בס"ד

Final Project Report

Chess Data Science Analysis

Team Members

- Avi Klings, avi.kfir@mail.huji.ac.il, ID: 318251519, CSE: avi.kfir
- Shachar Levy, [redacted], CSE: shahar99
- Romy Tzafrir, [redacted], CSE: romytz

Problem Description

Train a model that predicts the best next move from a given chess position.

Data

1. "Chess Games" dataset from Kaggle, which contains 6 million games played on lichess.org, a free and open-source chess platform. The full dataset (≈4 GB) is too large to include due to project size limit, so we provided a shorter CSV sample named "chess_games.csv".
   The full dataset is at this link: https://www.kaggle.com/datasets/arevel/chess-games
   It includes metadata for each game, such as player ratings (WhiteElo, BlackElo), game result (Result), game type (Event), termination reason, ECO code and opening name, time control settings, and the full sequence of moves in PGN (Portable Game Notation).
2. Games played by Grandmasters (rated 2500+, 2 million games): https://database.nikonoel.fr/
   We used 6 monthly slices: 2020-07/08/09 and 2023-07/08/09. Due to project size limit, we provided "07_23.csv" only. All 6 X,y features are cached.

Our solution

We investigated a simple question: Can a model learn to predict the best next move from past games using supervised learning? Early trials with logistic regression and KNN quickly revealed their ceiling, pushing us toward neural networks. Inspired by the book **"Neural Networks for Chess"** (included in our project), we focused on (1) representing each board state, (2) defining exactly what belongs in the input **X** versus the label **y**, and (3) making training reproducible and fast by caching **X,y** and the models instead of rebuilding them each run. Also set up a Colab with Drive workflow to train with free GPUs. Together, these choices formed a reliable pipeline from raw PGNs (Portable Game Notation) to trained models we can evaluate and deploy.

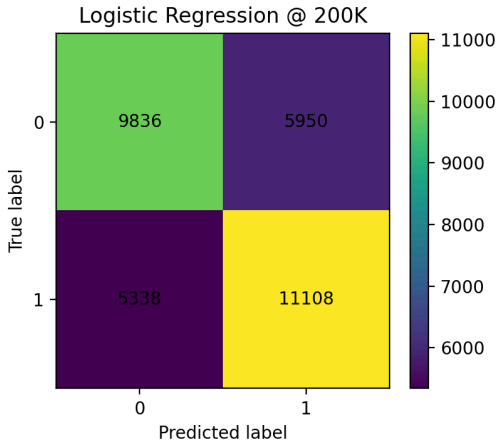
important note: In our models (Logistic Regression, KNN, CNN), we trained only on the board position and the side to move (White or Black). We did not include any additional features such as player rating, game length, or metadata. The goal was for the model to learn directly from the board position itself, to understand the game state, evaluate who is ahead, and predict the best next move purely from chess knowledge (without external hints).

At first, we thought to start by testing simple models to understand how far they could go. This was meant as a kind of sanity check, to see whether they truly struggle to generalize when faced with a challenging task like chess. To accommodate the simple models, we even reduced the level of difficulty and tested whether they could predict who would win (white or black) given a board position of move 20. This means After the opening phase has passed and the game has developed further. We filtered out all games from the dataset number 1 (chess games of all levels) that were shorter than 20 moves.

Since the skill levels of the players are varied, it is unlikely that the game is still balanced by move 20, as these are not professional players. We also excluded all games that ended in a draw.

We really tried to make the task as easy as possible, presenting the simplest question. If the models performed well, we planned to increase the difficulty. We did not want to jump straight to using heavy models. The results were not impressive, so we moved on to neural networks afterward.

Here are the results based on 200K samples from different games, with an 80% training and 20% testing split. We continued increasing the number of samples until we observed that the metrics plateaued. The models produced nearly identical results when trained on 800K samples compared to 200K.
Logistic regression (F1 score: 0.67) outperformed KNN (F1 score: 0.57).

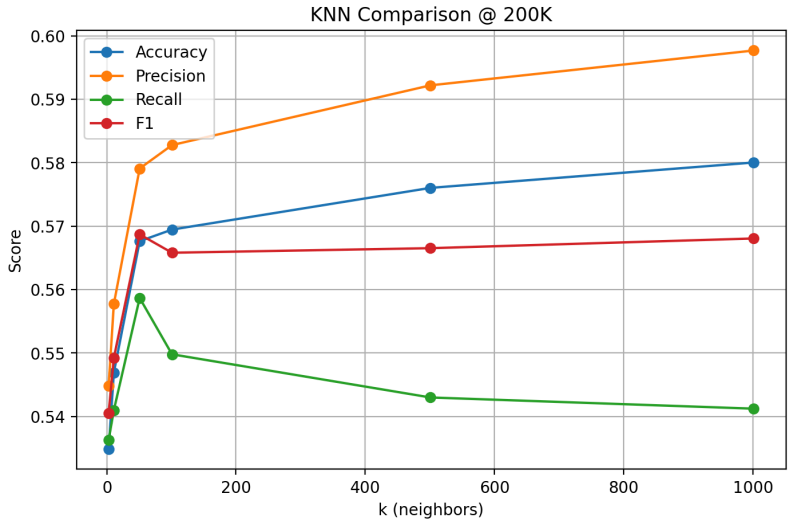



```
Processing chunks: 100%|██████████| 17/17 [11:13<00:00, 39.63s/chunk]
Done: X=(847429, 832), y=(847429,)
Built and cached X,y to models\xy_800K.npz (X=(847429, 832), y=(847429,))
Dataset: X=(847429, 832), y=(847429,)

------- Option 1: Logistic Regression -------

Evaluation Metrics:
Accuracy : 0.66
Precision: 0.66
Recall   : 0.68
F1 Score : 0.67
Confusion Matrix:
 [[51911 30923]
 [27535 59117]]

Process finished with exit code 0
```



