CASE: COMPUTERIZED ADAPTIVE STRENGTH ESTIMATOR FOR CHESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Accurately estimating a player's strength remains a challenging problem for adapting game AI to players of different strengths. Most existing approaches require players to complete or provide approximately 20–30 games before obtaining a reliable estimate, making the assessment time-consuming. Inspired by computerized adaptive testing (CAT), which sequentially selects informative questions according to the current ability estimate, we propose *Computerized Adaptive Strength Estimator* (CASE), a novel framework that estimates playing strength through adaptively selected board positions. Specifically, CASE uses an estimator to update the player's strength from the question-response history and a selector to choose the next question based on the current history and strength estimate. Experiments on chess show that CASE achieves over 80% rank accuracy with only 34 questions, whereas none of the compared baselines reaches this accuracy even after 200 questions. Compared with the prior game-based strength estimation method, CASE requires approximately 27 times fewer player responses to reach the same accuracy. Further analyses show that the selector identifies informative positions and adapts its question preferences according to the player's current strength estimate. These results demonstrate that accurate strength estimation can be achieved through a short adaptive assessment without requiring multiple complete games.

1 INTRODUCTION

Strength estimation is important for adapting game AI to players of different strengths. For example, an AI can adjust its playing strength to provide an appropriate opponent or select training material according to the player's current level (Hunicke & Chapman, 2004; Paulsen & Fürnkranz, 2010; Silva et al., 2015; Moon & Seo, 2020; Wu et al., 2019). However, such adaptation first requires an accurate estimate of the player's strength. Strength estimation in board games has therefore been studied extensively in recent years (Liu et al., 2020; Chen et al., 2024; Kuboki et al., 2025). Existing approaches generally rely on either direct interaction with the player or previously played game records. One line of work estimates strength by asking the player to play against an AI and updating the estimate based on the resulting game outcomes (Liu et al., 2020), while another infers strength directly from the player's moves in their historical games (Moudřík & Neruda, 2016; Chen et al., 2024; Kuboki et al., 2025). Despite their differences, both approaches require approximately 20–30 games to obtain an accurate estimate (Liu et al., 2020; Chen et al., 2024). In practice, asking a player to first play or provide 20–30 games before obtaining a reliable strength estimate is time-consuming and impractical. This raises a challenging research question: *can we accurately estimate a player's strength through limited interaction, without requiring the player to complete or provide full games?*

A natural way to reduce interactions is to reconsider what information is actually needed for strength estimation. For example, in chess, some positions involve critical decisions that can show differences between players of different strengths, whereas others are straightforward or nearly forced (e.g., when a player is in check and only one reasonable response is available) and provide little information. Thus, an efficient assessment should focus on asking players to make moves in a small number of informative positions rather than asking them to play several complete games. This idea is closely related to computerized adaptive testing (CAT) (van der Linden & Glas, 2000; Green et al., 1984; Weiss & Kingsbury, 1984), which is widely used in educational and psychological assessment to estimate ability efficiently. In CAT, questions are asked one at a time, and each response is used to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

update the current ability estimate and select the next question according to that estimate, allowing the assessment to focus on questions that are most informative for the test taker. However, conventional CAT relies on a calibrated question bank, where the response behavior for each question must be estimated in advance. As a result, its question selection is limited to previously calibrated questions, making it difficult to directly exploit new questions without additional calibration.

To address this limitation, we propose *Computerized Adaptive Strength Estimator* (CASE), a novel assessment framework for board games that estimates playing strength through a sequence of adaptively selected questions. CASE consists of two components: an *estimator*, which predicts the player's strength from the question-response history, and a *selector*, which chooses the next question based on both the current history and the estimated strength. After each response, CASE updates its strength estimate and uses the updated information to select the next question, allowing the assessment to adapt continuously to the player. Unlike conventional CAT, CASE does not require each question to be calibrated in advance. Instead, it learns to evaluate the informativeness of candidate questions and can therefore select from previously unseen questions. In this way, CASE jointly learns how to estimate a player's strength and which question should be asked next.

We evaluate CASE on chess, where it achieves over 80% rank accuracy with only 34 questions, whereas none of the compared baselines reaches this accuracy even after 200 questions. Importantly, the best prior game-based strength estimation method requires approximately 910 player responses from 26 games to exceed 80% accuracy, whereas CASE reaches the same accuracy with only 34 questions, requiring approximately 27 times fewer player responses. After 200 questions, CASE reaches 89.3% accuracy, compared with only 63.5% for the best baseline. Further analyses show that our selector effectively identifies informative positions, while its question preferences change systematically with the player's current strength estimate. Together, these results show that CASE can estimate playing strength accurately and efficiently through a short adaptive assessment, without requiring players to complete or provide multiple full games.

