameters are parameters that determine the agent's behavior and the learning process, such as the learning rate, discount factor, batch size and others. Hyperparameters must be tuned experimentally to achieve better results.

Results. This section presents the results of a study aimed at applying reinforcement learning methods to the development of an intellectual model for the game "Togyz Kumalak". The study results reflect the performance and adaptability of models developed using reinforcement learning and compare them with classical methods such as MiniMax. This section also provides an analysis of the key results, an assessment of the strengths and limitations of our approach, an examination of the key points, and a conclusion about their impact on the results.

A. Results of Q-learning performance. The Q-learning model was trained on 700 game episodes and showed the best result for a game with a MiniMax recursion depth of 1. For other options of recursion depth, the results were significantly lower. It should be noted that the algorithm was subject to restrictions on memory consumption; accordingly, it was necessary that the algorithm could learn as quickly as possible at lower costs for storing various states of the Q-table. However, even with all game variants, memory overflow was reached after the 700th episode of the game. The learning outcomes using the Q-learning model are shown in Figure 3.

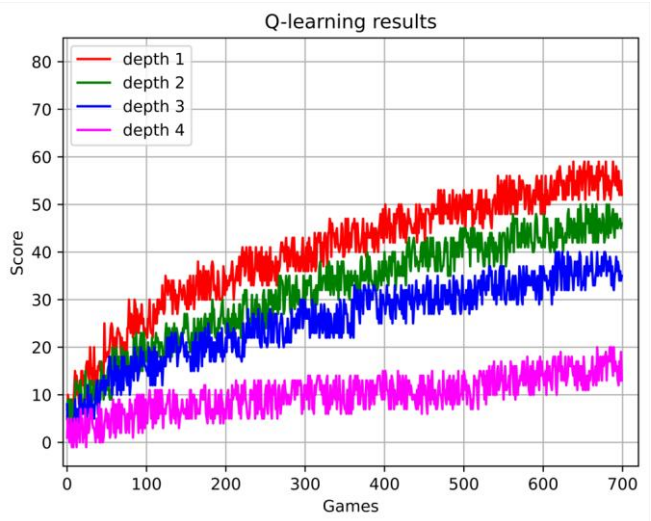

Figure 3. Q-learning score for MiniMax depths 2, 3, and 4

Note – compiled by the author personally (Akhmetov A.)

The graph above demonstrates that the 81 stone indicator is not achieved in any games, however, these are games that mostly ended either in a draw or in favor of the Q-learning model,

when MiniMax had no possible moves left. With each increase in the recursion depth of the Q-learning model, it became more difficult to win MiniMax, which is also illustrated by the results.

B. Results of DQN model performance. On the contrary, the DQN model showed better results and over a significantly larger number of games played. The result of DQN training is shown in Figure 4.

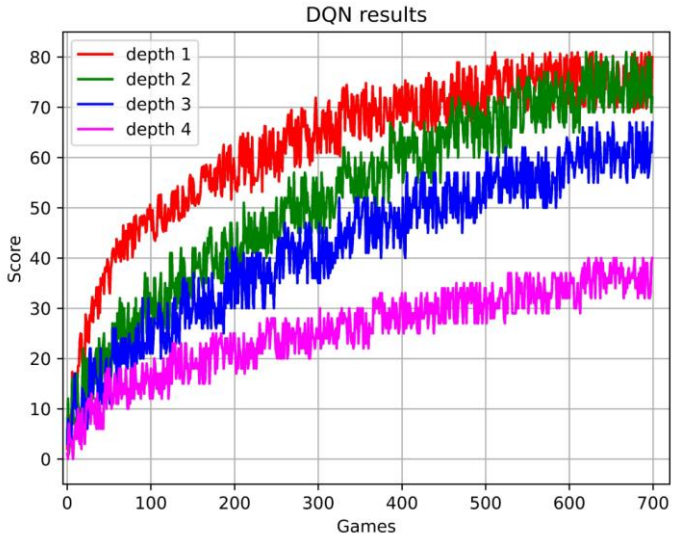

Figure 4. DQN score for MiniMax depths 1, 2, 3, and 4
Note – compiled by the author personally (Akhmetov A.)