Moving to Neural Networks. Supervised learning quickly taught us a simple rule: a network learns what we feed it. If we train on blunders and mistake-ridden games (noise), we produce a mistake-prone model. So we switched the diet from dataset number 1 (mixed-skill games) to dataset number 2: high-quality games only (players rated 2500+). The model was supervised on 2M samples, where **X** denotes the encoded board position and **y** the corresponding optimal move label. An average chess game contains about 80 board states (40 full moves). We sampled one position per game from 2M games, rather than ~80 positions from 25,000 games (25,000 × 80 = 2M).

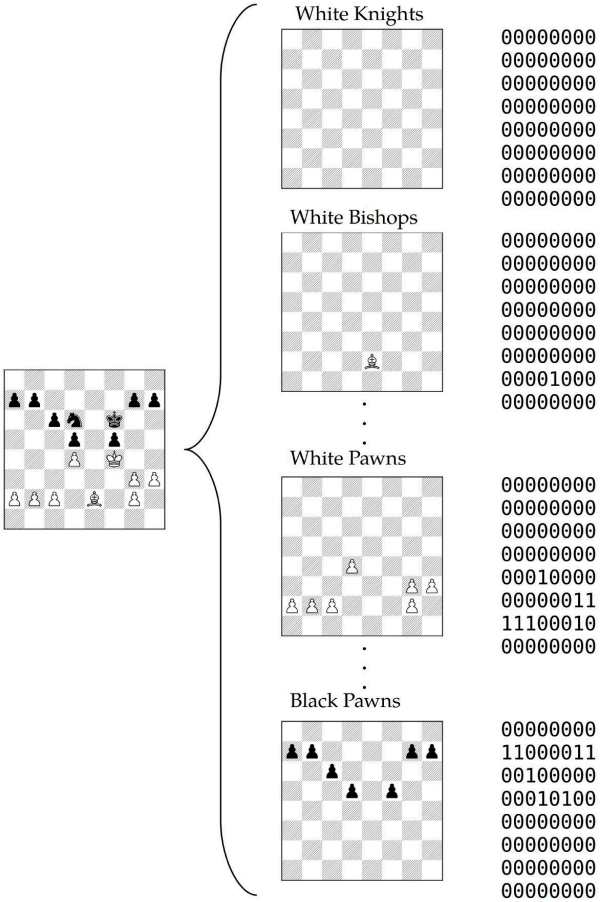
It surely would be much faster to just find, read and process 25K games (CSV rows) instead of 2M! The reason is overfitting. When the network was trained with full games, the network would adjust too much to the specifics of these games. In other words the network tended to memorize full games, instead of abstracting the underlying patters and actually learn from the games.
Initially, we sampled randomly from each game. This caused the model to overfit to common openings and perform poorly in the middle and endgame. To address this, we adjusted the sampling: only 2% of positions are taken from the opening phase, while 98% are drawn from middlegame and endgame positions. This gave the network more training examples from the harder, less repetitive phases of the game, where deeper chess understanding can be learned.

We deliberately used a compact CNN due to compute limits. Our compute budgets wouldn't support a deep network. Instead, we use a few conv layers with batch-norm and ReLU, then a small fully-connected head that outputs a probability over a fixed 4,672-move vocabulary (all possible moves in a chess board). Inputs are bitboards: we encode a board at a fixed point in a game as 13×8×8 (=832) float32 planes:

- 12 planes for piece occupancy (6 per color)
- 1 plane for side-to-move indicator

[0–5] : White pieces [PAWN, KNIGHT, BISHOP, ROOK, QUEEN, KING]
[6–11] : Black pieces [PAWN, KNIGHT, BISHOP, ROOK, QUEEN, KING]
[12] : Side-to-move indicator (all ones if White to move, else zeros)

To stay fast and memory light, we omit extras like repetition counters, 50-move plane, and castling masks. In short: a policy CNN that's intentionally not deep, so we can train it with Google Colab GPUs.

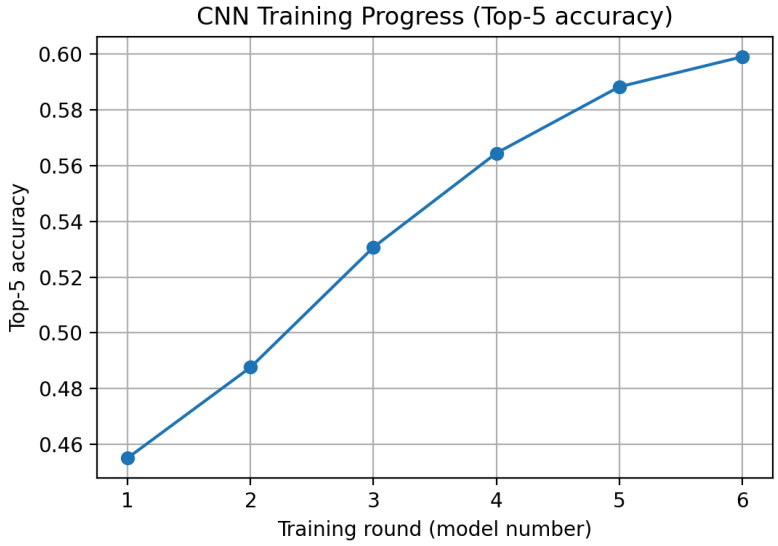


Why CNN? Convolutional Neural Networks are well-suited for chess because the board is a structured 8×8 grid, similar to an image. CNNs can efficiently capture local spatial patterns and deeper hierarchical structures (tactics, formations, strategic motifs) as layers stack. They exploit the 2D nature of the board by sharing weights (filters) across the grid, greatly reducing parameters while still capturing the essential structure, which makes training more efficient. Each chess board position is encoded as a 13×8×8 (=832) float32 tensor. By comparison, a fully connected network would require an enormous number of parameters to process this directly.