2 Background

2.1 Strength Estimation in Board Games

Strength estimation in board games has been studied extensively in recent years, with existing approaches mainly estimating a player's strength either by directly playing games against an AI or by using previously played game records. For the former, Liu et al. (2020) propose an approach to dynamically adjust the strength of an AI program according to the player's performance and estimate the player's strength based on the resulting win rate, requiring approximately 20 games to obtain an accurate estimate. For the latter, early work explores estimating a player's strength from individual position-move pairs (Moudřík & Neruda, 2016). Chen et al. (2024) propose *Strength Estimation* (SE), a learning-based approach that uses a neural network to predict a scalar strength score for each position-move pair. The network is trained using a Bradley–Terry model (Huang et al., 2006) by comparing moves from players of different ranks, such that moves played by stronger players are encouraged to receive higher strength scores. During evaluation, SE computes the strength score for each observed move and aggregates the scores across multiple positions to obtain an overall strength score. The resulting score is then compared with precomputed reference scores for different ranks to determine the player's estimated rank, achieving over 80% accuracy using 26 games in chess. Kuboki et al. (2025) further improve SE by incorporating additional information from policies trained at different skill levels. Despite these advances, accurate strength estimation still requires players to either complete multiple games for assessment or provide sufficient existing game records.

2.2 Computerized Adaptive Testing

Computerized Adaptive Testing (CAT) has been widely used in educational and psychological assessment to efficiently estimate a person's ability (van der Linden & Glas, 2000; Green et al., 1984; Weiss & Kingsbury, 1984). Unlike a fixed test that presents the same set of questions to every test taker, CAT adaptively selects questions according to the test taker's current ability estimate, allowing it to obtain an accurate estimate with fewer questions. Before testing, CAT first constructs a response model using a set of questions, commonly referred to as an *item bank*, and responses col-

lected from test takers of different ability levels. Specifically, given a question q and an ability level r, the response model estimates the probability $P(a \mid q, r)$ of observing each possible response a. During testing, after observing a response a to question q, CAT assigns higher likelihood to ability levels under which the observed response has a larger $P(a \mid q, r)$. The same response model can also be used to evaluate how informative each candidate question is for distinguishing among the current possible ability levels. CAT then selects the most informative question and repeats this process throughout the assessment. A common formulation uses a binary response model, where a indicates whether the response is correct or incorrect, while the nominal response model (Bock, 1972) allows multiple possible responses to each question and models the probability of each response separately.

3 METHOD

3.1 COMPUTERIZED ADAPTIVE STRENGTH ESTIMATOR (CASE)

We propose the *Computerized Adaptive Strength Estimator* (CASE), an adaptive system that assesses a player's strength by asking one question at a time, using the player's response to update its estimate and determine which question to ask next. Unlike existing strength estimation methods that require players to complete full games or rely on players' existing game records, CASE estimates playing strength through a sequence of adaptively selected questions. This allows CASE to estimate playing strength while requiring only limited interaction from the player.

Given a question dataset $\mathcal{D}$ and a set of ranks $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$, each rank represents a predefined range of playing strength. For example, in chess, $\mathcal{D}$ can consist of board positions collected from game records, where each position serves as a question q. Each rank r_i can correspond to an Elo-rating (Elo, 1978) interval, such as 2400–2599 for r_1 and 2200–2399 for r_2. For simplicity, we index the ranks in descending order of strength in this paper, with r_1 denoting the strongest rank and r_n the weakest.

For a player with an unknown rank $r^* \in \mathcal{R}$, CASE starts by selecting a question q_1 from $\mathcal{D}$ and receiving the player's response a_1, where a_1 is the next move chosen by the player from the board position q_1. From the first question-response pair (q_1, a_1), CASE produces an initial strength estimate and selects the next question q_2 accordingly. Thereafter, at each step t, CASE uses the current question-response history $H_t = ((q_1, a_1), (q_2, a_2), \ldots, (q_t, a_t))$ to update its strength estimate and select q_{t+1}. This process can continue iteratively, with a strength estimate available at every step, allowing the assessment to stop at any point.

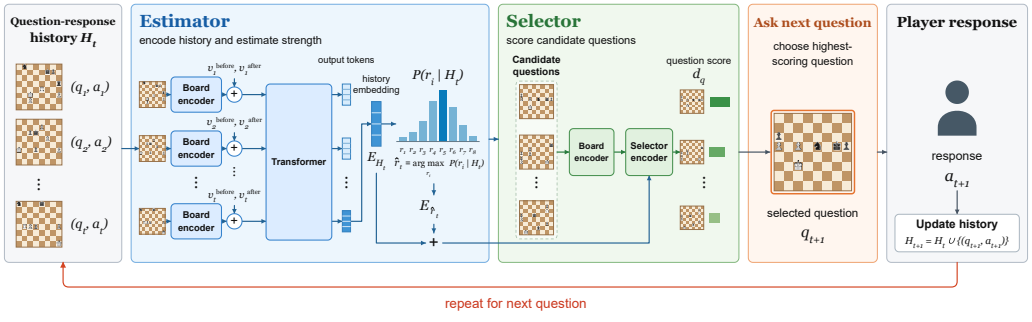


Figure 1: Overview of CASE. At each step, the estimator encodes the question-response history H_t into the history embedding E_{H_t} and predicts the rank distribution $P(r_i \mid H_t)$; the selector scores candidate questions conditioned on E_{H_t} and the rank embedding $E_{\hat{r}_t}$, and the highest-scoring question is asked. The player's response extends the history, and the process repeats.