Discussion of experimental results. In this section, we analyze the results of our experiments, particularly focusing on the "Games vs Score" graph for the Togyz qumalaq game at different depths of the MiniMax search (1, 2, 3, and 4) in comparison to the performance of DQN and Q-learning.

The "Games vs Score" graph reveals valuable insights into how the depth of the MiniMax search affects the performance of both DQN and Q-learning agents. It is evident that the performance of the agents can vary significantly depending on the depth of the opponent's search. Notably, DQN's performance shows a substantial improvement when the MiniMax depth is set to 2.

The significant improvement in DQN's performance at a MiniMax depth of 2 is a noteworthy finding. It suggests that DQN's ability to adapt and make informed decisions in response to the opponent's actions is particularly effective when facing an opponent with a moderately deep search strategy. This adaptability demonstrates the power of deep reinforcement learning in handling complex and evolving game situations.

A. Q-learning and Resource Requirements. On the other hand, Q-learning appears to face challenges, especially when confronted with opponents utilizing deeper MiniMax searches. The increased search depth introduces greater complexity, and Q-learning's resource-intensive nature becomes more apparent. The resource demands associated with Q-learning, particularly in memory and computation, may hinder its performance and scalability in scenarios where resource constraints are a concern.

B. Implications for Decision-Making in Game AI. The performance contrast between DQN and Q-learning highlights the importance of choosing the right algorithm for a specific context. DQN excels when adaptability and quick decision-making are crucial, making it an attractive

option in games with varying opponent strategies.

Additionally, our findings have implications for game developers and AI practitioners. When implementing game AI, the choice of algorithm should consider factors such as computational resources, adaptability, and the expected depth of the opponent's search.

Conclusions. In this study, we explored the performance of Q-learning and Deep Q-Network (DQN) as main of the Reinforcement Learning algorithms in the Togyz qumalaq game. We also implemented a MiniMax opponent to assess the adaptability and resource efficiency of both algorithms.

The standout finding of this research is the remarkable performance of DQN at a MiniMax search depth of 2. DQN demonstrated a high level of adaptability and competitiveness when faced with opponents employing moderately deep search strategies. This underscores the effectiveness of Deep Reinforcement Learning in handling complex and dynamic game environments.

On the other hand, our results indicate that Q-learning encounters resource-related challenges, especially when confronted with opponents utilizing deeper MiniMax searches. The algorithm's resource-intensive nature poses concerns regarding its scalability in resource-constrained settings.

Research findings underscore the importance of selecting the appropriate algorithm for a specific gaming context. DQN shines in scenarios where adaptability and swift decision-making are vital, whereas Q-learning may require significant computational and memory resources to maintain competitiveness.

The results of this study have several implications for the fields of Reinforcement Learning and game AI. They serve as a call to further optimize and fine-tune Deep Reinforcement Learning techniques, enhancing their suitability for complex games like Togyz qumalaq.

In conclusion, the performance of DQN at MiniMax depth 2 highlights the potential of Deep Reinforcement Learning in addressing the complexities of game AI. The choice of algorithm should be driven by the specific requirements of the game, ensuring that the game AI performs optimally while respecting constraints.

Conflict of interest. The authors declare that there is no conflict of interest.

Notification of the use of generative AI and technologies using it in the process of writing a manuscript. In preparing this work, the authors did not use generative AI and technologies based on its use in the process of writing the manuscript.

References

Dalieva, A., Matylov, S., Akshurev, A. (2021) Togyzkumalak: history, rules and strategies of the game. – Almaty: [ed. not specified]. – 200 p.

Zhumanov A., Mukhamedyarova N. (2021) Intellectual games as a tool for developing cognitive skills in educational practice // Education and Science. – 2021. – Vol. 23, No. 5. – Pp. 124-132.

Turgumbaev T., Aliev B. (2019) Modern approaches to the use of artificial intelligence in solving intellectual games // Bulletin of KazNU. The series is physical and mathematical. – Vol. 62. – No. 2. – Pp. 78-89.

Vasiliev A. N., Ilyin P. A.(2020) Artificial intelligence in education: application prospects and ethical aspects // Bulletin of the Digital Society. – Vol. 12. – No. 3. – Pp. 45-53.