We chose Top-K (Top-5) Accuracy for CNN evaluation. In chess, unlike many other classification tasks, there is rarely a single "perfect" move in most positions. In many board states there are multiple reasonable best moves to consider. If we only measured Top-1 accuracy, checking whether the model's most probable move exactly matches the "ground-truth next move (label y)", then any situation where the model preferred another strong candidate move would be unfairly counted as an error. This is misleading because often the model predicted the "ground-truth next move (label y)" in second, third, fourth or fifth place. After the CNN outputs probabilities across the entire move space (4672 possible moves), we check whether the "ground-truth next move (label y)" is within the top 5 moves with the highest predicted probability. Practically, it captures the idea that the CNN has "understood the position" if it ranks the correct move among its top predictions.

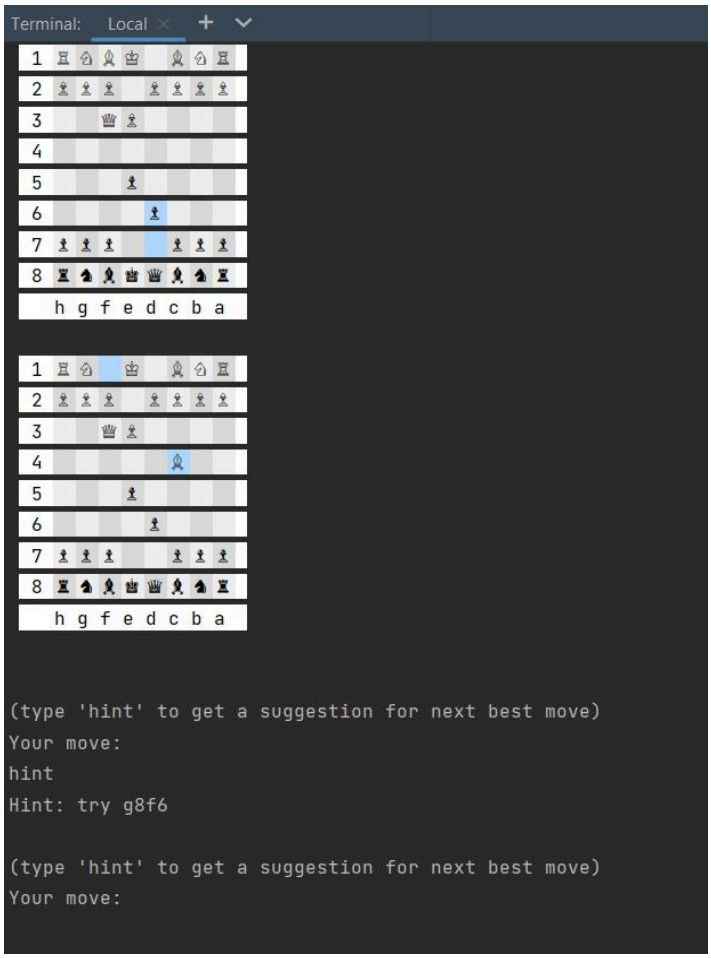
In our experiments, the best CNN model achieved a Top-5 accuracy of 0.60. In other words, in 60% of positions the ground-truth move was ranked among the five moves with the highest predicted probability. When we tested the CNN policy by actually playing games, we observed its limitations.
Opening phase: the model often reproduced strong and well-known opening moves, which reflects the fact that these were common in the training data.
Middlegame and endgame: performance dropped significantly. The CNN frequently suggested weak or losing moves, showing that it had not truly learned the deeper strategy of chess.

Our compact CNN policy head: Conv(13→64) → BN → ReLU → Conv(64→128) → BN → ReLU → Flatten(8192) → Linear(256) → BN → ReLU → Linear(4672). Outputs raw logits for the 4,672 move slots. Move encoding: 4,672 classes = 3,584 queen-like lines + 512 knight jumps + 576 under-promotions. Training loop: 80/20 index-split, BATCH_SIZE=1024, EPOCHS=6, LR=1e-3, weight decay 1e-4, label smoothing 0.10. Best checkpoint saved by TopK, macro precision/recall/F1, and Top-K (K=5) each epoch. Inference: The policy masks illegal moves, renormalizes, and selects either argmax or a sample with temperature.



We tried adjusting training parameters, changing data sampling, and extending training time, but we could not push performance beyond 0.6 Top-5 accuracy. The most reasonable conclusion is that our model is not deep or expressive enough to capture the complexity of chess.
This result also highlights a broader point. Supervised CNN training aimed at imitating human games has inherent limitations. Such models can learn patterns in openings and common tactical motifs, but without deeper architectures or reinforcement learning (as in AlphaZero) they struggle to generalize to novel positions or to plan strategically over many moves.

