Chess Rating Estimation from Moves and Clock Times Proposal

Michael Reda Riad 900203291
Abdelrahman Ihab Elazab 900213468
The American university in Cairo
CSCE460402: Advanced Machine Learning
Fall 2025
Dr. Moustafa Youssef

Problem statement

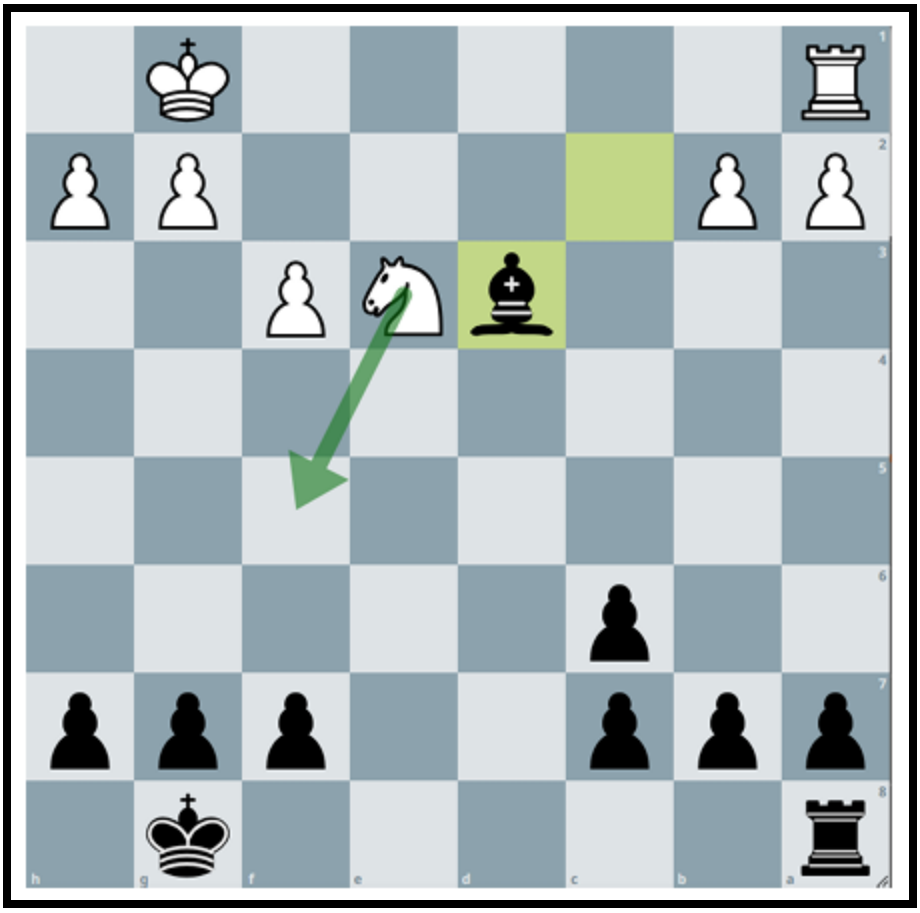
Given a sequence of chess moves (game history) and per-move clock information, predict each player's Elo rating (continuous value) using a deep-learning model trained on large-scale online chess games.

Motivation

- How linked your player rating is to your playing strength is considered a rating accuracy. Even in matchmaking, coaching, adaptive interfaces, and anti-cheating systems, estimating how a player will behave early on in a game will always be useful.

- Unlike handcrafted-statistic approaches, a deep model can learn spatial patterns (board configurations) and temporal dynamics (move sequences and time usage).

- This is a realistic, constrained, and reproducible deep-learning project: the input format is compact (8×8×12 tensors per move), the datasets (Lichess, FICS) are openly accessible, and a few GPU days will train models on several hundred thousand games.



