# Neural Chess: Structured Rewards in Reinforcement Learning for Chess AI

**Pala Tejaswi, Poonawala Mustafa Jabir, Ekaterina Galkina**

## Abstract

Chess has long served as an ideal domain for testing artificial intelligence systems due to its demand for strategic planning and decision-making. Our work explores structured reward mechanisms in reinforcement learning to enhance learning efficiency in chess agents, focusing on center control, king safety, and material advantage.

## 1 Problem and Motivation

Chess has long been regarded as a proving ground for Artificial Intelligence. The essence of strategic planning and rational decision-making in strong chess play makes the game an ideal domain in which to test systems aiming to master these human-like cognitive capabilities. Traditional Reinforced Learning (RL) chess engines employ generic reward functions that rely only on the outcome of each episode/game. They generally assign a reward of +1 for win, -1 for a loss, and 0 or +0.5 for a draw, providing little to no feedback for individual moves made during the game. However, this approach suffers from a fundamental flaw as it overlooks the pivotal moves that are made that ultimately decide the outcome of the game. It is this lack of understanding which inhibits the agent from learning which action affects the result positively or negatively. Consequently, the learning process becomes inefficient, resulting in a delayed model convergence and requirement of more games to deduce pattern and strategy leading to success. Hence, our project tries to study how reinforcement learning algorithms can be optimized through adaptive reward mechanisms, enhancing chess move evaluation while minimizing computational cost and resource usage?

## 2 Related Work

The study of move evaluation in the domain of chess originally became popular in the 19th century, but with the advent of AI it has made some unprecedented breakthroughs. The earliest advancement came in 2016, with AlphaGo, a neural network that had achieved superhuman performance in the game of Go. This laid the foundations of AlphaZero, which introduced a novel combination of the Monte Carlo Tree Search (MCTS) and neural network base value predictions. This approach was recreated by the open-source community developed Leela Chess Zero (Lc0), an independent attempt to replicate the similar approach with several subsequent enhancements. However, convolution-based models may be poorly suited for chess as chess has long-range interactions feature compared to Go, and convolutions are poorly equipped to deal with these because of their small receptive fields. More recently, NNUE (Efficiently Updatable Neural Network) has been introduced and integrated into chess engines like Stockfish, further advancing the field with lightweight yet powerful evaluation techniques.

In parallel, several smaller-scale projects have explored reinforcement learning in chess. One such project, Simple Multi Agent Deep Reinforcement Learning (MADRL) Chess, which serves as the foundation for our own work. MADRL Chess implements a basic self-play framework where two agents, trained using Proximal Policy Optimization (PPO), compete against each other to learn optimal strategies. However, like many others, it relies on a basic reward structure: +100 for a win,

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

-100 for a loss, 0 for a draw, and a minor penalty per move to encourage shorter games. Building upon this framework, our work introduces a more structured reward mechanism, enhanced board state representations, and improved training stability, aiming to overcome the limitations of sparse rewards and accelerate convergence toward stronger, more strategic play.

# 3 Hypothesis and Solution

While existing reinforcing learning chess agents like MADRL Chess do demonstrate a viable approach for training a Chess Ai, they still rely on sparse and outcome based rewards structures that provide limited guidance during the gameplay. This eventually creates a bottleneck in the learning curve of these agents as they struggle to correlate patterns and strategies with intermediate decisions that would lead to long term success. Therefore, we are hypothesizing that by incorporating a structured, multi-component and dynamic reward system that would align with the fundamental chess principles such as center control, king safety and offensive pressure, and material advantage will enable agents to get more granular feedback and promote development of richer strategies. In addition to this, the model will be still receiving traditional rewards of win, loss and draw.

## 3.1 Center Control

The first reward mechanism implemented focuses on center control, namely on the occupation of the center squares. Controlling the center is a fundamental chess strategy as it allows for greater piece mobility, faster transition between attack and defense and a strong control over the board. Centralized pieces can influence more squares and are able to restrict the opponent's development. Thus, overall, proper center occupation limits the opponent and allows us to be dominant and take control of the play. To capture this strategic value in our agent's learning process, we implement a center control reward mechanism. The main central squares (d4, d5, e4, and e5) are rewarded with 3 points per occupying piece while the secondary center squares (c3, d3, e3, f3, c6, d6, e6, f6, c4, c5, f4, and f5) bring 2 points per piece. We reward all pieces equally without considering their nature.