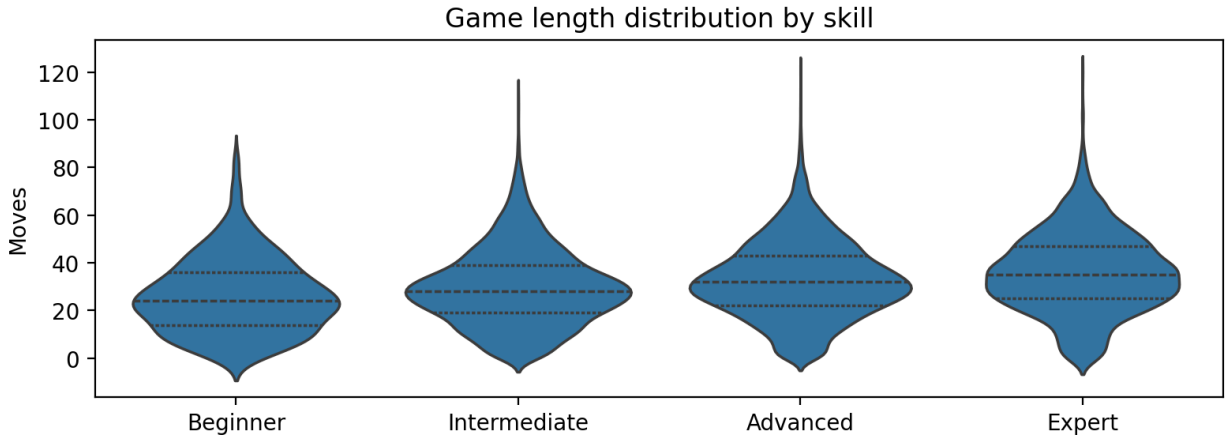
You can play chess in the Terminal against our CNN engine (instructions provided in the README).



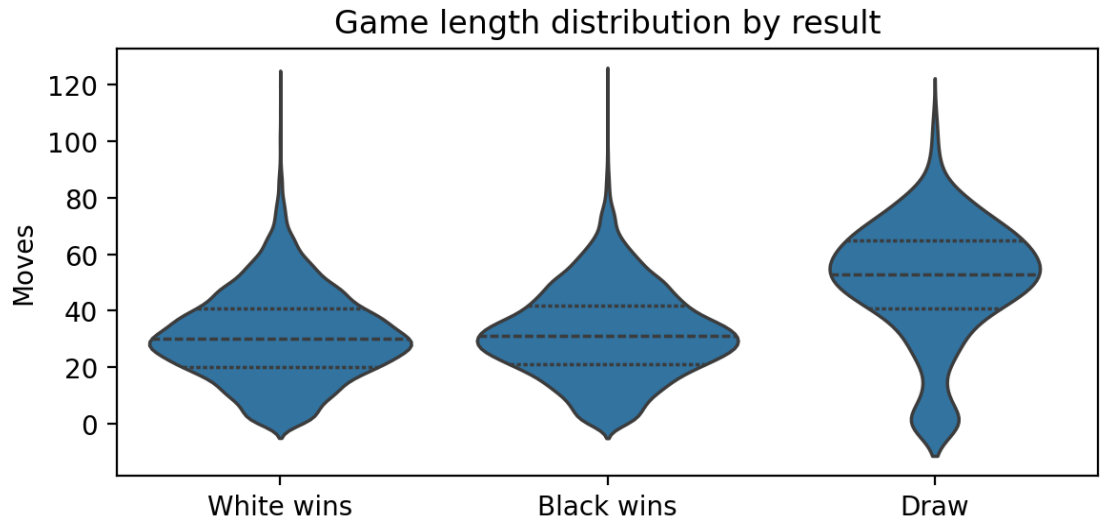
Future Work

- Add a hybrid mode that orders minimax moves by the network's probabilities so CNN guide search. This will give a meaningful boost without retraining.
- When resources allow: deepen the trunk into a modest residual stack and consider richer input planes (e.g., castling rights, repetition, 50-move counter).
- Deeper neural architectures: Larger CNNs or transformer-based models could capture more complex patterns and long-term dependencies.
- Reinforcement learning with self-play: Instead of only imitating human data, training through self-play would allow the model to explore novel strategies and learn from its own experience. A hybrid approach that starts with supervised training on grandmaster games and then applies reinforcement learning fine-tuning could balance fast initial learning with the development of deeper strategic understanding.