CASE implements this adaptive process with two components: an *estimator* and a *selector*, as illustrated in Figure 1. We describe the design of each component in detail below.

Estimator. Given the current question-response history H_t, the *estimator* predicts the player's rank as a probability distribution $P(r_i \mid H_t)$ over possible ranks. For each question-response pair (q_i, a_i), a board encoder first extracts a feature representation from the question q_i and the player's

move a_i. To better distinguish moves of different quality, the estimator additionally uses two value estimates that evaluate the position before and after the move, denoted by v_i^{before} and v_i^{after}, respectively. The two value estimates are then combined with the feature representation to form a token for each question-response pair. All tokens derived from H_t are then processed by a Transformer to aggregate information across the questions answered so far, with the output at the final position used as the history embedding E_{H_t}. Finally, the history embedding E_{H_t} is used to obtain $P(r_i \mid H_t)$, and the rank with the highest probability, $\hat{r}_t = \arg\max_{r_i} P(r_i \mid H_t)$, is taken as the current strength estimate.

Selector. At step t with the current assessment, the *selector* evaluates a set of candidate questions sampled from the question dataset $\mathcal{D}$ and aims to select a question q_{t+1} that is most informative for estimating the player's strength. For each candidate question q, the selector first extracts a feature representation using the same board encoder as the estimator. Since the candidate question has not yet been answered, its action input is left empty. Next, to incorporate both the current question-response history and strength estimate into question selection, we combine the history embedding E_{H_t} with an embedding of the estimated rank. Specifically, we take the current estimated rank $\hat{r}_t$ from the estimator and encode it as $E_{\hat{r}_t}$, which is added to the history embedding E_{H_t}. Although the estimated rank is already implicit in E_{H_t}, we include it explicitly to provide the selector with direct access to the estimator's current rank estimate. The resulting embedding is then concatenated with the feature representation of each candidate question and passed through the selector encoder to produce a question embedding $E_{q,t}$. Since $E_{q,t}$ depends on both the current question-response history and estimated rank, the same question can be represented differently as the assessment progresses, allowing the selector to interpret each question based on the player's previous responses and current rank estimate. To quantify the informativeness of each candidate question, we introduce a zero-vector anchor $E_{\varnothing}$ as a fixed reference point in the question-embedding space. The informativeness of a question is then represented by the distance between its embedding $E_{q,t}$ and the anchor $E_{\varnothing}$. Thus, more informative questions have embeddings farther from the anchor, while less informative questions remain closer to it. Finally, the selector selects the candidate question with the largest distance from the anchor:

$$q_{t+1} = \arg\max_{q} \mathrm{MSE}(E_{q,t}, E_{\varnothing}). \tag{1}$$

In summary, CASE alternates between the estimator and selector throughout the assessment. For a player of unknown strength, the selector first uses an empty history embedding together with an *unknown-rank* embedding to select the first question q_1. After the player responds to question q_t with a_t, the estimator computes the history embedding E_{H_t} and the corresponding estimated rank $\hat{r}_t$ based on H_t. The selector then uses E_{H_t} and $E_{\hat{r}_t}$ to evaluate the candidate questions and select the next question q_{t+1}. After receiving the response a_{t+1}, the new question-response pair is incorporated into H_{t+1}, and the process repeats until the assessment ends.

3.2 Training CASE

Training CASE directly from scratch is challenging because the estimator and the selector are mutually dependent. The selector relies on the estimator to provide a representation of the player's question-response history together with the current strength estimate, while the estimator learns from the sequence of questions chosen by the selector and the value estimates of the player's responses. Training both components jointly from the beginning is therefore difficult, since neither component can provide useful information to the other at the beginning of training. To address this issue, we train CASE in three stages: (1) pre-training the board encoder and value estimator to acquire general chess knowledge, (2) pre-training the estimator to distinguish players of different strengths, and (3) jointly training CASE through simulated adaptive assessments.

Stage 1: Pre-training the board encoder. We first pre-train the board encoder on collected game records using an AlphaZero-style (Silver et al., 2018) policy-value network. For each board position q, the move a played in the game and the final game outcome z are used to train the policy and value heads, respectively. The policy head outputs a probability distribution $\pi_\theta(\cdot \mid q)$ over all possible moves, while the value head predicts the outcome $v_\theta(q)$ of position q. The training objective is

$$\mathcal{L}_{\text{stage1}} = -\log \pi_\theta(a \mid q) + \left(v_\theta(q) - z\right)^2. \tag{2}$$

The policy head provides an auxiliary training objective to help the board encoder learn general chess representations, while the value head is retained to provide v^{before} and v^{after} in the subsequent stages. After pre-training, the latent representation before the policy and value heads is used as the board feature in CASE.