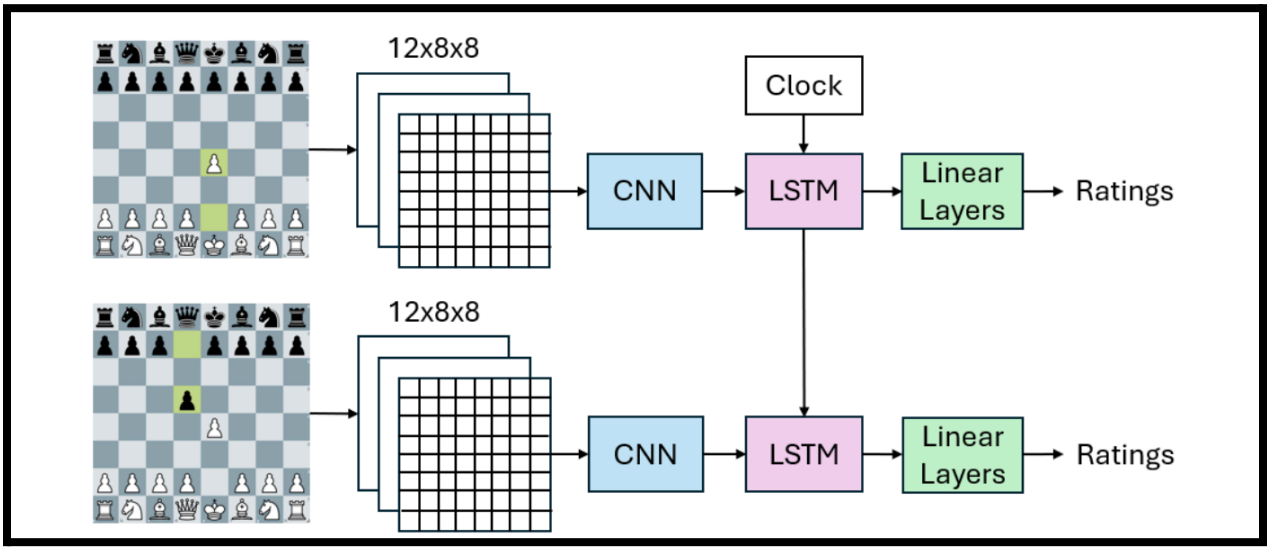
In this figure, the white played the move NF5 which is considered to be a bluder move because this Knight will be taken for free by the bishop on d3; so this decreases the rating of the white player (Omori and Tadepalli, 2024).

Input Data

- **Board state** as **12 binary 8×8 planes** (AlphaZero-style: one plane per piece–color; 1 where that piece is present, else 0).
- **Clock feature: remaining clock time** for that move, included as a numeric feature and **standardized**; ratings are standardized for training as well.

Output Data

Rating for both players



Survey of Evaluation Metrics

Given the nature of the problem–predicting continuous rating values; we wil focus on regression metrics. The most common metric is the **Mean Absolute Error (MAE)** because it offers an interpretable estimation of an average rating error. Omori and Tadepalli (2024) reported an MAE of 182 points. MAE is meaningful to the chess community as it is close to the 200-point class divisions in USCF ratings (the rating system that is used in USA). We also look at **Mean Squared Error (MSE)** and **Root Mean Squared Error (RMSE)** as they punish large deviations, which also was used in the IEEE Big Data Cup Chess Puzzle Competition (IEEE, 2024). We also plan to use the **coefficient of determination (R^2)** and **correlation coefficients** (Pearson and Spearman) to show how much ratings predictions explain, in variance, and how predictions preserve rank order (Chowdhary, Iacopini, & Battiston, 2023). Once the ratings are discretized into bins, classification metrics become relevant including **accuracy**, **F1-score**, and **confusion matrices,** particularly for uncovering systematic misclassifications (Tijhuis, Blom, & Spronck, 2023). As is the baseline approach, we will also use time control (bullet, blitz, rapid, classical) to stratify the evaluation. The prediction difficulty varies by format. All metrics can be implemented using Scikit-learn tools.

State of the Art

- **Omori & Tadepalli (2024) — "RatingNet" (CNN + LSTM):** trained on 1.2M Lichess games to predict *per-move* player rating from 8×8×12 board planes + per-move remaining time.
    - **Overall test MAE (Mean Absolute Error): 182 Elo**
    - **Per time-control MAE:** UltraBullet **186**, Bullet **182**, Blitz **183**, Rapid **182**, Classical **151**.
      Ablation (no clock feature): MAE worsens to **239** overall; clock time reduces error by **57 Elo** on average, and by **92 Elo** in Bullet
    - **MSE (for completeness):** overall **56,618** vs **91,619** without clock.
    - **Puzzle transfer (IEEE BigData Cup 2024):** public-LB **MSE 82,049** using the same architecture.

  Takeaway: For *rating regression from moves + clock*, the best published number is **MAE ≈ 182 Elo**, with clear gains from including time-usage.

Survey of Available Datasets

There are plenty of chess datasets available online that we can use in our training:

- Lichess Open Database: this data set has chess games that are played online each month and they are free to download.
- Free Internet Chess Server (FICS): This is an online source that we can get games from and they are free to download.

