Well-Calibrated Result Probabilities and Human Move Prediction for Elite Classical Chess

Albert W. Hamood — chessds.com — albert@chessds.com

Abstract

I describe two machine-learning models for elite classical chess: a result prediction model that outputs well-calibrated win/draw/loss probabilities for the current position, and a move prediction model that ranks legal candidate moves by how likely a strong human would play each one. Both are gradient-boosted decision-tree ensembles (LightGBM) built on features from Stockfish evaluations, an upstream human-imitation policy network, and a range of position- and game-level signals. The models are trained on ~266k classical chess games from The Week in Chess in which both players are rated 2400 Elo or higher, and evaluated on a 47k-game evaluation set held out from the training pool. On the result task the production model (which does not see the rating gap between the two players) reaches an expected calibration error of 0.002 on 4.17M held-out positions. On the move task the production model reaches 61.1% top-1 / 87.8% top-3 accuracy, versus 55.5% / 81.1% for an engine-best-move baseline evaluated on the same positions. Both models are deployed on chessds.com in two latency tiers (around 200 ms and 1 s per position on a single CPU core). I report scaling behavior, sliced metrics for both tasks, and a short ladder of toy baselines that situate the headline numbers against simpler alternatives.

1. Introduction

This paper describes two learned models for elite classical chess. The first takes a position together with the history of the current game and outputs probabilities over the three possible outcomes (white wins, draw, black wins). The second takes the same position and outputs a probability distribution over the legal moves, estimating how likely a 2400+ Elo human is to play each one.

Chess engines such as Stockfish (The Stockfish Developers, n.d.) produce strong numerical evaluations of positions, and neural-network systems such as Lc0 (The Lc0 Team, n.d.) and AlphaZero (Silver et al. 2017) produce move preferences through self-play. Engine evaluations are optimized for engine strength, not for estimating what humans actually do or for producing well-calibrated win probabilities. Neural move predictors trained on human games, such as Maia (McIlroy-Young et al. 2020) and its successors, show that a network can learn human-like play directly from move sequences. My approach here is hybrid: I train a small policy network on human games in the same family as Maia, and use both its outputs and Stockfish evaluations as features for gradient-boosted trees that predict result probabilities and rank candidate moves. This keeps training and inference cheap, produces well-calibrated probabilities in practice on this task, and makes it straightforward to ship two latency tiers from the same recipe.

I report results for the two current production models (one for each task). Both result-model variants — rating-gap-aware (RA) and rating-gap-unaware (RU) — see rating-

level information, but the RU model receives only symmetric features (specifically, the average rating of the two players, bucketed into 50-Elo bins and capped at 2700); only the RA variant additionally sees a side-specific rating-gap signal. The site defaults to RU. The goal is to represent the position itself as accurately as possible so that mistakes and strong moves can be judged objectively; when ratings are far apart that signal dominates and the position-quality signal becomes less interesting. The RA variant answers a different question — "given these two specific players, what outcome is likely?" — and is reported alongside the default in this paper.

2. Dataset

All games are drawn from The Week in Chess (Crowther, n.d.) issues 920 through 1639 (September 2012 to April 2026). From 3.08M total games I keep those in which both players are rated at least 2400 Elo (Elo 1978) and the time control is classical, yielding 313,119 games. The pool is split deterministically at the game level into a 266,152-game training pool and a 46,967-game evaluation holdout. The split is at the game level — every position from a given game ends up entirely in one of the two sets — so positions from the same game cannot leak across the split. The evaluation set is never touched during hyperparameter tuning or training.

The move-prediction model uses the same 266,152 / 46,967 split. Each game contributes roughly 88 plies on average, giving 23.5M training positions and 4.17M evaluation positions for the result task. For the move task each non-terminal position is exploded into one row per legal candidate move (≈30 on average), giving ~23.3M training positions and ~700M training candidate moves, and 4.12M evaluation positions / 122M evaluation candidate moves.

The filters above are exhaustive: classical time control, both players 2400+, valid result and ratings present. I do not filter by opening, tournament, online vs. over-the-board (OTB), or game length beyond what TWIC already curates. The 2400+ filter is chosen to match the target application (live analysis of elite classical chess); including faster time controls or lower-rated pools is outside the target distribution and, in earlier internal experiments not reported here, hurt performance on elite-classical evaluation at fixed training-set size.

