Chess Move Prediction Using Deep Learning

Alec Nipp, Benjamin Zhang, Kevin Dai, Jack Huo

COMP 562 Final Project Report

Abstract

Chess is a historically influential game that has transcended generations, and still remains popular to this day. There are three main outcomes to a chess game: white wins, black wins, or a draw. Using an online database from Lichess, which is one of the largest free, open-source chess servers, a multitude of machine learning methods can be used to predict the outcome of chess games. This report seeks to predict the outcome of chess games using two primary methods: neural networks and logistic regression.

Contents

1 Introduction

Chess is among the oldest games in the world. It is a two player game with a seemingly endless player base.[3] Chess, with its vast complexity and strategic depth, provides an ideal test bed for exploring how early-game decisions influence overall results. This study leverages modern machine learning techniques, including neural networks and logistic regression models, to predict game outcomes from the first few moves in a match. The motivation for this stems from our desire to understand patterns and decision-making in chess games, particularly how the sequence of initial moves can determine the trajectory of a match. By focusing on the first couple of moves, we aim to simplify the predictive task while capturing critical insights into the game's dynamics. The scope of this project encompasses the processing of hundreds of thousands of chess games in Portable Game Notation format, the creation of feature representations, and the implementation of machine learning pipelines to predict match results.

2 Related Works

From analyzing games to building chess bots, predicting the outcome of chess games has been an area of interest within the broader domain of machine learning and game theory for many years. Various approaches have been employed to forecast results, leveraging historical game data, player statistics, and board positions. Recent research has examined various approaches, including logistic regression, decision trees, random forests,

gradient boosting machines, support vector machines, k-nearest neighbors, and neural networks. David et al. demonstrated the potential of deep neural networks in chess prediction in 2016 with the introduction of DeepChess.[1] More recently, Sofia DeCredico compared each of the aforementioned machine learning methods and found that ensemble methods like Random Forests generally outperform simpler models. In addition, neural networks showed promise, with the caveat that they required more careful tuning and larger computational resources.[2]

3 Data Processing

3.1 Initial Cleaning

The dataset consisted of chess games stored in Portable Game Notation (PGN), a standard format widely used for recording games. The primary goal of the data processing was to clean, tokenize, and encode the moves and results of each game into a format suitable for machine learning models. The first step in processing was to clean the PGN strings. We first created a function to remove move numbers, annotations (e.g., !, ?,), and comments, ensuring only the sequence of moves was retained. For instance, annotations such as "Good opening move by White" were stripped away, leaving a sequence of moves like "1. e4 e5 2. Nf3 Nc6".

3.2 Tokenization and Vocabulary Creation

We tokenized the cleaned move sequences into individual moves, such as e4, e5, or Nf3. We then constructed a vocabulary dictionary, mapping each unique move to a corresponding integer token. This step ensured the dataset was compatible with machine learning models, which require numerical inputs. Additionally, we created a reverse mapping to allow conversion from numerical tokens back to move strings for interpretability.

3.3 Sequence Padding and Tail Removal

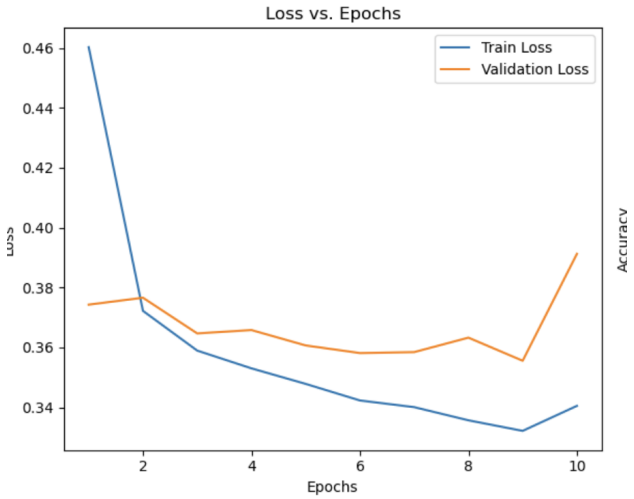
Since we were interested in predicting the outcome, we removed the last few moves of each game to simulate incomplete games and predict outcomes based on partial sequences. A parameterized approach removed the last five moves from the end of each sequence. As such, only games with sufficient tokens remaining after trimming were included in this subset. In order to standardize input lengths, the move sequences were either padded with <PAD> tokens or clipped to a fixed length. This helped ensure that all games had the same number of tokens, allowing batch processing during model training. Games that were too short were padded, while longer games were clipped to the desired length of 150 tokens.

4 Model Development and Results

4.1 Neural Networks

Our first method focused on developing and optimizing neural network architectures for multi-class classification tasks. Specifically, two models were explored: a basic feedforward neural network and an improved, enhanced model incorporating more advanced features. The simple model consists of three

fully connected layers, using ReLU activation functions for non-linearity and CrossEntropyLoss for multi-class classification. This architecture is lightweight and effective for smaller datasets, but struggles in high-dimensional data. To fix this, the enhanced model introduces an embedding layer, which is particularly useful for categorical inputs, and allows the model to learn dense representations of discrete features. We split the dataset into a 70% training, 15% validation, and 15% testing split. This stratification allows for a robust tuning and testing process. Next, we began the optimization process with Stochastic Gradient Descent for the simple model. However, for the enhanced model, the Adam optimizer was employed, which combines the benefits of adaptive learning rates and momentum to accelerate convergence. Performance metrics include loss, accuracy, and additional classification metrics such as precision, recall, and F1-score. These were calculated for both training and validation phases to analyze performance and detect overfitting. The use of a validation set also ensured that the best model was saved based on its lowest validation loss, preventing degradation in generalization ability. Finally, to assess final model performance, predictions on the test set were compared with ground truth labels, achieving a test accuracy of 88.28%. Removing games that ended in draws and modifying the neural network to perform binary classification yielded an accuracy of 91.62%.