Chosen Dataset and its description

Dataset Description

For this project, we use the Lichess Open Database of standard rated games. It contains billions of games in PGN format, published monthly. Each record includes player ratings (Glicko-2), results, opening codes, move sequences, and — in most games since 2017 — per-move clock times (%clk).

Why this dataset?

Scale: Billions of games provide enough data to train deep learning models.

Rich features: Includes move sequences, time usage, and metadata like openings and results.

Labels: Player ratings serve as reliable continuous targets for prediction tasks.

Accessibility: Freely available, open-licensed, and supported by common tools (e.g., python-chess).

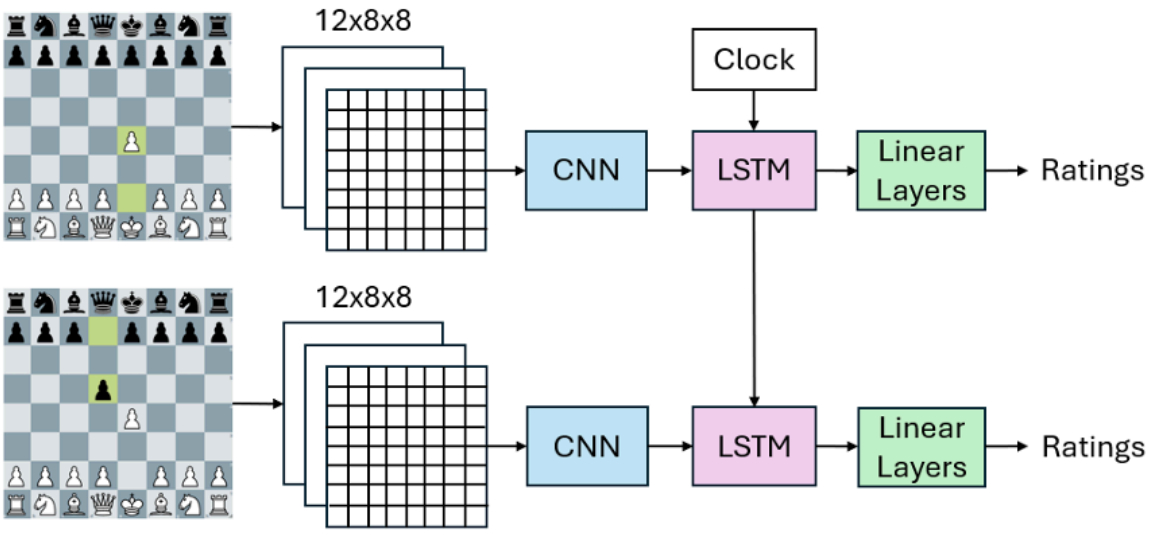
Reproducibility: Public domain status ensures others can replicate our experiments.

Survey of Avaiable Models

A. RatingNet — CNN + BiLSTM baseline (move-by-move rating regression)

Main idea & architecture: RatingNet encodes each board state as 12 input planes (piece × color), processes each board with a 4-layer convolutional network (CNN) to extract spatial features, concatenates a standardized clock-time feature, and feeds the sequence of per-move embeddings into a bidirectional Long Short-Term Memory (LSTM). The LSTM output passes through two fully connected layers to regress continuous player ratings at every move (move-by-move prediction). This model intentionally avoids hand-crafted features and predicts ratings continuously rather than binning them.

Reference (paper): Omori, M., & Tadepalli, P. (2024). Chess rating estimation from moves and clock times using a CNN-LSTM.



Public repo: https://github.com/AstroBoy1/RatingNet

Framework: PyTorch

Weights/model zoo: The paper reports training and provides code; checkpoints are available in the repository or can be reproduced from their training script.

Training resources reported:Trained on a single NVIDIA A40 GPU for ~12 hours using batch size 32, LR 1e-4, weight decay 1e-5, for ~50 epochs (hyperparameters listed in the appendix).
.
Reported performance: Mean Absolute Error (MAE) ≈ 182 rating points across Lichess test split; ablation (no clock) increases MAE substantially (paper table)

B. Glicko-2

Main idea & architecture: Glicko-2 is a rating system focused on statistical accuracy. After a rating period, it calculates expected scores and rating variances (v) then, it updates volatility with Illinois or other root- finding algorithms, and adjusts ratings. Glicko-2 updates only after a set of games or a tournament and not after every turn, thus it is and will always be the classical interpretable system for what a rating is.