3. Results

3.1 Result prediction

I report two metrics on the 47k-game evaluation set. **Normalized entropy (NE)** is the slice's log loss divided by the entropy of that slice's outcome distribution: NE = 1 corresponds to a log loss equal to the slice's own prior entropy (what a predictor that exactly tracked the slice's class frequencies would achieve), and NE = 0 corresponds to a model that places all probability mass on the realized outcome on every position. The latter is unattainable in chess: outcomes are not a deterministic function of position (the same position can be reached in two games and end differently), and even ignoring that, predicting blunders before they happen is genuinely hard. NE puts slices with very different inherent predictability — opening positions versus the last few plies of a game — on a common scale: the model's raw loss is

much lower in the endgame, but most of that gap reflects endgame outcomes being more determined rather than the model being more skilled there. **Expected calibration error (ECE)** bins predictions by the model's top predicted probability and reports the size-weighted average gap between predicted confidence and empirical frequency in each bin (Naeini, Cooper, and Hauskrecht 2015; Guo et al. 2017). Informally: when the model says 70%, is it right about 70% of the time? Note that ECE here is computed on the model's top predicted probability — a confidence-calibration measure — rather than per-class; per-class reliability is not separately reported. NE measures overall probabilistic quality (sharpness and calibration jointly), and ECE isolates whether the model's top-confidence probabilities can be read as frequencies. I drop top-1 accuracy because it discards probability information, is dominated by "obvious" positions such as terminal endgames, and is not a proper scoring rule for probabilistic forecasts.

Figure 1 places the production result model against a ladder of simpler baselines trained on the same data and evaluated on the same holdout. The class-prior baseline (predicting the empirical outcome distribution, independent of the position) sits at NE ≈ 0.999, as expected. A single-feature logistic regression on the engine's evaluation of the current position already cuts about 16% of the prior entropy (NE ≈ 0.839); the engine evaluation alone is a strong signal in the late middlegame and endgame, though nearly useless in the opening. Adding ply as a second feature is a small additional improvement (NE ≈ 0.830). A depth-5 decision tree on three features — engine evaluation, ply, and the rating gap between the players — reaches NE ≈ 0.776. The production rating-gap-unaware model improves on this simple tree on overall NE despite not seeing the rating gap (NE ≈ 0.744), and the rating-gap-aware model improves further to NE ≈ 0.712, with ECE ≈ 0.003 (RA) and 0.002 (RU).

Per-slice numbers are shown in Table 1. NE varies sharply across slices, as expected: positions near the end of a game are largely determined and easy to predict (NE ≈ 0.27), while opening positions are nearly as hard as the prior (NE ≈ 0.91). Concretely, the model explains about 73% of the prior entropy in the last eleven positions and only about 9% in the opening. The opening slice is also the only one where the rating-gap-aware decision tree (NE = 0.946) beats the production rating-gap-unaware model (NE = 0.963): in the opening, engine evaluation is nearly flat across moves and rating gap is the dominant predictor of outcome, and the RU model has rating *level* but not the rating *gap* between the two players. The production rating-gap-aware model wins the opening slice (NE = 0.910), as expected.

The calibration numbers in Table 1 are tight: overall ECE around 0.002–0.003 means that when the model reports a probability, that probability is, on average, within two to three tenths of a percentage point of the empirical frequency at that confidence level. This is the most user-visible property of the result model. The one slice where calibration is meaningfully weaker is the near-end-of-game slice (ECE ≈ 0.08, driven by the draw class). The mechanism is explained next.

Table 1. Per-slice normalized entropy and expected calibration error on the 47k-game evaluation set. RA = rating-gap-aware variant (sees the rating gap between the two players); RU = rating-gap-unaware variant (sees only symmetric rating-level features). Lower is better for both metrics. Slice definitions in Appendix A.

slice	n_positions	RA NE	RU NE	RA ECE	RU ECE
overall	4,169,548	0.712	0.744	0.003	0.002
opening	1,438,177	0.910	0.963	0.005	0.003
middlegame	1,796,903	0.691	0.717	0.003	0.002
endgame	934,471	0.457	0.467	0.004	0.003
near-end-of-game	516,637	0.274	0.287	0.074	0.078

The elevated draw-class ECE in the near-end slice has a specific source: short, GM-agreed draws on nominally-active positions. Strong players sometimes agree to draws in positions where a decisive result is still possible — the well-known "grandmaster draw" phenomenon. These early agreements skew the empirical draw rate above what the position would predict, and the effect is concentrated in shorter games where draw agreements occur more often. Bucketing the near-end positions by total game length makes this clean: in games under 30 plies — almost all of which end in agreed draws on positions the engine still considers active — the empirical draw rate is 97.7%, but the model assigns only 55.2% mean probability to draw, a gap of −0.42. The gap shrinks monotonically as game length grows, reaching −0.01 (well-calibrated) for games of 120+ plies that are played out to a real endgame.