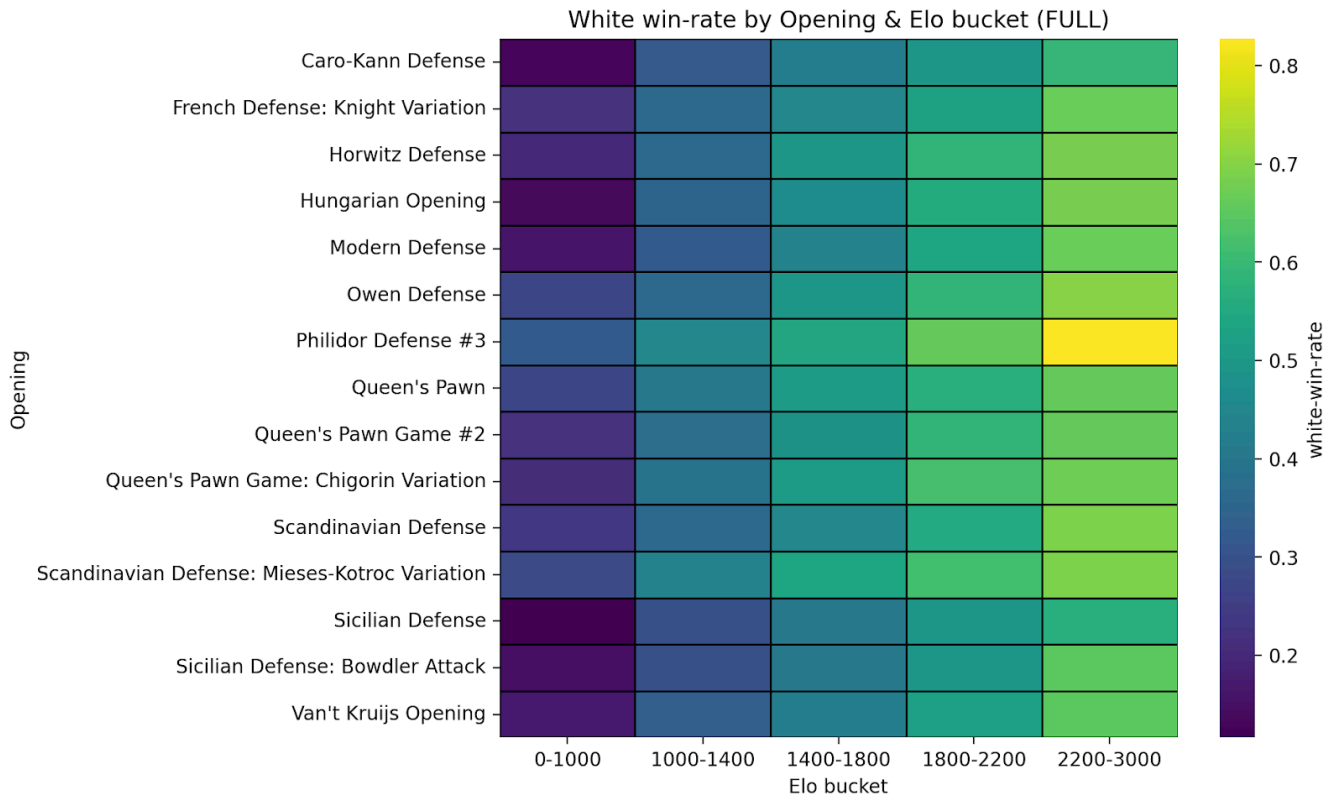
Insights and conclusions revealed in our analysis:



The violin plot shows the distribution of game lengths (in moves) across four skill levels: Beginner, Intermediate, Advanced and Expert. This plot is particularly interesting because it highlights how game duration evolves across different skill levels, and the use of violin plots is well-suited here as it not only shows the typical game lengths but also the full distribution, making it easy to compare both the central tendency and the spread across groups.
We observe that Beginner players are more likely to conclude their games in the early stages, as indicated by the wider base of the violin compared to higher skill levels. This suggests that novices often make early blunders that lead to quick losses. Additionally, the relatively short tail beyond 60 moves shows that Beginner players rarely sustain very long games. Still, the presence of a moderate upper tail suggests that some games are prolonged, likely reflecting situations where inexperienced players struggle to convert positions or find a clear plan for making progress. As expected, the Intermediate players show a more balanced distribution, with most games centered around 20 to 40 moves. Advanced and Expert players generally play longer games, with higher typical game lengths. Their distributions are narrower compared to Beginner players, suggesting greater consistency in game duration. This trend becomes noticeable at the Advanced level and is even stronger among Expert players, where games are both longer and more stable. Overall, as skill increases, games tend to last longer and become more predictable, reflecting improved defensive play and reduced early mistakes.