The Glicko-2 system updates a player's rating by considering three key values:

- **Rating (μ)**: The player's skill level.

- **Ratings Deviation (φ)**: The degree of uncertainty in the player's rating. A lower value means the rating is more reliable.
- **Rating Volatility (σ)**: The degree of expected fluctuation in a player's rating over time.

$$v = \left[\sum_{j=1}^{m} g(\phi_j)^2 E(\mu, \mu_j, \phi_j) \left\{ 1 - E(\mu, \mu_j, \phi_j) \right\} \right]^{-1} \tag{1}$$

where

$$g(\phi) = \frac{1}{\sqrt{1 + 3\phi^2/\pi^2}} \tag{2}$$

$$E(\mu, \mu_j, \phi_j) = \frac{1}{1 + \exp(-g(\phi_j)(\mu - \mu_j))} \tag{3}$$

$$\Delta = v \sum_{j=1}^{m} g(\phi_j) \left(s_j - E(\mu, \mu_j, \phi_j) \right) \tag{4}$$

$$\phi^* = \sqrt{\phi^2 + \sigma'^2} \tag{5}$$

$$\phi' = \frac{1}{\sqrt{\frac{1}{\phi^{*2}} + \frac{1}{v}}} \tag{6}$$

$$\mu' = \mu + \phi'^2 \sum_{j=1}^{m} g(\phi_j) \left(s_j - E(\mu, \mu_j, \phi_j) \right) \tag{7}$$

Reference: Glickman, M. E. (Glicko-2 documentation).

Public repo/implementations: Many language bindings and packages implement Glicko-2 (e.g., R PlayerRatings::glicko2)

.

Framework/weights: Analytic algorithm (no training weights).

Training Resources: None; computation is minimal and involves iterative and closed-form numerical procedures.

Reported performance / relevance: Not an ML/DL model — used as a baseline to illustrate how rating systems have reacted historically and to promote per-move learning approaches. It is used to evaluate high-level rating trends and to illustrate whether the learned model generates ratings that are closer to real results.

Chosen Model (Baseline) and its Description

The model is composed of a 4-layer CNN which encodes each board position. Its features are concatenated with the per-move remaining clock time and passed to a bidirectional LSTM, then two FC layers to predict both players' ratings after each move.

Why this model?

It exactly matches our task (continuous rating regression from moves + time) and is reproducible on an open benchmark; it achieves MAE ≈ 182 Elo overall and shows clear gains from the clock feature versus a no-clock ablation.

Time Control	RatingNet	RatingNetNoClock	Mean
Average Test Loss	182	239	346
Ultrabullet Loss	186	269	250
Bullet Loss	182	274	340
Blitz Loss	183	244	378
Rapid Loss	182	207	305
Classical Loss	151	185	246

Source Code

This is the link for our source code files: **Baseline code reference**

Model Weights/Zoo

No weights are released.
Plan: train from scratch, efficiently.

- Subset → full: validate on a stratified 50–100k–game slice, then scale.

- Efficiency: mixed precision (AMP), gradient accumulation, bucket by game length, early stopping, epoch checkpoints.
- Warm-start (no external zoo): brief pretraining on next-move or rating-bin classification, then switch to rating regression.
- Scope control: start with a narrower CNN/LSTM, or shorter game segments / specific time controls if compute is tight.
- Report: MAE (primary) + MSE/R^2, with Mean and No-Clock baselines.

Our Proposed updates

1) **Increase CNN Depth**: Increase the CNN from 4 layers to 6–8 layers, and possibly residual connections, to better learn more complicated spatial relationships on the board, including long-range interactions among pieces and positional structures pulled from pieces.
2) **Stack Additional LSTM Layers**: Increase the number of BiLSTM layers or the number of hidden units to allow the network to better model long-term dependencies when learning from moves, especially if learning from longer classical games.
3) **Add Engine-Derived Features**: Add variables for centipawn loss and probability of a blunder from Stockfish policy evaluations as additional supplemental inputs in order to improve the representational learning of move quality.
4) **Apply Data Augmentation**: Utilize board symmetries (mirroring/rotation) and swapping player color to increase the dataset size and better generalize.
5) **Improve Training Strategy**: Use cosine learning rate schedules, stronger weight decay, and early stopping to improve training stability and model overfitting.