Stage 2: Pre-training the estimator. Next, we pre-train the estimator to distinguish players of different strengths. Specifically, we randomly sample t questions and ask a player from each rank to answer the same set of questions. This produces a question-response history for each rank, allowing the estimator to identify how players of different strengths respond differently to the same questions. The estimator is then trained to predict the corresponding rank r^* from the question-response history H_t using a cross-entropy loss:

$$\mathcal{L}_{\text{estimator}} = -\log P(r^* \mid H_t). \tag{3}$$

Note that throughout this stage, only the estimator is trained, while the pre-trained policy and value heads are kept fixed.

Stage 3: Joint training of CASE. After pre-training the board encoder and estimator, we jointly train CASE through simulated adaptive assessments. For each assessment, we sample a player with ground-truth rank $r^* \in \mathcal{R}$ and let CASE interact with the player for T questions. At each step t, the selector chooses the next question q_{t+1} based on the current history H_t, and the simulated player provides the corresponding response a_{t+1}. This produces question-response histories in which the questions are adaptively selected by the selector rather than randomly sampled as in Stage 2. After completing an assessment, we use its intermediate question-response histories for model updates. Specifically, we sample an intermediate step $t < T$ and extract the corresponding history H_t for model updates.

For the estimator, we use the same loss defined in equation 3 to predict r^* from the history H_t at any intermediate step t. Unlike Stage 2, where the questions are randomly sampled, the estimator is now trained on questions adaptively selected by the selector, allowing it to adapt to the question distribution encountered during actual assessments. For the selector, unlike the estimator, there is no direct supervision indicating how informative each candidate question is. We therefore simulate asking each candidate question q and evaluate how much the resulting response improves the estimator's confidence in the ground-truth rank. Specifically, for each candidate question q, we append the player's simulated response to the current history H_t to obtain a hypothetical next history $\tilde{H}_{t+1}$. Then, we define the confidence gain of q as

$$\Delta(q) = \log P(r^* \mid \tilde{H}_{t+1}) - \log P(r^* \mid H_t). \tag{4}$$

Thus, a positive $\Delta(q)$ indicates that asking q increases the estimator's confidence in the correct rank, whereas a non-positive $\Delta(q)$ indicates that the question does not improve the current estimate. We use this confidence gain to train the question-embedding space introduced in Section 3.1. Specifically, candidate questions with non-positive confidence gain are encouraged to have small d_q, while questions with positive confidence gain are ordered such that larger gains correspond to larger d_q, where $d_q = \text{MSE}(E_{q,t}, E_{\varnothing})$ denotes the distance from the question embedding to the anchor.

For candidate questions with non-positive confidence gain, we minimize their distances to the anchor:

$$\mathcal{L}_{\text{neg}} = \frac{1}{|\{q : \Delta(q) \le 0\}|} \sum_{q:\Delta(q)\le 0} d_q. \tag{5}$$

For candidate questions with positive confidence gain, we first rank them according to $\Delta(q)$ and select the top-K questions for the ranking objective. Suppose $q_1, \ldots, q_K$ are these questions ordered such that $\Delta(q_1) \ge \Delta(q_2) \ge \cdots \ge \Delta(q_K) > 0$. We then apply a ListMLE ranking loss (Xia et al., 2008) to encourage the same ordering in their distances:

$$\mathcal{L}_{\text{pos}} = -\frac{1}{K} \sum_{i=1}^{K} \log \frac{\exp(d_{q_i})}{\sum_{j=i}^{K} \exp(d_{q_j})}, \tag{6}$$

and the overall selector loss is $\mathcal{L}_{\text{selector}} = \mathcal{L}_{\text{neg}} + \mathcal{L}_{\text{pos}}$.

During this stage, all components of CASE are jointly updated. In particular, we continue optimizing the Stage 1 policy-value objective together with the estimator and selector objectives, allowing the

board encoder to continue refining its chess representations during joint training. The overall training loss of CASE is therefore

$$\mathcal{L}_{\text{CASE}} = \mathcal{L}_{\text{stage1}} + \mathcal{L}_{\text{estimator}} + \mathcal{L}_{\text{selector}}. \tag{7}$$

Overall, the three-stage training procedure first learns general chess representations, then establishes an initial strength estimator, and finally jointly optimizes the full model for adaptive strength estimation.

