

ПОЄДНАННЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ ТА АЛГОРИТМУ МОНТЕ-КАРЛО ДЛЯ ПОВБУДОВИ ШАХОВОГО РУШІЯ

І. Козій, Р. Яремко

*Львівський національний університет імені Івана Франка,
вул. Університетська 1, Львів, 79007, Україна,
e-mail: iryna.kozyi@lnu.edu.ua, roman.yaremko.pmi@lnu.edu.ua*

Розглянуто підхід до створення шахового рушія на основі поєднання штучних нейронних мереж та алгоритму пошуку Монте-Карло. Актуальність дослідження зумовлена необхідністю підвищення ефективності аналізу шахових позицій у задачах із великим простором можливих ходів, де традиційні методи повного перебору варіантів потребують значних обчислювальних ресурсів. Запропонований підхід передбачає використання нейронної мережі для оцінювання шахових позицій і формування апіорних ймовірностей можливих ходів, які надалі застосовуються для спрямування процесу пошуку оптимального рішення в алгоритмі Монте-Карло. Для навчання моделі сформовано набір даних із позицій шахових партій гротмейстерів і тактичних задач, отриманих з відкритих шахових баз. Досліджено вплив найважливіших гіперпараметрів нейромережі та складу навчального набору даних на оцінку шахових позицій та силу гри розробленої системи в умовах обмежених обчислювальних ресурсів. Описано архітектуру системи, процес підготовки навчальних даних та реалізацію алгоритмів. Проведено тестування розробленого програмного забезпечення та проаналізовано ефективність запропонованого підходу.

Ключові слова: шаховий рушій, дерево пошуку Монте-Карло, нейронна мережа, штучний інтелект.

1. ВСТУП

Створення програм, здатних ефективно аналізувати шахові позиції й обирати оптимальні ходи, залишається актуальним завданням упродовж багатьох десятиліть [1]. Останніми роками значного розвитку набули підходи, що використовують методи машинного навчання. Зокрема, використання нейронних мереж дає змогу автоматично формувати функцію оцінювання позицій на основі навчальних даних. Поєднання нейронних мереж з алгоритмом Монте-Карло стало важливим підходом у створенні сучасних систем штучного інтелекту для ігор із великим простором можливих ходів, допомагаючи ефективно оцінювати позиції та планувати стратегічні дії [2].

2. МЕТОДОЛОГІЯ ДОСЛІДЖЕННЯ

Основою запропонованого шахового рушія є поєднання нейронної мережі з алгоритмом Монте-Карло [2].

Нейронну мережу використовують для оцінювання поточної шахової позиції та прогнозування ймовірності можливих ходів. Алгоритм Монте-Карло використовує ці оцінки для ефективнішого пошуку оптимального ходу.

Реалізовано згорткову нейромережу з двома виходами: перший формує апіорний розподіл ймовірностей найліпшого ходу серед усіх можливих ходів, а другий – прогнозує ймовірний результат завершення партії з поточної позиції.

Архітектура нейромережі складається з трьох основних частин: початкового шару, послідовності залишкових блоків і кінцевої частини, яка формує два результати. Початковий шар є згортковим і забезпечує розширене подання позиції шляхом збільшення кількості каналів до 64. Таке подання дає змогу знаходити локальні шаблони, наприклад, “кінь атакує центр”, “пішак захищає іншого пішака” тощо. Після початкового шару йде послідовність шести залишкових блоків, у якій вихід одного залишкового блока є входом наступного. У межах послідовності цих блоків два згорткові шари з нормалізацією та функцією активації обробляють дані. У цю структуру також інтегрований блок, який дає змогу моделі підсилювати інформативні канали та пригнічувати менш значущі. Після цього результат додається до вхідного сигналу. Використання саме залишкових блоків дає змогу будувати глибокі моделі без втрати початкової інформації: сигнал із першого шару не втрачається і впливає на результати останніх шарів [3].

Після проходження всіх залишкових блоків вихід передається до двох окремих гілок: одна гілка відповідає за розподіл ймовірностей всіх ходів, а інша – за прогнозований результат з вхідної позиції.