More profound CNNs are likely to yield more powerful spatial feature extraction from board states, as pointed out by the success of AlphaZero (Silver et al., 2017) using deep residual CNNs in chess. Adding more LSTM layers will improve modeling of the sequential dependencies between moves, in keeping with results established by analysis of sequential models (Graves & Schmidhuber, 2005). Features from the engine will introduce domain knowledge that is directly related to skill, as implied by studies on intrinsic rating using centipawn loss (Regan & Haworth, 2011). The game of chess provides effective methods for data augmentation and exporter augmentation for both board symmetries and player color, and it has been shown to improve robustness in the evaluation of human-style chess players (McIlroy-Young et al., 2020). Finally, new training strategies such as advancing learning rate schedules and stronger regularization could help reduce convergence time while generalizing

after deployment. Collectively, these possible updates would be expected to produce less prediction error and a stronger, more trustworthy model, while we kept the architecture very similar to RatingNet to ease reproducibility.

Chosen Evaluation Metrics

In the paper by Omori & Tadepalli (2024) titled "Chess Rating Estimation from Moves and ClockTimes Using a CNN-LSTM" which we consider the "state of the art" paper in which we will base our work on, they used the mean absolute error as the primary metric for rating prediction. They also used the mean squared error which was provided for the game rating task and was used as the official metric when the same architecture is evaluated on the IEEE Big Data Cup 2024 Chess Puzzle Difficulty dataset.

Graduation Project

This project is not associated with any of our graduation projects.

Other Projects (ML/DL/CV)

Last Semester I have taken Fundamentals of Machine learning course here in the university where I worked on a project to predict the sleep stage prediction for a group of people using wearable sensor data, where I applied preprocessing, feature engineering, and models such as Random Forest, SVM, and neural networks, achieving around 86% accuracy using the Random Forest (GitHub link). This work gave me experience in handling sequential data, evaluating multiple models, and drawing parallels between time-series classification which I believe will help me in this DL project. (Michael Reda)

Each Member Contribution

Michael Reda Riad	Abdelrahman Ihab Elazab
Proposed the idea of extending RatingNet with deeper CNN/LSTM layers.	Suggested exploring dataset availability (Lichess, FICS) and confirmed Lichess is the best option.

Collected and filtered research papers related to rating estimation (RatingNet, Elo/Glicko, intrinsic ratings).	Collected and filtered papers related to alternative approaches (single-game models, cheat detection).
Focused on surveying model architectures, extracting main ideas, and preparing proposed updates.	Focused on surveying datasets, analyzing accessibility and size, and summarizing their suitability.
Drafted a comparison of models and organized references for the literature review.	Gathered some research papers related to our topic and ensured they have available GitHub links and code.

References

- Chowdhary, S., Iacopini, I., & Battiston, F. (2023). Quantifying human performance in chess. Scientific Reports, 13(1), 2113. https://doi.org/10.1038/s41598-023-29272-6
  IEEE. (2024). Big Data Cup 2024: Predicting chess puzzle difficulty. KnowledgePit. https://knowledgepit.ml/predicting-chess-puzzle-difficulty/

- Omori, M., & Tadepalli, P. (2024). Chess rating estimation from moves and clock times using a CNN-LSTM. arXiv preprint arXiv:2409.11506. https://arxiv.org/abs/2409.11506

- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830. https://www.researchgate.net/publication/51969319_Scikit-learn_Machine_Learning_in_Python

- Tijhuis, T., Blom, P. M., & Spronck, P. (2023). Predicting chess player rating based on a single game. In 2023 IEEE Conference on Games (CoG) (pp. 1–8). IEEE. https://doi.org/10.1109/CoG58078.2023.10324584

- Snyder, J.: Inverse interpolation, a real root of f (x)= 0. University of Illinois Digital Computer Laboratory, ILLIAC I Library Routine H1-71 4 (1953) https://scispace.com/pdf/several-new-methods-for-solving-equations-3s8oa3otks.pdf

- Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks, 18(5–6), 602–610. https://doi.org/10.1016/j.neunet.2005.06.042

- McIlroy-Young, R., Sen, S., Kleinberg, J., & Anderson, A. (2020). Aligning superhuman AI with human behavior: Chess as a model system. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 1677–1687). ACM. https://doi.org/10.1145/3394486.3403219

- Regan, K. W., & Haworth, G. (2011). Intrinsic chess ratings. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 25, No. 1, pp. 834–839). AAAI. https://www.researchgate.net/publication/221606300_Intrinsic_Chess_Ratings