Table 2. Draw-class calibration on near-end-of-game positions, bucketed by total game length. The under-prediction is concentrated in short games (agreed draws on still-active positions) and disappears in long games (played-out endgames where outcome follows from the position).

game length (plies)	n_games	n_positions	empirical draw rate	mean P(draw)	gap	draw-class ECE
<30	2,052	22,572	0.977	0.552	−0.425	0.425
30–39	1,627	17,897	0.887	0.512	−0.375	0.377
40–49	2,552	28,072	0.703	0.443	−0.260	0.271
50–59	3,460	38,060	0.473	0.352	−0.122	0.152
60–69	5,777	63,547	0.500	0.403	−0.098	0.120
70–79	5,465	60,115	0.353	0.311	−0.042	0.077
80–89	5,881	64,691	0.384	0.342	−0.042	0.074
90–99	4,527	49,797	0.371	0.339	−0.032	0.068
100–119	6,949	76,439	0.377	0.354	−0.022	0.059
≥120	8,677	95,447	0.435	0.425	−0.010	0.042

3.2 Move prediction

I report four primary ranking and distribution metrics for the move model, and separately evaluate calibration. Top-1 accuracy is the fraction of positions where the played move has the highest predicted probability among the legal candidates. Top-3 accuracy is the fraction of positions where the played move is in the top three. Mean reciprocal rank (MRR) averages 1 / rank(played) over positions, giving smooth partial

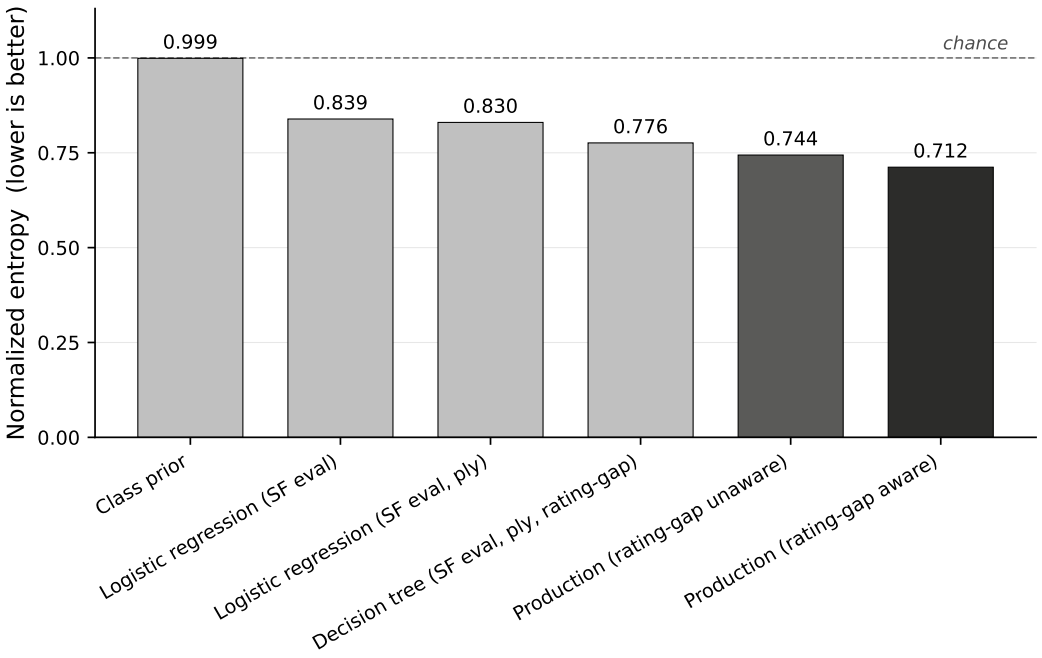

Figure 1: Result-prediction normalized entropy on the 47k-game evaluation holdout. Bars from left to right: class-prior baseline; logistic regression on Stockfish evaluation alone; logistic regression on Stockfish evaluation and ply; depth-5 decision tree on Stockfish evaluation, ply, and the rating gap between players; the production rating-gap-unaware model; the production rating-gap-aware model. Lower NE is better. The dashed line at NE = 1 indicates the slice's own prior entropy (chance-level performance).

credit when the played move is ranked second, third, or further down (Voorhees 1999). Per-position log loss is the negative log of the probability assigned to the played move under the softmax-over-legal-candidates distribution at that position; lower is better. The first three metrics treat the model as a ranker over legal moves, which matches the training objective described in Section 4; log loss measures how concentrated the predicted distribution is on the played move and is comparable to the per-position softmax log loss reported during the LambdaRank-vs-binary comparison in Section 4.