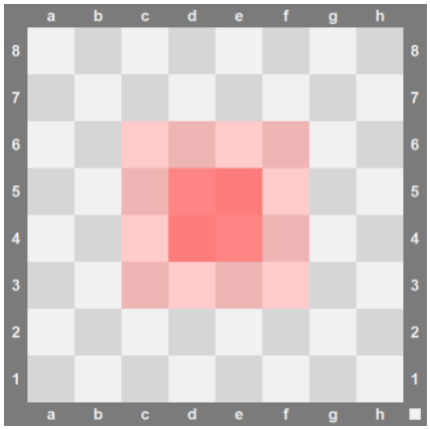


Figure 1: Centre Control Heatmap

### 3.1.1 Maximum reward possible for this category

Theoretically, the maximum rewards for the center control can reach 36 points. This is calculated based on 4 main center squares (d4, e4, d5, e5), each valued at 3 points, and 12 secondary center squares, each valued at 2 points. If all the 16 squares are occupied by a single agent, then potential rewards would be:

- Main center: $4 \times 3 = 12$ points
- Secondary center: $12 \times 2 = 24$ points
- Total maximum: $12 + 24 = 36$ points

However, in practice, it is almost impossible to have all 16 squares simultaneously occupied by our own pieces. Chess pieces require mobility, many center squares are contested by the opponent and some pieces may be already lost. Realistically, a strong center control would typically involve occupying around 6 - 8 squares. For example, it can be done by occupying 2 main center squares (2 × 3 = 6 points) and 4 secondary center squares (4 × 2 = 8 points), which gives a total of 14 points. This is an appropriate maximum reward if we compare it to the check or checkmate rewards, which are respectively 10 and 100 points. Because on average this will not exceed 10 points, ensuring that while central control is motivated, the agent's primary agenda is still to attack the opponent's king.

#### 3.1.2 Simple Example

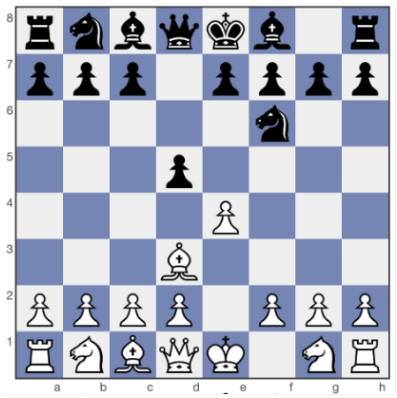

Figure 2: Center Control Reward Example

- White:
  - A pawn on e4 (main center, +3)
  - A bishop on d3 (secondary center, +2)
  - Total: 5 points
- Black:
  - A knight on f6 (secondary center, +2)
  - A pawn on d5 (main center, +3)
  - Total: 5 points

### 3.2 King Safety / Attack

The second mechanism implemented focuses on the king's safety and the attack on the opponent's king. Maintaining king safety is essential for survival, while adding pressure to the enemy king often leads to a decisive strategy and material gains. When the king is under direct attack, it forces the opponent to make defense plays and limits the king's space to escape and can lead to weakened coordination among defending pieces. Likewise, exposing our own king can result in a counterattack and significant loss. Hence, rewarding the agent for threatening the enemy's king and penalizing for compromising its own king's safety will lead to a more balanced and effective gameplay.

To quantify this, we examine all 8 squares surrounding each king and evaluate how many are under attack. Unlike the central control system, in this mechanism we are factoring the type of attacking piece. Specifically:

- Attacks by kings or pawns incur a ±0.2 value
- Knights: ±0.3
- Bishops and rooks: ±0.5
- Queens: ±1.0

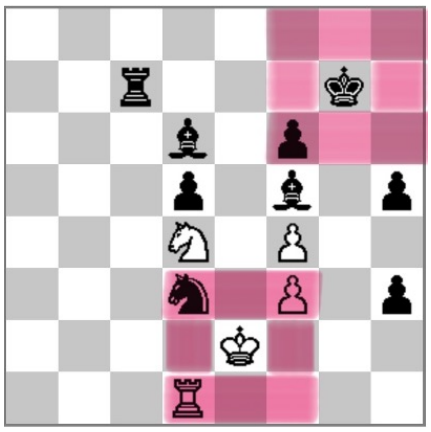

Figure 3: King Safety Evaluation

If our king is under threat, the total value is subtracted as a penalty. If our pieces threaten the opponent's king, the same logic applies in reverse as a reward. The score is calculated per square, multiplying the piece type and the number of squares under attack. The total penalty for this category is the sum of these values for all attacking pieces. This mechanism promotes offensive pressure while encouraging the agent to protect its own king.

**Note:** These rewards are calculated independently of whether the agent's pieces are actually occupying the squares near the king or not.