Для подання шахової позиції використовується числове зображення у вигляді масиву бітових дошок, що містить інформацію про розташування фігур на шаховій дошці.

Алгоритм Монте-Карло виконує пошук оптимального ходу шляхом багаторазового моделювання можливих варіантів розвитку гри. Такий підхід став основою для програм, що розв’язують складні задачі вибору дій у настільних іграх, де повний перебір варіантів є неможливим або потребує значних часових та обчислювальних ресурсів [4].

Кожна вершина дерева відповідає певній шаховій позиції, а ребра – можливим ходам. У процесі роботи алгоритм поступово розширює дерево та накопичує статистику результатів симуляцій. З кожною ітерацією алгоритм поліпшує оцінку найліпшого ходу. Цей підхід дає змогу балансувати між дослідженням нових ходів та використанням вже досліджених ходів.

Проте, якщо, згідно з класичним підходом, покладатися тільки на статистику результатів симуляцій, то це призводить до повільної збіжності алгоритму в задачах із великим простором можливих дій, оскільки ефективність пошуку значною мірою залежить від кількості виконаних симуляцій.

Для вирішення цієї проблеми запропоновано модифікацію алгоритму, що дає змогу інтегрувати апріорні знання про перспективність можливих дій, отриманих за допомогою нейронної мережі, у процес побудови дерева пошуку. Основна ідея цього підходу полягає у використанні ймовірнісного прогнозу для кожної дії, що отримується за допомогою розглянутої вище нейронної мережі [2].

У рамках запропонованого підходу значення функції вибору визначається за формулою

$$PUCT(s, a) = \frac{w(s, a)}{n(s, a)} + C \cdot P(s, a) \cdot \frac{\sqrt{N(s)}}{1 + n(s, a)}, \quad (1)$$

де s – батьківський вузол дерева пошуку; a – дія, що розглядається; $w(s, a)$ – накопичений виграш після виконання дії a ; $n(s, a)$ – кількість відвідувань вузла, що відповідає дії a ; $N(s)$ – кількість відвідувань батьківського вузла s ; $P(s, a)$ – апріорна ймовірність вибору дії a у вузлі s ; C – константа, що визначає баланс між дослідженням нових дій і використанням уже досліджених.

У такому підході апріорні ймовірності $P(s, a)$ та оцінка ймовірного результату партії отримуються за допомогою нейронної мережі. Значення $P(s, a)$ використовується у формулі (1) для спрямування пошуку до найбільш перспективних дій, тоді як оцінка результату партії застосовується для оновлення статистики вузлів дерева замість виконання випадкових симуляцій. Такий підхід дає змогу значно підвищити ефективність пошуку та скоротити кількість необхідних симуляцій.

3. МОДЕЛЬ ДАНИХ

Опишемо подання вхідних і вихідних даних моделі детальніше.

3.1. ФОРМАТ ВХІДНИХ ДАНИХ

Розглянемо бітову дошку як 64-бітове число, в якому кожен біт відповідає одній клітинці на шахівниці. Значення біта засвідчує наявність чи неаявність фігури на відповідній клітинці.

Для повного опису позиції треба принаймні 12 бітових дошок – по одній для кожного типу фігури (пішак, кінь, слон, тура, ферзь, король) окремо для білого та чорного кольорів [5]. Приклад бітової дошки для білих коней на початку гри зображено на рис. 1.