The production move model reaches 61.11% top-1 / 87.77% top-3 / MRR 0.754 / per-position log loss 1.122 on the full 47k-game evaluation set. For reference, an unconditional uniform distribution over legal moves gives log loss ≈ 3.56, and a single-feature baseline using only the upstream policy network (described in Section 4) gives 1.535; the model's distribution is meaningfully concentrated relative to either. Per-slice numbers are shown in Table 3 (slice definitions in Appendix A).

Table 3. Per-slice move-prediction metrics on the 47k-game evaluation set. Higher is better for top-1, top-3, MRR; lower is better for log loss. Slice definitions in Appendix A.

slice	n_positions	top-1	top-3	MRR	log loss
overall	4,122,579	0.6111	0.8777	0.754	1.122
opening	1,435,773	0.6447	0.9094	0.782	0.991
middlegame	1,775,048	0.5926	0.8563	0.737	1.206
endgame	911,761	0.5944	0.8692	0.742	1.163
near-end-of-game	516,608	0.6283	0.8770	0.762	1.111

A natural baseline for this task is "just predict the engine's best move." On the same evaluation set, this baseline reaches 55.47% top-1 / 81.39% top-3 (see Figure 2). The learned model is 5.64 pp better in top-1 overall, with the largest gap — +7.22 pp — in the opening.

Although the move model is trained as a ranker rather than as a probability calibrator, the softmax-over-candidates probabilities turn out to be approximately well-calibrated, particularly at extreme confidence levels. Per-candidate ECE (computed by treating each candidate row as a binary instance — predicted probability is the softmax mass on that candidate, outcome is 1 if it was the played move) on candidates with predicted probability ≥ 1% (matching the deployment display threshold) is 0.009; top-1 confidence ECE — comparing the predicted probability of the model's most-likely candidate to whether that candidate was actually played — is 0.026. The model is well-calibrated when it strongly endorses a move (predicting 98% for clear forced moves that are played 97% of the time) and well-calibrated for displayed low-probability candidates, and is moderately over-confident in the mid-range (predicting 70% likelihood for moves that are played around 65% of the time). This is a non-trivial empirical property of the trained model rather than a property of the LambdaRank objective.

Table 4. Selected per-candidate reliability bins for the move model on the 47k-game evaluation set, restricted to candidates with predicted probability ≥ 1%. Predicted-probability bins span the full [0, 1] range; six representative bins are shown. The

first bin's lower edge is truncated by the ≥1% filter. The model is well-calibrated near both extremes and moderately over-confident in the 0.6–0.8 range.

predicted prob	n	avg pred	empirical	gap
[0.01, 0.07]	11,784,813	0.029	0.031	−0.002
[0.13, 0.20]	1,718,895	0.163	0.177	−0.014
[0.40, 0.47]	456,401	0.432	0.418	+0.014
[0.60, 0.67]	343,485	0.633	0.585	+0.049
[0.73, 0.80]	320,419	0.767	0.719	+0.048
[0.93, 1.00]	599,261	0.979	0.975	+0.004