4 EXPERIMENT

4.1 EXPERIMENT SETUP

We implement CASE based on MiniZero (Wu et al., 2025), an open-source board game training framework, and evaluate it on chess using game records from the open Lichess database (Lichess, 2026). We collect blitz games played between January 2022 and February 2023 by players rated from 1000 to 2599 Elo, and divide this range into $n = 8$ rank intervals of 200 Elo points each, with r_1 denoting the strongest rank (2400–2599) and r_8 the weakest (1000–1199). Since the game records contain only the move played by a player of a specific rank, training CASE requires responses from players of different ranks to the same question. In practice, such responses can be obtained either by asking players of different ranks to answer a common set of questions or by using chess models of different strengths as simulated players. In this work, following the approach of Maia (McIlroy-Young et al., 2020), we train several policy models corresponding to the ranks in $\mathcal{R}$ and use them as simulated players.

For training CASE, we train six simulated players for each rank, providing multiple independently trained players at the same rank. Specifically, for each rank, we split 3 million games into six sets of 500k games and train one simulated player on each set using supervised learning to predict the moves played by players in that rank. This results in 6 players/rank $\times$ 8 ranks $= 48$ simulated players in total. Separately, a disjoint set of 800k games per rank is used to construct the training question dataset $\mathcal{D}_{\text{train}}$ for CASE. For each rank, four of the six simulated players are used to train CASE, while the remaining two are reserved for evaluation. For each board position in this dataset, we precompute the responses of the four simulated players used during training, allowing the question-response histories to be constructed efficiently during simulated assessments.

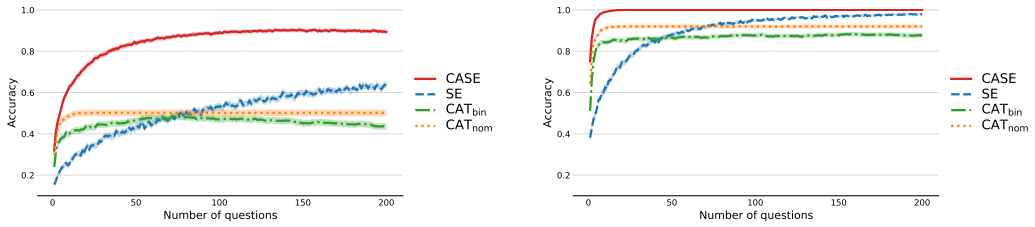
CASE uses a residual board encoder with 3 blocks and 256 hidden channels. The question-response history is aggregated by a Transformer into a 384-dimensional history embedding, and the selector consists of 3 residual blocks. During training, each simulated assessment consists of $T_{\text{train}} = 30$ questions, where each question is selected from a pool of 64 candidates sampled from $\mathcal{D}_{\text{train}}$. Further architecture and training details are provided in Appendix A, and ablations of the design choices are reported in Appendix D.

For evaluation, we sample 1.2 million board positions per rank from a separate set of 200k games per rank, resulting in 9.6 million questions in total. These questions form the question dataset $\mathcal{D}_{\text{eval}}$. We evaluate CASE on players not used during training, including the two reserved simulated players and ALLIE (Zhang et al., 2024), a state-of-the-art human-like chess model whose playing strength can be conditioned on a target rank. This results in three evaluation players for each rank and 3 players/rank $\times$ 8 ranks $= 24$ evaluation players in total. For each evaluation player, we conduct 150 independent CASE assessments, each starting from an empty history and consisting of $T_{\text{eval}} = 200$ questions. At each step, we randomly sample 1,024 candidate questions from $\mathcal{D}_{\text{eval}}$, and CASE selects the next question from this candidate pool.

4.2 STRENGTH ESTIMATION PERFORMANCE

We compare CASE with three baselines: SE (Chen et al., 2024), CAT_{bin}, and CAT_{nom}. For a fair comparison, all baselines are trained or calibrated using only the same four simulated players per rank used for training CASE. SE follows the same network architecture and training settings in the original paper. Given a question-response pair, SE predicts a strength score and estimates the player's rank by aggregating the scores over multiple questions. Unlike CASE, SE does not select questions adaptively, so its questions are sampled randomly from $\mathcal{D}_{\text{eval}}$. Following the original SE procedure, SE requires rank-specific reference scores to determine the predicted rank. We therefore

prepare a separate set of 10k games per rank, used only by SE, to compute these reference scores. For CAT_{bin}, it scores each question with Stockfish (Romstad et al., 2026) and treats a move as correct if it is close enough to the best move, while CAT_{nom} treats each legal move as a distinct response; in both cases $P(a \mid q, r)$ is obtained by averaging over the four calibration players. During evaluation, both methods use these calibrated response probabilities to update the rank estimate and select the next question following the CAT procedure described in Section 2.2. More details about these baseline settings are given in Appendix B.



(a) Rank accuracy.

(b) Accuracy within one rank (± 1).