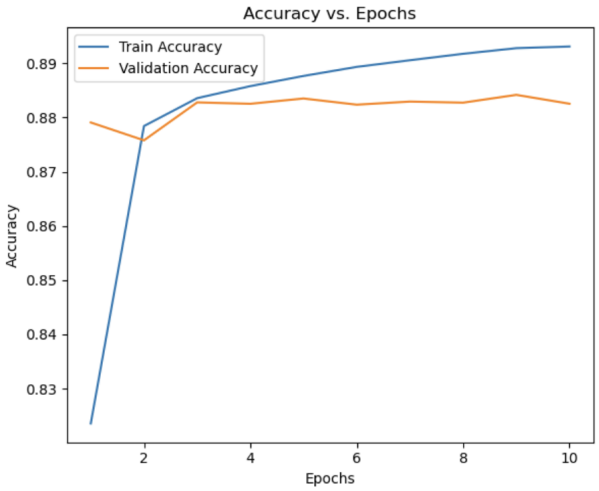


Figure 1 (Above) Loss vs. Epochs and Figure 2 (Below) Accuracy vs. Epochs

4.2 Logistic Regression

Next, we sought to quantify the relative values of each piece. Most often, pawns are assigned a value of 1, with knights and bishops worth 3, rooks worth 5, and queens worth 9. However, many have theorized the values to be different, such as the first American world chess champion Bobby Fischer, who thought that bishops were actually worth slightly more than knights with a value of 3.25. Recently, a method using the strong chess engine AlphaZero determined the relative values of the pieces to be, with pawns assigned a value of 1, knights worth 3.05, bishops worth 3.33, rooks worth 5.65, and queens worth 9.5.[4]

To tackle this problem, we chose to use logistic regression. The main motivation behind this approach was to take a position, say 30-40 moves into a game, count up the number of each piece on both sides, and use a logistic regression model to classify whether white will win or lose. The resulting model will have coefficients that we can interpret as the (log) relative value of each piece. The final value for a piece that is reported is the average of its value for both black and white, relative to the average value of a pawn.

To avoid class imbalances, we took a random sample of 10000 white wins and 10000 black

wins, with a train test split of 80-20. Using the position after 40 moves in each game, the result was 77.83% accuracy. With the pawn being value 1, the model assigned knights a value of 1.65, bishops a value of 1.69, rooks a value of 2.66, and queens a value of 9.08. We do see that the relative ordering of the piece values are the same as conventional wisdom. However, the value of knights, bishops, and rooks are far off conventional wisdom and AlphaZero's analysis, but we see a similar relative value of the queen. While this could certainly be due to the limitations of our approach and model, it could also reflect a fundamental difference in how strong the knight, bishop, and rook are to an average human player versus their theoretical values or their values in the hands of a strong chess engine. For example, even though two knights and a bishop should be theoretically worth the same as a queen, the average human player may find it more difficult to orchestrate those three pieces as effectively as one queen.

5 Conclusion

This article demonstrates how neural networks and logistic regression can be applied to predict the results of chess and evaluate the value of the pieces of chess. The enhanced neural network performed well, achieving 88.28% accuracy. This predictive power demonstrates the model's ability to interpret move sequences and recognize the features of a black/white victory, but the exact underlying logic remains opaque. Logistic regression offered a simpler and more interpretable approach, providing insight into piece values, although some results differed from traditional wisdom and engine analyses.

These findings show the promise of these techniques in revealing new perspectives on chess strategy and game play. However, there is room for improvement, such as refining neural network models for more complex data and understanding why piece values vary between human and machine play. Furthermore, the impact of draw outcomes on overall accuracy is a salient point of improvement, as drawn games represent only a fraction of the total number and are harder to predict. Removing them resulted in a marginal improvement in accuracy to 91.62%. More detailed examination is required to better account for their unique characteristics and influence on model predictions. Future research could also explore the combination of different approaches to increase accuracy and deepen understanding of chess dynamics.

References

[1] Omid E. David, Nathan S. Netanyahu, and Lior Wolf. Deepchess: End-to-end deep neural network for automatic learning in chess. In *Villa, A., Masulli, P., Pons Rivero, A. (eds) Artificial Neural Networks and Machine Learning – ICANN 2016.*, 2016.

[2] Sofia DeCredico. Using machine learning algorithms to predict outcomes of chess games using player data. Master's thesis, Rochester Institute of Technology, 2024.

[3] A. Revel. Chess games, version 1, January 2021. Retrieved December 8, 2024 from https://www.kaggle.com/datasets/arevel/chess-games.

[4] Nenad Tomasev, Ulrich Paquet, Demis Hassabis, and Vladimir Kramnik. Assessing game balance with alphazero: Exploring alternative rule sets in chess. *CoRR*, abs/2009.04374, 2020.