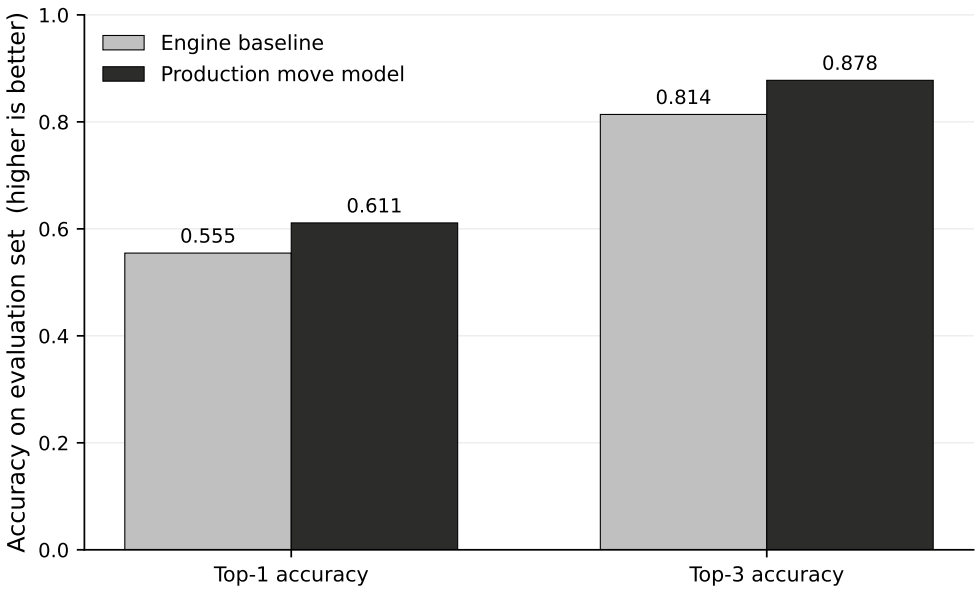

Figure 2: Move-prediction accuracy on the 47k-game evaluation holdout. Top-1 accuracy is the fraction of positions where the played move ranks first; top-3 is the fraction where the played move ranks in the top three. The engine-best-move baseline ranks moves by Stockfish's preference and reports the top-1 and top-3 sets accordingly. Higher accuracy is better.

3.3 Scaling behavior

Both models were trained at seven nested training-set sizes from 5k to 266k games (each a subset of the next), with every other setting held fixed within each sweep, and evaluated on the 47k-game holdout. Figure 3 shows the curves together. For the result model NE drops from 0.760 at 5k games to 0.744 at 266k games; for the move model top-1 accuracy rises from 60.77% to 61.05% over the same range. Both curves are monotonically improving (or flat) across every point. The marginal gains from additional data shrink sharply past about 100k games: the 266k endpoint buys roughly 0.001 NE on the result model and is essentially flat from 186k onward on the move model. The move-model curve is notably flat overall — top-1 moves only 0.28 pp

from 5k to 266k games. Within the current feature set and model capacity, additional training data is unlikely to substantially improve either metric; capacity or features are the more likely next bottleneck.

The two sweeps were configured differently. The result-model sweep used the same hyperparameters at every scale point — those tuned at full data for the production model — so the 266k endpoint matches the production overall NE. The move-model sweep was run during development, before the clean out-of-fold policy-feature rebuild and with a slightly leaner feature set and default hyperparameters; the production model's clean 266k top-1 of 61.11% (Section 3.2) closely matches the sweep endpoint of 61.05%, so the qualitative scaling conclusion is unchanged.

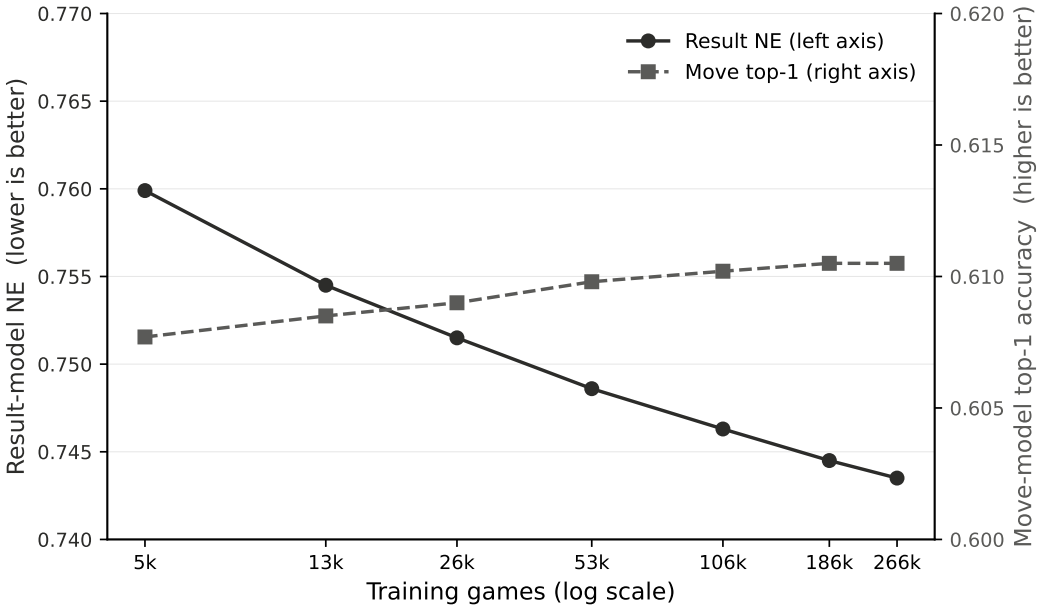

Figure 3: Data-scaling behavior on the 47k-game evaluation holdout. Both models were trained at seven nested training-set sizes from 5k to 266k games. Left axis: result-model overall NE (rating-gap-unaware), where lower is better. Right axis: move-model top-1 accuracy, where higher is better. See Section 3.3 for sweep configuration details.