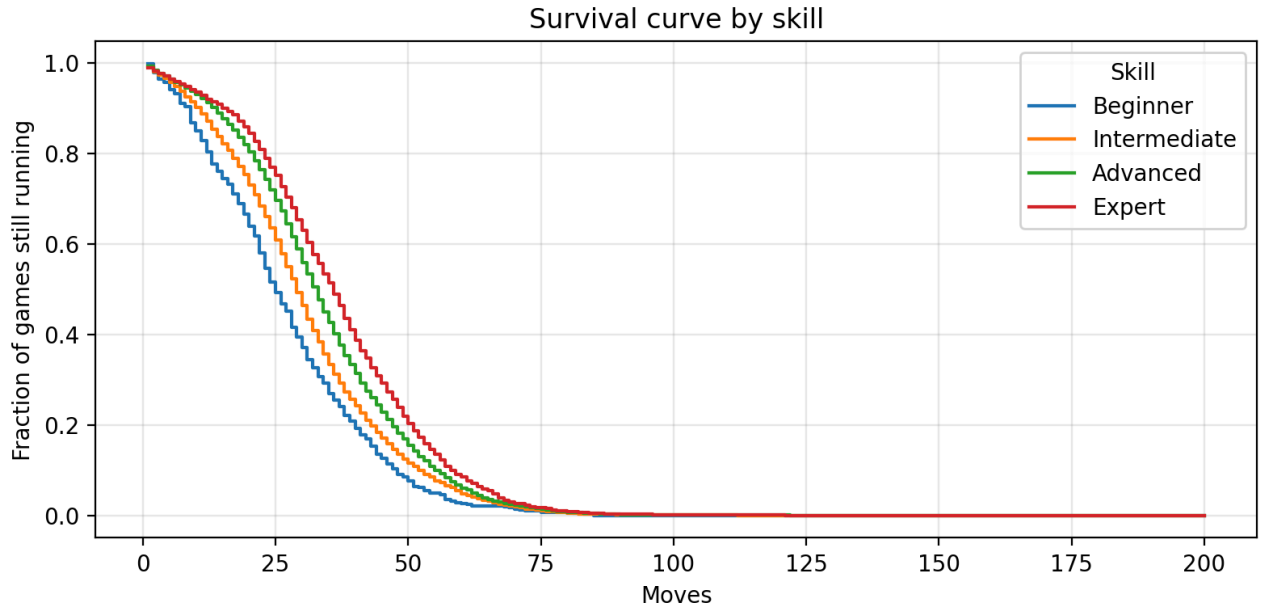


This violin plot compares game lengths for three possible outcomes: White wins, Black wins, and Draws. The distributions for both White and Black wins cluster around 20-40 moves, suggesting that decisive results in chess typically occur in relatively short to medium-length games. On the other hand, draws show a distinct pattern with a clear shift toward longer game lengths, often exceeding 50 moves. This is expected in chess, since draws typically occur in balanced positions where neither side can convert an advantage, leading to extended play. The small peak near very short draws indicates the occurrence of quick draws, such as those resulting from agreed draws or repetition in the opening. This plot emphasizes that decisive results often require fewer moves, while drawn games tend to involve more extended struggles. This plot is particularly interesting because it reveals how game length differs across outcomes, and the use of violin plots is ideal here as it not only visualizes the central tendency of game durations but also captures the entire distribution, allowing for an easy comparison of both the concentration and variation in game lengths for each result type.



This heatmap displays White's win rate for the top 15 most common openings, broken down by Elo rating buckets. A clear upward trend is visible across most openings: White's win rate increases with Elo, peaking in the highest bracket (2200-3000). This suggests that stronger players exploit the initiative of the first move more effectively. Some openings, such as the Philidor Defense, show relatively high win rates for White at higher Elo levels, possibly due to structural weaknesses in these defenses that stronger players can exploit more effectively - though this may also be influenced by the limited number of games played with this opening in the dataset, which can make the win rate less reliable. In contrast, more balanced openings, like the Caro-Kann Defense and Modern Defense, maintain lower White win rates across all Elo levels, consistent with their reputation as solid defenses. The variance across openings highlights how choice of opening interacts with skill: weaker players are less able to capitalize on theoretically strong openings, while stronger players convert these advantages more reliably.

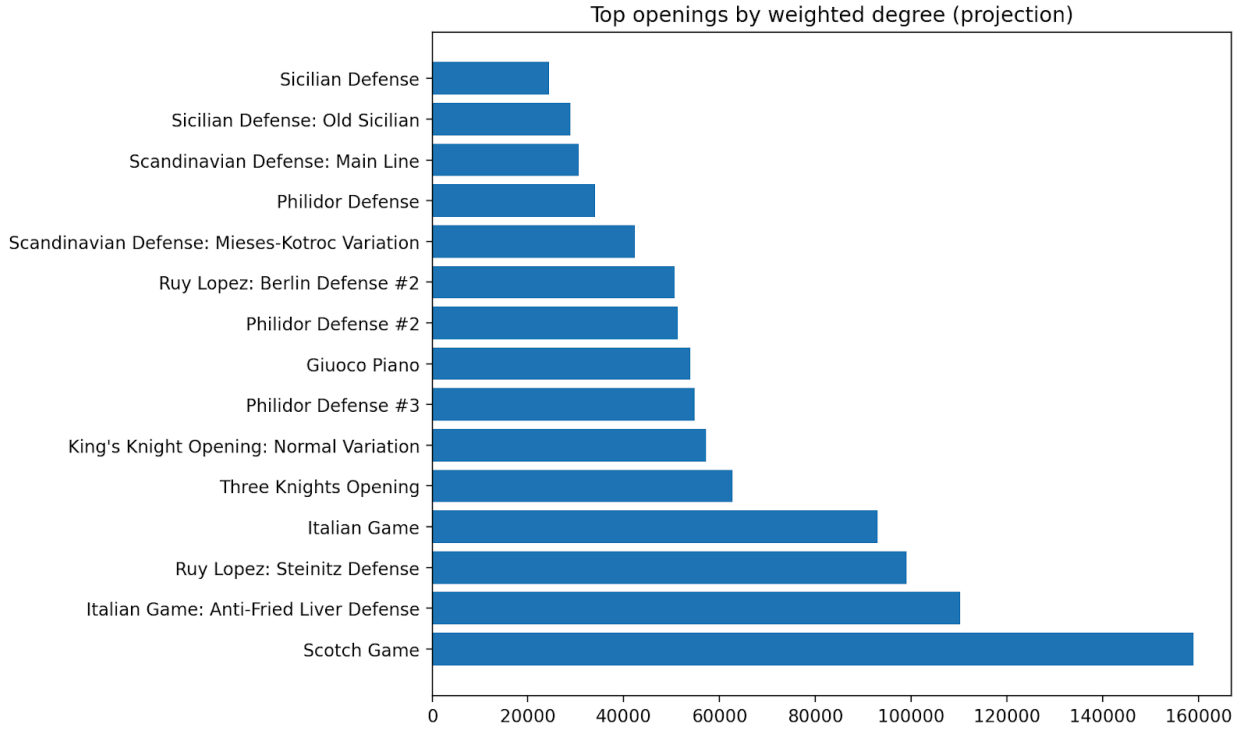
This analysis demonstrates the interplay between opening theory and player skill, showing that both opening choice and player rating significantly influence win probabilities. This heatmap is especially insightful because it provides a detailed view of how opening strategies and player skill interact to influence outcomes. The heatmap format is particularly effective here, as it not only reveals the win rates for different openings across Elo levels but also allows for easy comparison of trends and outliers across both variables, making it simple to visualize how win rates shift as players' skills increase.