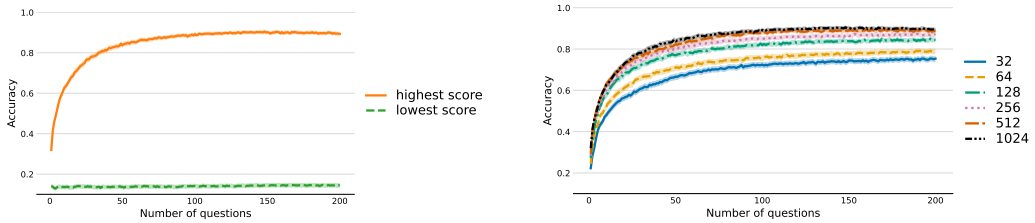
Figure 2: Rank prediction accuracy against the number of questions asked for CASE and the three baselines, evaluated on the two reserved simulated players and ALLIE. The shaded area represents the 95% confidence interval.

Figure 2 presents the strength estimation accuracy of CASE and the three baselines as the number of questions increases. As shown in Figure 2a, CASE substantially outperforms the baselines and achieves 80% accuracy after only 34 questions, whereas none of the baselines reaches this number even after 200 questions. After 200 questions, CASE achieves 89.3% accuracy, compared with 63.5% for SE, 50.1% for CAT_{nom}, and 43.5% for CAT_{bin}. Notably, although CASE is trained with only $T_{\text{train}} = 30$ questions per assessment, its accuracy continues to improve with more questions at evaluation time, increasing from 78.1% at 30 questions to 89.3% at 200 questions. Figure 2b further shows the accuracy when predictions within one rank of the ground-truth rank are considered correct. CASE also outperforms all baselines, reaching 95% after only five questions, 99% after 10 questions, and 100% after 22 questions. These results show that even when CASE does not predict the exact rank, its estimate quickly converges to the correct strength range. Detailed accuracy results for each method, evaluated separately for each player, are provided in Appendix C.1.

It is worth noting that the performance of SE in our evaluation is consistent with that reported in the original study (Chen et al., 2024). In chess, each game usually contains approximately 70 plies on average, corresponding to about 35 plies made by a player being evaluated. Thus, 200 questions correspond to approximately six games in terms of the number of player responses, at which SE achieves 63.5% accuracy. Similarly, the original SE achieves around 60% accuracy with six chess games and requires approximately 26 games to exceed 80% accuracy. Based on the same conversion, 26 games correspond to roughly 910 player responses. In contrast, CASE reaches 80% accuracy with only 34 questions, requiring about 27 times fewer player responses than SE. This substantial reduction allows CASE to estimate a player's strength through a short assessment, without requiring many complete games to be played or accumulated beforehand.

4.3 Effectiveness of Question Selection

To examine whether the performance of CASE benefits from its question selection, we first investigate whether the selector assigns higher scores to questions that are more informative for strength estimation. We compare the standard selection strategy, which chooses the highest-scoring question from the candidate pool, with a reversed strategy that instead chooses the lowest-scoring question. All other components and evaluation settings remain unchanged. As shown in Figure 3a, selecting the highest-scoring questions achieves 89.3% accuracy after 200 questions, whereas selecting the lowest-scoring questions remains close to the chance level of 12.5% ($1/8$ for eight ranks), reaching only 14.4%. Moreover, the accuracy of the reversed selector changes little as more questions are asked. These results show that the selector successfully distinguishes questions that are informative for strength estimation from those that provide little useful information.



(a) Selection by question score.

(b) Different number of candidate questions.

Figure 3: Analysis of the effectiveness of question selection in the CASE selector.

Next, we analyze how the number of candidate questions affects strength estimation. With a larger candidate pool, the selector has more opportunities to identify informative questions. As shown in Figure 3b, the accuracy increases from 75.3% with 32 candidates to 89.3% with 1,024 candidates at the end. Notably, although CASE is trained with a candidate pool of only 64 questions, its accuracy continues to improve when more candidates are provided at evaluation time. This consistent improvement indicates that the selector can effectively exploit a larger candidate pool to identify more informative questions. With 1,024 candidates, the pool is sufficiently large for the selector to identify highly informative board positions, leading to the best performance observed in our experiments. The corresponding accuracy within one rank for both analyses is reported in Appendix C.4.

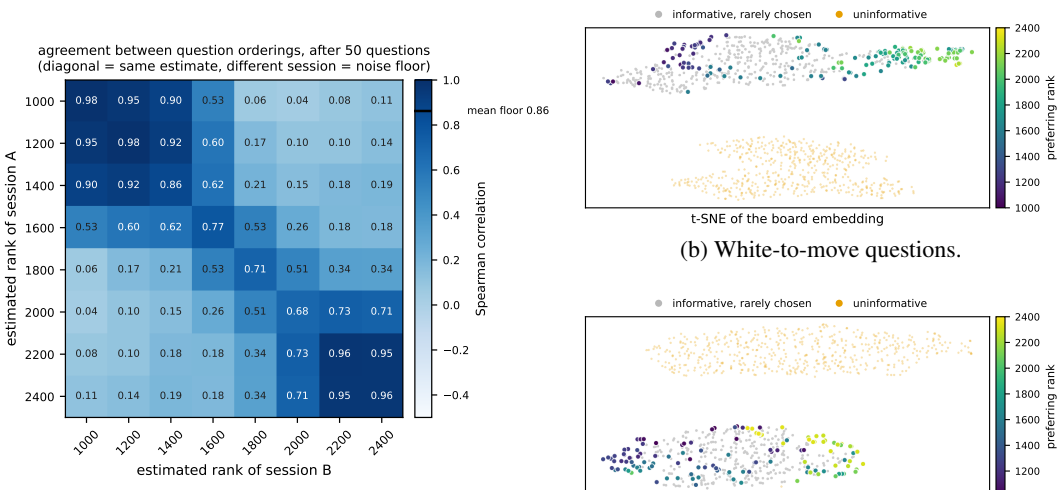
4.4 Analysis of Adaptive Question Selection