3.4 Inference speed

Both models ship in two tiers that differ in how rich a feature set each position receives. On a single CPU core the fast tier evaluates a position in around 200 ms and the deep tier in around 1 s; the bulk of each budget is the engine pre-pass, not the tree. Live analysis of ongoing games requires per-position latency below the typical move-arrival rate, which both tiers comfortably achieve. The remainder of this paper reports deep-tier numbers.

3.5 Example

Figure 4 illustrates the result model's calibrated probability series in deployment. The probability series is a richer summary of game state than a single centipawn evaluation: it captures uncertainty around critical moves and resolves cleanly toward the realized outcome as the game closes.

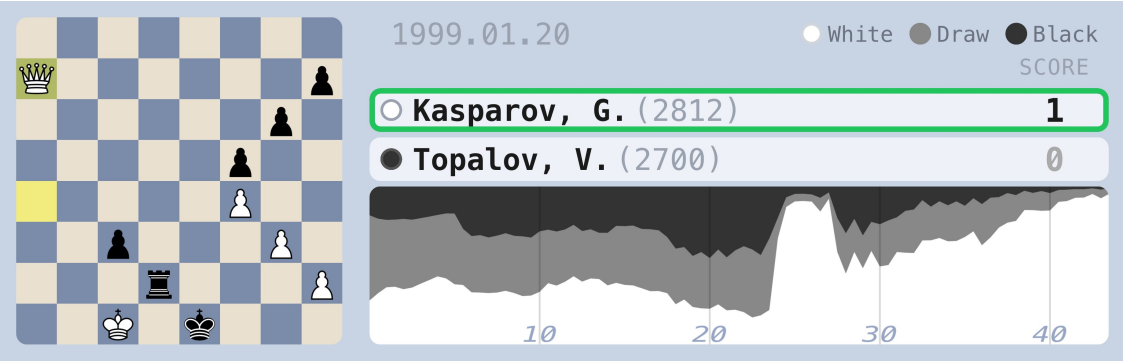

Figure 4: Kasparov–Topalov, Wijk aan Zee 1999, displayed in the chessds.com game-card style: final position on the left and the result model's per-ply win/draw/loss probabilities on the right. The probability ribbon stacks white-win (white), draw (gray), and black-win (black) for each ply; the vertical extent at any ply sums to 100%. Game replay at chessds.com/game/8a38d247add4.

4. Methods

Both models are LightGBM gradient-boosted decision-tree ensembles (Ke et al. 2017). The result model is a three-class softmax over win/draw/loss; the move model is trained with the LambdaRank objective (Burges 2010), with each (game, ply) treated as a query group — the model ranks the legal moves at that position against each other — and the played move as the single relevant candidate. Ranking is a better fit than binary classification for this task because the signal is fundamentally relative — which of the legal moves is most likely to be played — rather than an absolute probability per candidate. On otherwise matched configurations (same features, same data, same hyperparameters), switching from binary classification to LambdaRank reduced per-position softmax log loss from approximately 2.95 to 1.24 — a roughly 2.4× improvement — while leaving top-1, top-3, and MRR unchanged within noise. I attribute this to the binary classifier producing bounded per-candidate probabilities that flatten into a poorly-calibrated distribution under softmax, whereas LambdaRank's unbounded pairwise-loss-trained scores admit sharper, better-aligned post-softmax distributions. Per-position scores are turned into probabilities by a softmax over the legal candidates at inference time.

Inputs to both models are position-level feature vectors: approximately 140 features for the result model and approximately 70 features for the move model. The features draw on Stockfish 18 evaluations of the current position and its candidate moves, the move sequence leading to the current position, properties of the position itself, and the players' Elo ratings (with the rating-gap-aware and rating-gap-unaware variants as described in Section 1). Among these features, a small upstream policy network —

a ~1.2M-parameter ResNet trained on the same TWIC pool of 2400+ classical games with cross-entropy on the played move — produces policy-derived features that feed the move model directly, plus a smaller set of position-level summaries that feed the result model. The policy network was trained on the 266k-game training pool only; the 47k-game evaluation set was held out from policy training as well as from LightGBM training. To avoid in-sample policy features during LightGBM training, the policy network was trained with 5-fold cross-validation across the training pool, and out-of-fold policy features were used to fit the LightGBMs. A separate policy network trained on the full 266k pool generates the features used at evaluation and inference time.