Nurtay M. (2022). Solving the problem of detecting phishing websites using ensemble-learning models. Scientific Journal of Astana IT University, 12(12), 55–64

Smith J. A., Green M. (2020) Scientific intelligence and traditional games: possibilities and conclusions // Journal of Artificial Intelligence Research. – Vol. 7. – No. 2. – Pp. 45-60.

Gavrilov S., Nikitin P. (2021) Artificial intelligence in the management of complex systems: prospects and challenges // Management and Informatics. – Vol. 58. – No. 4. – Pp. 32-40.

Mkondiwa, Maxwell. (2020). Mancala board games and origins of entrepreneurship in Africa. PLoS ONE. 15. 10.1371/journal.pone.0240790.

Pekař, Libor, Radek Matušů, Jiří Andrla, and Martina Litschmannová (2020) Review of Kalah Game Research and the Proposition of a Novel Heuristic–Deterministic Algorithm Compared to Tree-Search Solutions and Human Decision-Making Informatics 7, no. 3: 34. https://doi.org/10.3390/informatics7030034

Souchleris, Konstantinos & Sidiropoulos, George & Papakostas, George. (2023). Reinforcement Learning in Game Industry—Review, Prospects and Challenges. Applied Sciences. 13. 2443. 10.3390/app13042443.

Wu K., Xu N. (2023). Improving the Performance of Deep Q-learning in Games Pong and Ms. Pacman. Highlights in Science, Engineering and Technology. 39. 1127-1130. 10.54097/hset.v39i.6718.

Vasiliev A., Shchekotova M. (2020) Role of AI in Strategy Games: The Case of AlphaGo. Journal of Computational Intelligence. – Vol. 22. – No. 2. – Pp. 102-115.

Pekař L., Matušů R., Andrla J., Litschmannova M. (2020). Review of Kalah Game Research and the Proposition of a Novel Heuristic–Deterministic Algorithm Compared to Tree-Search Solutions and Human Decision-Making. Informatics. 7. 34. 10.3390/informatics7030034.

Shakya, A., Pillai G., Chakrabarty S. (2023). Reinforcement Learning Algorithms: A brief survey. Expert Systems with Applications. 231. 120495. 10.1016/j.eswa.2023.120495.

Buchanan, B. G. (2021) Artificial intelligence and human thinking: Lessons from traditional games. — Chicago: University Press – 223 p.

Diddigi R., Kamanchi C., Bhatnagar S. (2022). A Generalized MiniMax Q-Learning Algorithm for Two-Player Zero-Sum Stochastic Games. IEEE Transactions on Automatic Control. 67. 1-1. 10.1109/TAC.2022.3159453.

Shevtekar M., Bhaila M. (2022). Analysis of Game Tree Search Algorithms Using MiniMax Algorithm and Alpha-Beta Pruning. International Journal of Scientific Research in Computer Science, Engineering and Information Technology. 328-333. 10.32628/CSEIT1228644.

Zhu, W., Zeng, D. and Song, R. (2019). Proper inferences for value function in high-dimensional Q-learning for dynamic treatment regimes. Journal of the American Statistical Association, 1404-1417.

Zhou, X., Mayer-Hamblett, N., Khan, U. and Kosorok, M. R. (2017). Residual weighted learning for estimating individualized treatment rules. Journal of the American Statistical Association,169–187.

Ms.S.Manju, & Dr.Ms.M.Punithavalli. (2011). An Analysis of Q-Learning Algorithms with Strategies of Reward Function. International Journal on Computer Science and Engineering. 3.

Information about authors

M. Nurtay – Abylkas Saginov Karagandy Technical University, Karagandy, Kazakhstan, *solano.lifan2@bk.ru,* 87024270103

A. Akhmetov – student gr.VT-22-1, Abylkas Saginov Karagandy Technical University, Karagandy, Kazakhstan, *furybam@gmail.com,* 87081236834

A. Tau – Abylkas Saginov Karagandy Technical University, Karagandy, Kazakhstan, *ardak.tau@mail.ru,* 87056883413

G. Alina – Abylkas Saginov Karagandy Technical University, Karagandy, Kazakhstan, *alinagaukhar@gmail.com,* 87786725725

N. Mutovina – Abylkas Saginov Karagandy Technical University, Karagandy, Kazakhstan, *mutovina_natalya@mail.ru,* 87017568538