The previous subsection has shown that the selector can identify informative questions. However, the selector is designed to use both the current question-response history and the rank estimate produced by the estimator when selecting the next question. In this subsection, we examine how the selector changes its question preferences according to the current strength estimate.

To analyze this behavior, we compare pairs of player ranks and examine whether the selector produces different rankings of the same candidate questions for players of different strengths. Specifically, for two players P_i and P_j from different ranks, we independently conduct a 50-question adaptive assessment using the standard CASE procedure, resulting in different question-response histories and current rank estimates for the two players. For the 51st question, we provide the selector with the same set of candidate questions $\mathcal{Q} = \{q_1, \ldots, q_m\}$ for both players, while retaining their respective histories and rank estimates. While CASE normally selects only the question with the highest score, here we sort all candidate questions according to their scores, producing two rankings π_i and π_j. We then compute the Spearman correlation between π_i and π_j, where a higher correlation indicates greater agreement between the two rankings and thus more similar question preferences for the two players. Further details of the comparison procedure are provided in Appendix C.2.

As shown in Figure 4a, the correlations are highest along the diagonal. For each player rank, the correlation generally decreases as the rank difference increases. This pattern suggests that the informativeness of a question is strength-dependent: a question that is useful for distinguishing players within one strength range may be less informative for another. The selector successfully learns this relationship and adjusts its question priorities according to the current strength estimate.

Next, we further examine whether different strength ranks systematically prefer different types of questions. We first construct two sets of questions based on the selector scores before any player-specific history or rank estimate is available: the 1,000 highest-scoring questions as generally informative questions and 1,000 low-scoring questions as an uninformative reference. For the 1,000 informative questions, we repeatedly sample 200 questions and let the selector rank them for players with different current rank estimates. For each rank, we take the 10 highest-scoring questions as those preferred by the selector for that rank. By repeating this procedure many times, we can identify which ranks tend to prefer each question. In particular, if a question q is consistently ranked highly when the estimator predicts the player to be in rank r, we consider q to be preferred for evaluating players in rank r. Details of the procedure are provided in Appendix C.3.



(a) Ranking agreement.

(b) White-to-move questions.

(c) Black-to-move questions.

Figure 4: Analysis of rank-dependent question preferences. **(a)** Spearman correlation between selector rankings for different current rank estimates. **(b)-(c)** t-SNE visualizations of board representations for White-to-move and Black-to-move questions, respectively.

Figures 4b and 4c show the t-SNE (van der Maaten & Hinton, 2008) visualization of the board representations for White-to-move and Black-to-move questions, respectively. Questions frequently preferred by the selector are colored according to their preferred ranks, while informative questions that are rarely preferred are shown in gray, and the uninformative reference questions are shown in orange. Among the frequently preferred questions, the colors show a clear transition from weaker to stronger ranks, suggesting that the learned board representation captures a gradual structure from questions suited to weaker players to those suited to stronger players. Interestingly, the uninformative reference questions form separate regions, showing that the selector can distinguish positions that are less informative for strength estimation. This further highlights the benefit of adaptive question selection, as the selector only focuses on positions that are informative for the player's current strength rather than relying on all positions encountered in complete games.

5 Discussion

This paper presents *Computerized Adaptive Strength Estimator* (CASE), a novel adaptive assessment framework that estimates playing strength through a sequence of adaptively selected questions rather than requiring players to complete or provide full games. CASE combines an estimator that predicts the player's strength from the question-response history with a selector that chooses the next question according to the current history and strength estimate. Experiments on chess show that CASE achieves over 80% rank accuracy with only 34 questions, requiring approximately 27 times fewer player responses than the previous strength estimation method. Our experiments also show that the selector successfully identifies informative positions and adjusts its question preferences according to the player's current strength estimate.