The move model is genuinely hybrid: policy-derived features and Stockfish-derived features each account for roughly half of total split-gain (around 45% and 53% respectively), with under 2% from rating, board, material, and history features. Split-gain is a feature-importance diagnostic during training rather than an ablation; the relative magnitudes are indicative, not a precise measurement of contribution. The result model is much less policy-dependent: policy-derived features account for 4.3% of total split gain in the rating-gap-aware variant and 4.8% in the rating-gap-unaware variant. Further feature-level details and specific engine-search settings are not disclosed in this report; this is a deployed-system tech report rather than a fully reproducible benchmark, and the trade-off is deliberate. Hyperparameters for both models were selected by Optuna (Akiba et al. 2019) using the TPE sampler (Bergstra et al. 2011) against an isolated validation set carved out of the training pool, so that the 47k-game evaluation set was untouched during tuning. All metrics in Section 3 are on that untouched holdout. Time-on-clock is not used as a feature; the models see only the position and game history, not the time situation. Stockfish instance configuration is managed consistently across training and inference; engine output is sensitive to internal state in ways that matter for feature reproducibility.

Inference is sequential within each game: positions are fed through the pipeline in order so that history-derived features describe the actual game history up to the current ply. The models are less accurate on isolated positions presented without their preceding game history.

5. Discussion

I am not aware of published numbers on the elite-classical slice that exceed those reported in Sections 3.1–3.2 on either task. The novelty is in the feature pipeline and the calibration discipline rather than in the model architecture: a boosted-tree ensemble fed with engine evaluations of the main line and candidate moves, trained with task-appropriate objectives (cross-entropy for outcomes, LambdaRank for moves), produces well-calibrated probabilities and strong human-move ranking at this level. The feature engineering benefited from chess expertise — I am rated at USCF Expert class — which guided feature design.

Published work on human-move prediction has largely targeted broader rating ranges and used different inputs. The Maia family (McIlroy-Young et al. 2020) is the canonical example: the original Maia trained nine separate networks targeting Elo bands from 1100 to 1900, learning from move sequences in Lichess games rather than from

engine features, and reporting approximately 52% top-1 accuracy at the 1900 band. Maia-2 (Tang et al. 2024) generalizes this to a single skill-conditional model. Jacob et al. (Jacob et al. 2022) augment Maia-style policy/value models with KL-regularized search to better replicate strong play. ALLIE (Zhang et al. 2025) extends this line up to grandmaster level (2500 Elo) using a decoder-only Transformer trained on Lichess game sequences with adaptive search at inference, and reports state-of-the-art human move-matching across skill levels relative to Maia. In a separate line, Ruoss et al. (Ruoss et al. 2024) show that a transformer distilled from Stockfish reaches grandmaster-level play strength without search. These approaches train end-to-end on human games (or engine play) and operate without separate engine input. The models reported here take a hybrid approach: a small policy network trained on human games provides policy-derived features, Stockfish evaluations provide engine-aware features, and additional position-, game-, and history-level features round out the inputs. All are combined by gradient-boosted trees with task-appropriate objectives. They also target a different population (2400+ classical play, rather than the broader Lichess distribution that includes substantial blitz and rapid play), and I am not aware of published human-move-prediction numbers on elite classical chess that exceed those reported in Section 3.2.

For calibrated win/draw/loss probabilities the prior-art comparison is thinner. Some prior work trains models that produce value or probability outputs — notably ALLIE (Zhang et al. 2025) and engine-distillation work such as Ruoss et al. (Ruoss et al. 2024) — but I am not aware of published reports that include explicit calibration metrics on those outputs at scale. The result model's expected calibration error of around 0.002 (RU) and 0.003 (RA) on a 4.17M-position held-out set is, to my knowledge, unmatched in the published literature on chess outcome prediction.

The rating-gap-unaware variant is the default on chessds.com because it answers a question I think is more useful for commentary: how good or bad is this position, independent of who is holding the pieces? The rating-gap-aware variant, which also sees the rating gap between the two players, reaches lower NE overall because it has strictly more information, but it answers a different question: given these two players, what is the likely outcome? Both variants are reported in this paper, and the rating-gap-aware variant is planned for future availability on chessds.com.