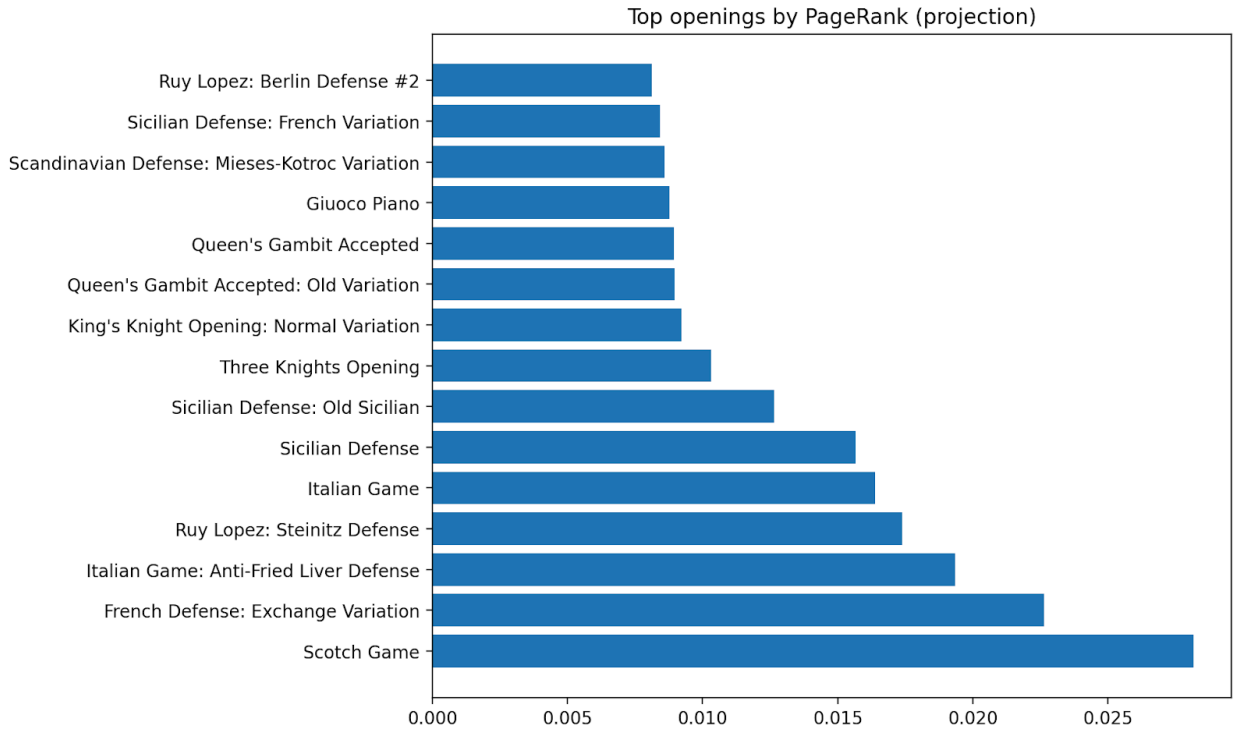


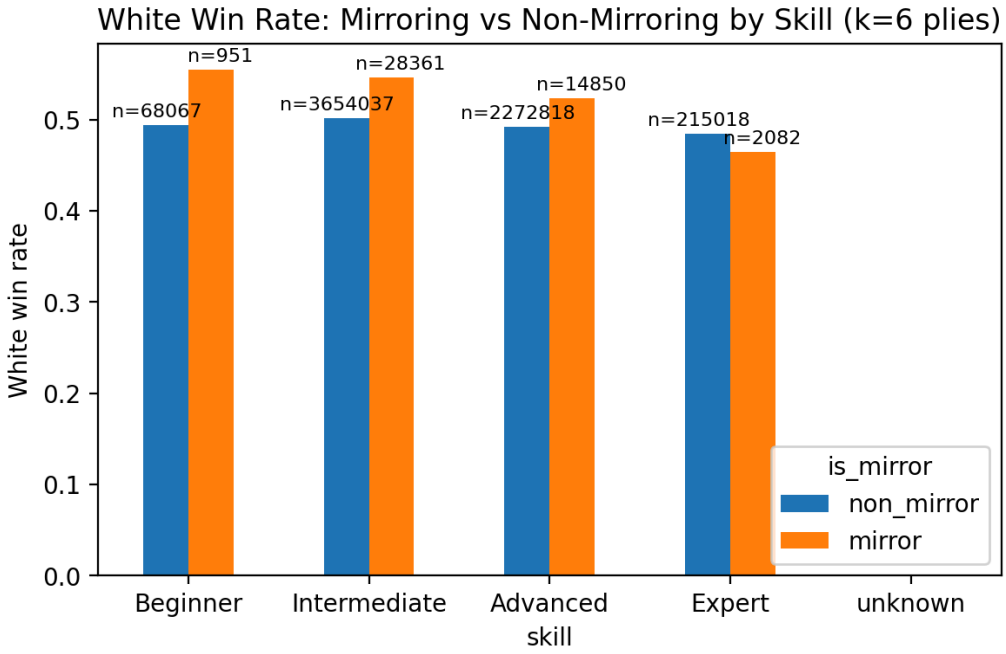
This survival curve illustrates how the fraction of games still ongoing decreases over time, categorized by player skill level (Beginner, Intermediate, Advanced, Expert). The plot reveals that lower-skilled players (Beginner and Intermediate) tend to have shorter games, with more games concluding in fewer moves, while higher-skilled players (Advanced and Expert) have games that last longer, with a higher proportion of games still ongoing at a greater move number. This suggests that stronger players engage in more complex positions that require more moves to reach a conclusion, while less experienced players tend to reach a result more quickly, either through mistakes or decisive errors. The survival curve format is particularly useful here as it provides a clear comparison of how game duration evolves over time across skill levels, showing that more skilled players tend to extend their games into more complex stages. Unlike other graph types, such as histograms, it offers a cumulative view of game progression, highlighting how games unfold and change as they progress.