#### 3.2.1 Maximum reward possible for this category

Theoretically, we can of course imagine a position as follows: Here, almost all black pieces are attacking the squares near the king, namely:

- Queen: d5, d4, d3, e5, f4 $\rightarrow 5 \times 1.0 = 5$
- 1st Bishop: d4, e3 $\rightarrow 2 \times 0.5 = 1$
- 2nd Bishop: f3 $\rightarrow 1 \times 0.5 = 0.5$
- 1st Knight: e3, e5 $\rightarrow 2 \times 0.3 = 0.6$
- 2nd Knight: e3, e5 $\rightarrow 2 \times 0.3 = 0.6$
- 1st Rook: d5, e5, f5 $\rightarrow 3 \times 0.5 = 1.5$
- 2nd Rook: d3, e3, f3 $\rightarrow 3 \times 0.5 = 1.5$
- 1st Pawn: d5 $\rightarrow 1 \times 0.2 = 0.2$
- 2nd Pawn: d5, f5 $\rightarrow 2 \times 0.2 = 0.4$
- 3rd Pawn: e5 $\rightarrow 1 \times 0.2 = 0.2$
- 4th Pawn: f5 $\rightarrow 1 \times 0.2 = 0.2$
- King: e3, f3 $\rightarrow 2 \times 0.2 = 0.4$

Therefore, in total, that would be: 12. (If our agent is playing Black)

In practice, of course, it is impossible to have a similar situation on the board unless one is playing against themselves with the intent to fail. In real games, the king is often shielded by its own pawns or well surrounded by its own pieces, limiting the number of possible squares of direct threats. Furthermore, advancing high value pieces like the queen near the enemy king (while friendly pieces are still nearby, which is often the case) often results in material loss. Thus, in a practical situation, we can typically expect a strong attack when our pieces are attacking 2 to 3 squares around the king using valuable pieces such as the queen, rook, or bishop. For example:

- Queen attacking 2 squares: $2 \times 1.0 = 2.0$ points

- Rook attacking 1 square: $1 \times 0.5 = 0.5$ points
- Second rook attacking 1 square: $1 \times 0.5 = 0.5$ points

This results in a practical maximum reward of approximately 3.0 points, with typical values ranging between 2 to 4 points during standard play.

#### 3.2.2 Simple example of these rewards

Figure 4: King Attack Reward Example

In this case, for White: the reward is calculated as follows:

-1.0 (the black queen is attacking h2) + (-1) × (-1.0 × 2) (the white queen is attacking e7 and e8) = 1

For Black:

-1.0 × 2 (the white queen is attacking e7 and e8) + (-1) × (-1.0) (the black queen is attacking h2) = -1

### 3.3 Capture Rewards and Loss Penalties

The final reward mechanism concerns capturing and losing pieces, a cornerstone of effective chess strategy. While tactical sacrifice in some cases can be justified material loss such as when delivering a checkmate or gaining a positional advantage, material superiority still remains as one of the most reliable indicators of success. In chess, retaining more pieces on the board permits the agent with more options for attacking, defending, and controlling key squares. It provided more strategic possibilities and allowed for greater flexibility. As a result, maintaining material can help protect the king more effectively or increase pressure on the opponent's king and vital pieces.

In the endgame, material advantage becomes more pronounced. A single pawn can be decisive, especially when it advances the last rank and promotes itself to a queen. Therefore, integrating material evaluation into the reward system is essential for guiding the agent toward sound decision-making and efficient piece management.

To reflect this, we assign rewards based on the type of captured piece:

- Pawn: +1 point
- Knight or Bishop: +3 points
- Rook: +5 points
- Queen: +9 points

Simultaneously, if the agent loses a piece, it incurs a penalty equal to double the value of that piece.

#### 3.3.1 Maximum reward possible for this category

For this reward category, everything is much simpler. Since it is only possible to capture one piece per move, and the highest reward is given for capturing a queen, the maximum reward per move is 9

points. This ensures that while material gain is incentivized, it does not overshadow critical rewards like checkmate (100 points) or key positional advantages.

### 3.3.2 Simple example of these rewards

Below, you can see the initial board state, followed by the next state where a back-rank pawn has been captured.