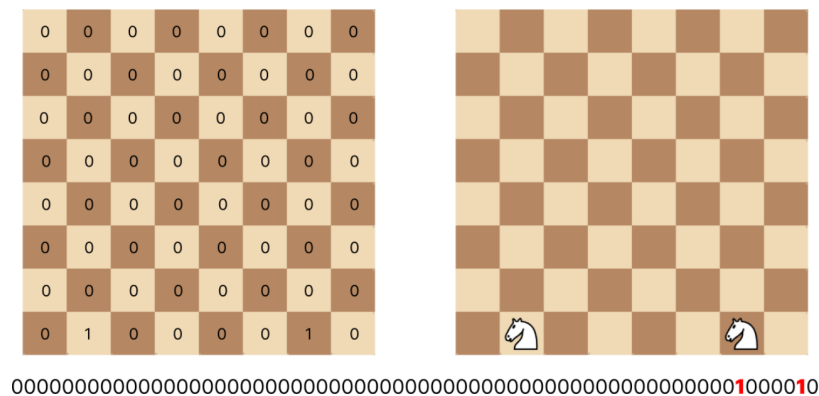


Рис. 1. Приклад бітової дошки для білих коней на початку гри

Крім розташування фігур, для повного опису позиції потрібна також додаткова інформація, а саме: право на рокірування, можливість взяття на проході, номер ходу, кількість повторень позиції та сторона, яка має право ходу в цій позиції. Вибраний формат вхідних даних охоплює 21 бітову дошку:

- 12 бітових дошок для фігур;
- 2 бітові дошки, які відповідають за повторення позиції;
- 4 бітові дошки, які відповідають за можливість рокірування короля;
- 1 бітова дошка, яка відповідає за можливість взяття на проході;
- 1 бітова дошка для номера ходу;
- 1 бітова дошка для лічильника ходів без побиття фігур та без просування пішаків (правило 50 ходів).

Тому на вхід нейромережі подається тривимірний масив розміром $21 \times 8 \times 8$.

3.2. ФОРМАТ ВИХІДНИХ ДАНИХ

Після проходження всіх залишкових блоків нейромережа розгалужується на дві гілки: одна обчислює розподіл ймовірностей найліпшого ходу для усіх можливих ходів з цієї позиції, інша – скалярне значення прогнозованого результату.

Тут розподіл ймовірностей – це масив довжиною 4864, у якому всі можливі ходи на шахівниці закодовано у фіксованому порядку, що задається перед початком створення набору даних. Кожен індекс цього масиву відповідає за певний тип ходу. Усі ходи поділяються на три групи [5].

- Ковзні ходи. Для кожної клітинки у восьми напрямках можливі ходи від однієї до семи клітинок. Сумарна кількість таких ходів $64 \times 8 \times 7$.
- Ходи коня. Окремо кодуються всі можливі стрибки коня. Сумарна кількість 64×8 .
- Перетворення пішака на фігуру. Окремо виділено ходи для перетворення пішака на іншу фігуру (ферзь, тура, слон і кінь). Перетворення можливе простим ходом вперед і взяттям іншої фігури вліво чи вправо. Сумарна кількість таких ходів 64×12 .

Хоча багато з цих ходів є неможливими в реальних позиціях, такий спосіб кодування зручний для програмування і структурований для навчання мережі. Нейромережа досить швидко навчається розпізнавати межі дошки та надавати високі ймовірності тільки ходам в межах дошки.

Скалярне значення прогнозу – це дійсне число в межах $[-1; 1]$, де -1 означає повну впевненість у програві, а 1 – повну впевненість у перемозі. Значення між цими межами відображає оцінку позиції для сторони, яка має право ходу.

3.3. ФОРМУВАННЯ НАБОРУ ДАНИХ ДЛЯ НАВЧАННЯ

Для формування набору даних взято дві різні вибірки. Перша – це позиції з офіційних партій (з сайту PGN Mentor [6]) найсильніших гравців в шахи останнього десятиліття. Друга вибірка – це тактичні позиції, які отримано з відкритої бази даних платформи Lichess [7].

Для формування міток використано популярний шаховий рушій Stockfish. За допомогою Stockfish для кожної позиції отримано п'ять найліпших ходів та оцінки позиції в пішаках. Ці оцінки за допомогою нормованої експоненційної функції перетворювалися на ймовірнісні мітки у вигляді вектора дійсних чисел між 0 та 1, що в сумі дають 1. Таким алгоритмом проаналізовано 1.5 мільйона позицій з партій гротмейстерів і 1.5 мільйона шахових задач.

Для збільшення обсягу даних для навчання нейромережі застосовано техніку розширення даних за допомогою горизонтального дзеркалення.

4. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Розроблений шаховий рушій реалізовано у вигляді програмної системи, що складається з кількох функціональних модулів:

- модуль зображення шахової дошки;
- модуль генерації можливих ходів;
- модуль нейронної мережі;

- модуль алгоритму Монте-Карло;
- інтерфейс взаємодії з користувачем.

Мета експериментального дослідження – оцінювання впливу гіперпараметрів нейромережі та характеристик навчального набору даних на точність моделі, що використовується у складі алгоритму Монте-Карло. У межах дослідження було навчено та протестовано понад двадцять моделей із різними конфігураціями параметрів і різними наборами навчальних даних.

Для оцінювання якості роботи моделі застосовували такі метрики: точність top-1 (частка випадків, коли правильний хід відповідає найбільш імовірному передбаченню моделі), точність top-5 (частка випадків, коли правильний хід входить до п'яти найбільш імовірних варіантів), а також MAE (середня абсолютна похибка прогнозування результату партії).

Першим етапом дослідження було вивчення впливу співвідношення між складовими функції втрат, що відповідають за прогноз розподілу ймовірностей ходів та оцінку результату партії. Експеримент проводили на наборі з 2 240 000 шахових позицій, серед яких 380 000 тактичних задач і 1 860 000 позицій із партій гросмейстерів. Результати експерименту наведено у табл. 1.

Таблиця 1

Співвідношення між втратами ймовірностей і прогноз

Співвідношення	MAE	Точність top-1	Точність top-5
0.5 : 0.5	0.477	0.306	0.647
0.6 : 0.4	0.470	0.306	0.654
0.7 : 0.3	0.440	0.310	0.663
0.8 : 0.2	0.491	0.309	0.663

Експериментальні результати підтверджують, що найліпші показники досягаються за співвідношення 0.7:0.3, яке забезпечує підвищення точності визначення найліпшого ходу на 1.3 % та зменшення середньої абсолютної похибки на 7.6 % порівняно з базовим варіантом.

Другий етап дослідження – аналіз впливу складу навчального набору даних на точність моделі. Для кожного експерименту використовували набір обсягом 4 000 000 позицій, у якому змінювалося співвідношення тактичних задач і позицій із партій гросмейстерів. Отримані результати наведено в табл. 2.

З отриманих результатів видно, що найбільшу точність демонструє модель, навчена на наборі з 75 % тактичних позицій. У цьому випадку точність визначення найліпшого ходу зростає на 18.8 %, а значення MAE зменшується на 17.4 %, порівняно з рівномірним розподілом даних. Водночас під час практичного застосування така модель демонструвала менш стабільну ігрову поведінку, оскільки часто намагалася знаходити тактичні мотиви у позиціях, де їх фактично немає.

Окремо було досліджено вплив використання блоків стиснення та активації (SE-блоків) у структурі нейромережі. Результати наведено в табл. 3.

Використання SE-блоків дає змогу підвищити точність визначення найліпшого ходу на 6.6 %, однак супроводжується незначним збільшенням середньої абсолютної похибки.

Таблиця 2

Вплив набору даних на точність моделі

Позиції з партій гросмейстерів, %	Позиції тактичного характеру, %	MAE	Точність top-1	Точність top-5
25 %	75 %	0.247	0.689	0.869
35 %	65 %	0.307	0.589	0.815
50 %	50 %	0.299	0.580	0.808
65 %	35 %	0.311	0.526	0.772
75 %	25 %	0.348	0.433	0.742

Таблиця 3

Вплив SE-блоків на точність моделі

SE-блоки	MAE	Точність top-1	Точність top-5
так	0.299	0.580	0.808
ні	0.291	0.544	0.801

За результатами проведених експериментів було визначено конфігурацію моделі, що забезпечує найліпший баланс між точністю прогнозування та стабільністю гри. Параметри цієї моделі наведено в табл. 4.