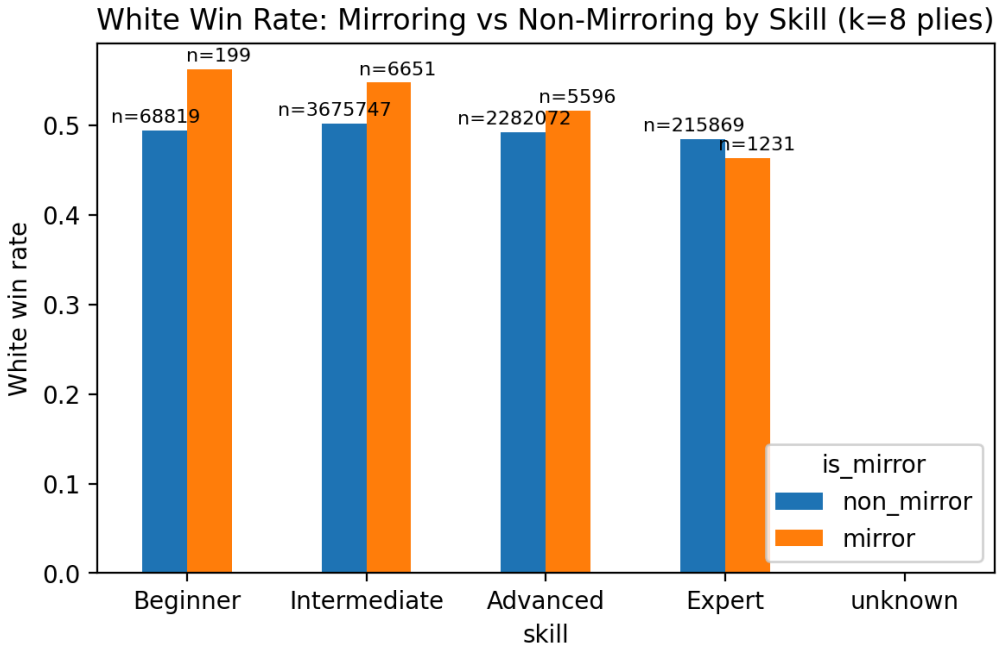
The two figures below show different measures of centrality applied to chess openings, namely Weighted Degree Centrality and PageRank. Weighted degree centrality, shown in the first figure, reflects the frequency and significance of a given opening within the network of chess moves. For example, the Scotch Game is ranked highest in this measure, indicating it is one of the most central openings. On the other hand, PageRank in the second figure measures the authority or influence of an opening, taking into account not only the direct connections but also the importance of the openings that are linked to it. Here, Ruy Lopez: Berlin Defense #2 emerges as the most authoritative, showing that it has a significant influence in the overall opening network. Using centrality metrics like weighted degree and PageRank allows for a comprehensive analysis of both popularity and importance, offering insights into how certain openings dominate or influence the progression of games.

בס"ד


Top openings by weighted degree (projection)
Sicilian Defense
Sicilian Defense: Old Sicilian
Scandinavian Defense: Main Line
Philidor Defense
Scandinavian Defense: Mieses-Kotroc Variation
Ruy Lopez: Berlin Defense #2
Philidor Defense #2
Giuoco Piano
Philidor Defense #3
King's Knight Opening: Normal Variation
Three Knights Opening
Italian Game
Ruy Lopez: Steinitz Defense
Italian Game: Anti-Fried Liver Defense
Scotch Game
0
20000
40000
60000
80000
100000
120000
140000
160000


Top openings by PageRank (projection)
Ruy Lopez: Berlin Defense #2
Sicilian Defense: French Variation
Scandinavian Defense: Mieses-Kotroc Variation
Giuoco Piano
Queen's Gambit Accepted
Queen's Gambit Accepted: Old Variation
King's Knight Opening: Normal Variation
Three Knights Opening
Sicilian Defense: Old Sicilian
Sicilian Defense
Italian Game
Ruy Lopez: Steinitz Defense
Italian Game: Anti-Fried Liver Defense
French Defense: Exchange Variation
Scotch Game
0.000
0.005
0.010
0.015
0.020
0.025





The bar charts above, compare the white win rates between mirroring and non-mirroring strategies across different skill levels (Beginner, Intermediate, Advanced, Expert), with evaluations conducted at 4, 6, and 8 plies. In most cases, the mirroring strategy (orange bars) shows a higher white win rate compared to non-mirroring (blue bars), suggesting that when both players adopt similar moves, the white player tends to have a better chance of winning. Specifically, for Beginner and Intermediate skill levels, mirroring becomes more effective as the number of plies increases. However, at the Expert skill level, the trend shifts, and non-mirroring outperforms mirroring at both k=6 and k=8. This indicates that mirroring's advantage may diminish as players' skills increase, where more sophisticated strategies and deeper thinking are involved.