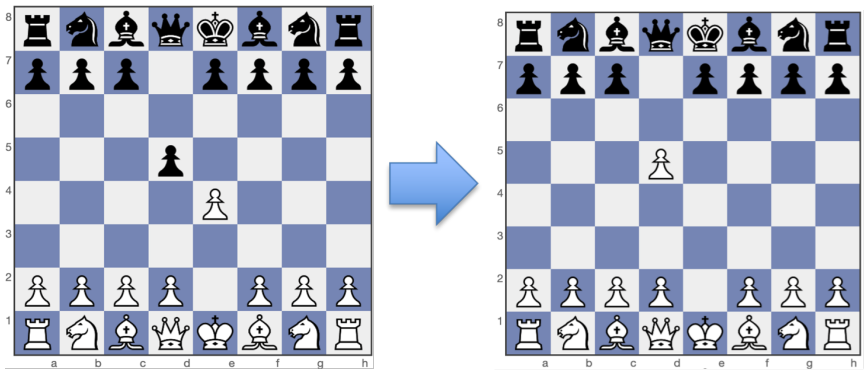

Figure 5: Material Capture Reward Example

In this case, the rewards are as follows:

- For White: $+1$
- For Black: $-1 \div 2 = -\frac{1}{2}$

## 4 Experiments and Results

We used four metrics to compare against the reference model, MADRL Chess. The metrics are overall rewards, total moves, number of checks and lastly, checkmate rates.

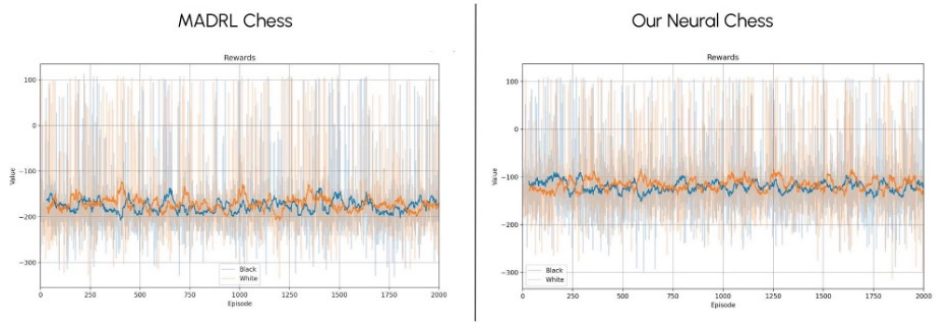


Figure 6: Reward Comparison Between MADRL Chess and Our Neural Chess Over 2000 Episodes

For overall rewards, the MADRL Chess model had average values around -200, especially for black. This shows that black did not do as well as white and that the results were inconsistent across games. Our Neural Chess model did better since rewards went up to about -100 for both black and white. This means overall performance improved and both players were more evenly matched. It also shows that our model was more balanced and stable as training went on.

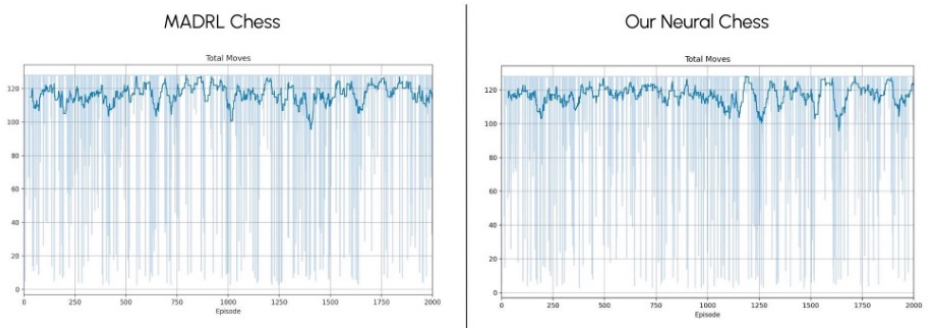


Figure 7: Total Number of Moves per Episode for MADRL Chess and Our Neural Chess

Looking at the number of moves per game, both models showed similar results. Most games lasted between 100 and 120 moves. This means that game length did not change much after training our new model. So even though the model improved in other ways, it didn't make games shorter or longer overall.

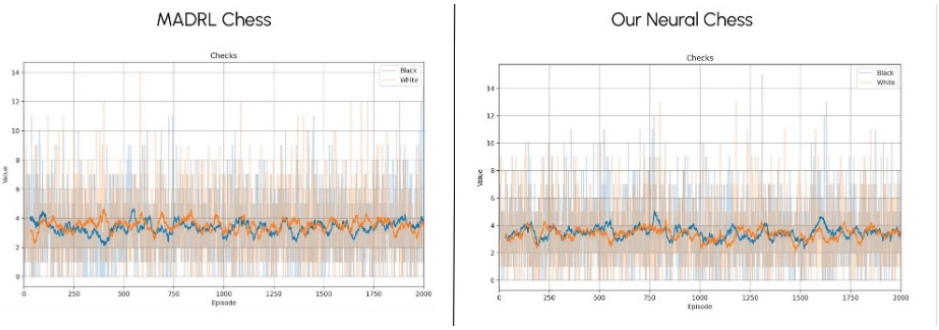


Figure 8: Number of Checks by Black and White Agents Over Training Episodes

When we look at how often players gave checks, both models had around 3 to 4 checks per game. In the MADRL model, white gave more checks than black which means that white was more aggressive. But in our Neural Chess model, the number of checks was more evenly split between both sides. This shows that our model helped black play more offensively too, and overall, the gameplay became more balanced in terms of attacking.