Таблиця 4

Параметри найліпшої моделі

Кількість фільтрів	64
Кількість залишкових блоків	6
Розмір пакетів вхідних даних в мережі	128
Коефіцієнт навчання	0.001
Наявність SE-блоків	Так
Співвідношення між втратами ймовірностей та прогнозу	0.7:0.3
Коефіцієнт dropout	0
Коефіцієнт L2 регуляризації	0.0005
Кількість позицій з партій гросмейстерів	2 600 000
Кількість тактичних позицій	1 400 000
Кількість епох навчання	5
Втрати на розподілі ймовірностей ходів	1.821
MAE	0.311
Точність top-1	0.526
Точність top-5	0.772

Отримана модель демонструє силу гри, що приблизно відповідає четвертому шаховому розряду.

5. ВИСНОВОК

Отримані результати тестування розробленого шахового рушія на наборі шахових позицій демонструють ефективність використання нейромережових моделей у поєднанні зі статистичними алгоритмами пошуку для розв’язання складних ігрових задач.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Shannon C.E. Programming a Computer for Playing Chess / C.E. Shannon // Philosophical Magazine. – 1950. – Vol. 41. – P. 256-275. – URL: <https://vision.unipv.it/IA1/ProgrammingaComputerforPlayingChess.pdf>.
2. Kemmerling M. Beyond Games: A Systematic Review of Neural Monte Carlo Tree Search Applications / M. Kemmerling, D. Lütticke, R.H. Schmitt // arXiv:2303.08060v1 [cs.AI]. – 2023. – URL: <https://arxiv.org/pdf/2303.08060>.
3. Hu J. Squeeze-and-Excitation Networks / J. Hu, L. Shen, S. Albanie, G. Sun, E. Wu // arXiv:1709.01507v4 [cs.CV]. – 2019. – URL: <https://arxiv.org/pdf/1709.01507>.
4. Cazenave T. Monte Carlo Search Algorithms Discovering Monte Carlo Tree Search Exploration Terms / T. Cazenave // arXiv:2404.09304v1 [cs.AI]. – 2024. – URL: <https://doi.org/10.48550/arXiv.2404.09304>.
5. Яремко Р. Оптимізація дерева пошуку Монте-Карло у шахах за допомогою policy-value нейромережі / Р. Яремко // Міжнародна студентська наукова конференція з питань прикладної математики та комп’ютерних наук (МСНКПМКН-2025). – 8-9 травня 2025 р. – Львів: 2025. – С. 208–212.
6. PGN Mentor [Електронний ресурс]. – Accessed: 12.03.2025. – Режим доступу: <https://www.pgnmentor.com/files.html>.
7. Lichess Database [Електронний ресурс]. – Accessed: 12.03.2025. – Режим доступу: URL: <https://database.lichess.org/#puzzles>.

Стаття: надійшла до редколегії 04.02.2026

доопрацьована 25.02.2026

прийнята до друку 04.03.2026

COMBINING ARTIFICIAL NEURAL NETWORKS AND MONTE CARLO ALGORITHM TO BUILD A CHESS ENGINE

I. Koziy, R. Yaremko

Ivan Franko National University of Lviv,

1, Universytetska str., 79007, Lviv, Ukraine,

e-mail: iryna.koziy@lnu.edu.ua, roman.yaremko.pmi@lnu.edu.ua

The paper considers an approach to developing a chess engine based on the integration of artificial neural networks and the Monte Carlo search algorithm. The relevance of the study is determined by the need to improve the efficiency of chess position analysis in problems with a large space of possible moves, where traditional exhaustive search methods require significant computational resources. The proposed approach involves the use of a neural network to evaluate chess positions and generate prior probabilities of possible moves, which are subsequently used to guide the search process toward the optimal decision within the Monte Carlo algorithm.

To train the model, a dataset was constructed from positions taken from grandmaster games and tactical problems obtained from open chess databases. The influence of key

neural network hyperparameters and the composition of the training dataset on the evaluation of chess positions and the playing strength of the developed system under limited computational resources was investigated. The architecture of the system, the process of preparing the training data, and the implementation of the algorithms are described. The developed software was tested, and the effectiveness of the proposed approach was analysed.

Key words: chess engine, Monte Carlo tree search, neural network, artificial intelligence.