Several limitations are worth flagging. First, the models are trained on 2400+ classical games only, so accuracy on faster time controls or weaker players is not the focus and should not be assumed to match. Second, the near-end-of-game slice shows meaningfully higher calibration error than other slices, driven by the draw class and concentrated in short, GM-agreed draws (see Section 3.1); the model has no position-derived signal for predicting agreement to draw. Third, inference is sequential — the models require the move history leading to a given position (see Section 4). Fourth, the paper does not include a head-to-head comparison with prior human-move-prediction models on the elite-classical evaluation set, since neither Maia nor ALLIE was trained or evaluated on this slice; such a comparison would require running those models on the chessds evaluation set and is left to future work.

Acknowledgments

The author thanks Mark Crowther for maintaining The Week in Chess, which provided the training data for this work. Only PGN move sequences and player ratings were used; no editorial or curatorial content from TWIC is reproduced or relied on.

References

Akiba, Takuya, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. "Optuna: A Next-Generation Hyperparameter Optimization Framework." In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Bergstra, James, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. "Algorithms for Hyper-Parameter Optimization." In *Advances in Neural Information Processing Systems*.

Burges, Christopher J. C. 2010. "From RankNet to LambdaRank to LambdaMART: An Overview." MSR-TR-2010-82. Microsoft Research.

Crowther, Mark. n.d. "The Week in Chess." https://theweekinchess.com.

Elo, Arpad E. 1978. *The Rating of Chessplayers, Past and Present*. Arco Publishing.

Guo, Chuan, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. "On Calibration of Modern Neural Networks." In *Proceedings of the 34th International Conference on Machine Learning*.

Jacob, Athul Paul, David J. Wu, Gabriele Farina, Adam Lerer, Hengyuan Hu, Anton Bakhtin, Jacob Andreas, and Noam Brown. 2022. "Modeling Strong and Human-Like Gameplay with KL-Regularized Search." In *Proceedings of the 39th International Conference on Machine Learning*.

Ke, Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree." In *Advances in Neural Information Processing Systems*.

McIlroy-Young, Reid, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. 2020. "Aligning Superhuman AI with Human Behavior: Chess as a Model System." In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1677–87.

Naeini, Mahdi Pakdaman, Gregory F. Cooper, and Milos Hauskrecht. 2015. "Obtaining Well Calibrated Probabilities Using Bayesian Binning." In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Ruoss, Anian, Grégoire Delétang, Sourabh Medapati, Jordi Grau-Moya, Li Kevin Wenliang, Elliot Catt, John Reid, and Tim Genewein. 2024. "Grandmaster-Level Chess Without Search." In *Advances in Neural Information Processing Systems*. https://arxiv.org/abs/2402.04494.

Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, et al. 2017. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm." *arXiv Preprint arXiv:1712.01815*.

Tang, Zhenwei, Difan Jiao, Reid McIlroy-Young, Jon Kleinberg, Siddhartha Sen, and Ashton Anderson. 2024. "Maia-2: A Unified Model for Human-AI Alignment in Chess." In *Advances in Neural Information Processing Systems*. https://arxiv.org/abs/2409.20553.

The Lc0 Team. n.d. “Leela Chess Zero.” https://lczero.org.
The Stockfish Developers. n.d. “Stockfish: Open-Source Chess Engine.” https://stockfishchess.org.
Voorhees, Ellen M. 1999. “The TREC-8 Question Answering Track Report.” In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*.
Zhang, Yiming, Athul Paul Jacob, Vivian Lai, Daniel Fried, and Daphne Ippolito. 2025. “Human-Aligned Chess with a Bit of Search.” In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/2410.03893.

Appendix A — Slice definitions

The same slice boundaries are used for both the result task (Table 1) and the move task (Table 3).

Opening. Positions at ply ≤ 30.

Endgame. A position is an endgame if either: (i) no queens are on the board, and total non-king material is at most 30; or (ii) at least one queen is on the board, and no rooks, bishops, or knights are on the board (i.e., the only pieces are queens, pawns, and kings). Material is summed using standard piece values (Q=9, R=5, B=3, N=3, P=1; full board = 78). Examples: K+P vs K+P (any pawn count, total material ≤ 30) is an endgame by (i); K+Q+pawns vs K+Q+pawns is an endgame by (ii); K+Q+R vs K+Q is not an endgame.

Middlegame. All non-opening, non-endgame positions.

Near-end-of-game. The final eleven positions of each game.