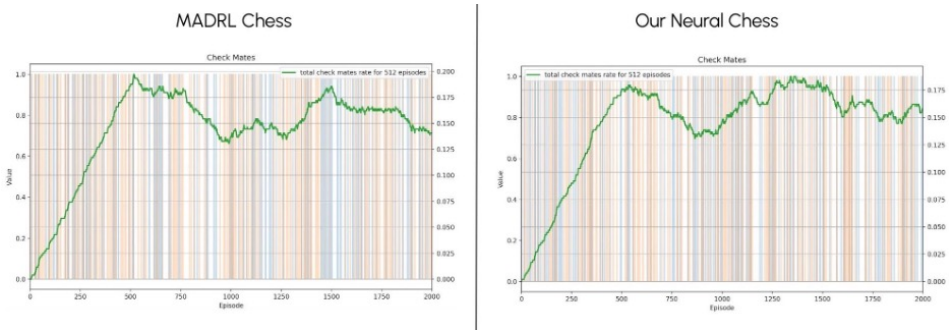


Figure 9: Checkmate Rate Across Episodes for MADRL Chess and Our Neural Chess

The checkmate rate tells us how often games ended in checkmate. In the MADRL model, checkmate rate went up quickly at first and peaked around 1.0 by episode 500, but it slowly dropped to 0.7 by the end. In our model, the checkmate rate also increased quickly and peaked a bit lower at 0.95, but it stayed more stable afterward at around 0.8. That means our model was better at consistently finishing games with a checkmate even in later episodes.

# 5 Conclusions and Future Work

In conclusion, our Neural Chess model demonstrated clear improvements over rewards, checks and checkmate consistency compared to the baseline MADRL framework. It achieved higher cumulative rewards, exhibited more consistent check and checkmate patterns, and showed a more balanced and strategic style of play. Although average game length remained largely unchanged, the overall quality and effectiveness of gameplay improved, indicating that the shaped reward mechanisms contributed positively to the agent's decision-making.

Looking ahead, there are several promising directions for enhancing this work. One area of improvement lies in refining the central occupation reward mechanism by considering the stage of the game: whether it is the opening, middlegame, or endgame While central occupation is critical during development phases, its importance in the middlegame and endgame diminishes as the game progresses, as at this point, we must concentrate on how to develop these strategies and attack the king. Another compelling avenue is design of a context-aware positional security reward. This proposed mechanism would evaluate how many of the agent pieces (e.g., those controlling the center, defending our king, or attacking the opponent's key pieces) are attacking our key pieces in the current position, and how many of our own pieces are protecting them. This idea seems quite interesting, and worth trying to implement and compare results, but unfortunately, it is quite hard to implement, and we lacked time.

# 6 References


Aditia. (2023). Reinforcement learning in chess. Medium. https://medium.com/@samgill1256/reinforcement-learning-in-chess-73d97fad96b3

Chess.com user. (2016). Evaluating positions. https://www.chess.com/forum/view/general/what-are-good-things-to-look-at-when-evaluating-positions

Chessprogramming Wiki. (2022). AlphaZero. https://www.chessprogramming.org/AlphaZero

Chessprogramming Wiki. (2025). Leela Chess Zero. https://www.chessprogramming.org/Leela_Chess_Zero

Chessprogramming Wiki. (2025). NNUE. https://www.chessprogramming.org/NNUE

GeeksforGeeks. (2024). Reward functions in RL. https://www.geeksforgeeks.org/how-to-make-a-reward-function-in-reinforcement-learning/

Klein, D. (2022). Neural networks for chess. arXiv. https://arxiv.org/abs/2209.01506

Mahyar. (2023). Simple MADRL Chess. GitHub. https://github.com/mhyrzt/Simple-MADRL-Chess

Reddit user. (2022). Importance of center. https://www.reddit.com/r/chess/comments/ula83b/why_is_it_important_to_control_the_center/

Stack Exchange user. (2018). RL for chess-like games. https://ai.stackexchange.com/questions/4482/how-could-i-use-reinforcement-learning-to-solve-a-chess-like-board-game

Yannic Kilcher. (2018). What is Q-Learning. YouTube. https://www.youtube.com/watch?v=nOBm4aYEYR4