One limitation is that collecting responses from players of different strengths to the same questions is difficult, so we use simulated players for training and evaluation. Since simulated behavior may not fully match human behavior, future work can evaluate CASE with human players or use other human-like models (Tang et al., 2024; Monroe et al., 2025) to better approximate their responses. Nevertheless, our results demonstrate that CASE can learn effective adaptive strength estimation given sufficient response data across different strength levels. Another interesting direction is to use the informative positions selected by the selector as personalized training material, as these positions may reveal what a player needs to improve to reach a higher level. Finally, rather than selecting from a pre-collected question set, CASE could be extended to generate informative positions (Feng et al., 2025) directly, potentially further reducing the number of questions required for assessment.

AI USE STATEMENT

In this work, we used generative AI tools for code editing/debugging assistance and for improving the readability of the manuscript. We did not use generative AI tools to develop the method, design experiments, process data, or interpret results. All AI-assisted code was reviewed and tested by the authors, and all AI-assisted text was reviewed and edited by the authors. We take responsibility for the final content of this work, including text, claims, code, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses chess game records from the publicly available Lichess database and does not involve newly collected human-subject data. The proposed method estimates playing strength for applications such as adaptive assessment and training, rather than inferring sensitive personal attributes.

REPRODUCIBILITY STATEMENT

We provide the details needed to reproduce our experiments throughout the paper and appendix. The CASE architecture and training objective are described in Section 3, while the dataset construction, train/evaluation/candidate splits, evaluation protocol, and baseline setup are specified in Section 4.1. Appendix A gives the full implementation and training details, including model dimensions, optimization settings, training stages, hardware, and the hyperparameters summarized in Table 1. Upon acceptance, we will publicly release our code, configuration files, and the model checkpoints used in our experiments to facilitate independent reproduction of the reported results.

REFERENCES

R. Darrell Bock. Estimating item parameters and latent ability when responses are scored in two or more nominal categories. *Psychometrika*, 37(1):29–51, March 1972.

Chun Jung Chen, Chung-Chin Shih, and Ti-Rong Wu. Strength Estimation and Human-Like Strength Adjustment in Games. In *The Thirteenth International Conference on Learning Representations*, October 2024.

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

Xidong Feng, Vivek Veeriah, Marcus Chiam, Michael D. Dennis, Federico Barbero, Johan Obando-Ceron, Jiaxin Shi, Satinder Singh, Shaobo Hou, Nenad Tomasev, and Tom Zahavy. Generating Creative Chess Puzzles. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, October 2025.

Bert F. Green, R. Darrell Bock, Lloyd G. Humphreys, Robert L. Linn, and Mark D. Reckase. Technical Guidelines for Assessing Computerized Adaptive Tests. *Journal of Educational Measurement*, 21(4):347–360, 1984.

Tzu-Kuo Huang, Ruby Weng, and Chih-Jen Lin. Generalized Bradley-Terry Models and Multi-Class Probability Estimates. *Journal of Machine Learning Research*, 7:85–115, January 2006.

Robin Hunicke and Vernell Chapman. AI for dynamic difficulty adjustment in games. *Challenges in game artificial intelligence AAAI workshop*, 2, January 2004.

Kyota Kuboki, Tatsuyoshi Ogawa, Chu-Hsuan Hsueh, Shi-Jim Yen, and Kokolo Ikeda. Policies of Multiple Skill Levels for Better Strength Estimation in Games. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 21(1):81–90, November 2025.

Lichess. Lichess. https://database.lichess.org/#standard_games, 2026.

An-Jen Liu, Ti-Rong Wu, I-Chen Wu, Hung Guei, and Ting-Han Wei. Strength Adjustment and Assessment for MCTS-Based Programs [Research Frontier]. *IEEE Computational Intelligence Magazine*, 15(3):60–73, August 2020.

Reid McIlroy-Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. 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, Virtual Event CA USA, August 2020. ACM.

Daniel Monroe, George Eilender, Philip Chalmers, Zhenwei Tang, and Ashton Anderson. Chessformer: A Unified Architecture for Chess Modeling. In *The Fourteenth International Conference on Learning Representations*, October 2025.

Hee-Seung Moon and Jiwon Seo. Dynamic Difficulty Adjustment via Fast User Adaptation. In *Adjunct Publication of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pp. 13–15, October 2020.

Josef Moudřík and Roman Neruda. Determining Player Skill in the Game of Go with Deep Neural Networks. In *Theory and Practice of Natural Computing*, pp. 188–195, Cham, 2016. Springer International Publishing.

Philip Paulsen and Johannes Fürnkranz. A Moderately Successful Attempt to Train Chess Evaluation Functions of Different Strengths. In *Proceedings of the ICML-10 Workshop on Machine Learning and Games*, 2010.

Tord Romstad et al. Stockfish. `https://github.com/official-stockfish/Stockfish/commit/06675f70`, 2026.

Mirna Silva, Victor Silva, and Luiz Chaimowicz. Dynamic Difficulty Adjustment through an Adaptive AI. In *14th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pp. 173–182, November 2015.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, December 2018.

Zhenwei Tang, Difan Jiao, Reid McIlroy-Young, Jon Kleinberg, Siddhartha Sen, and Ashton Anderson. Maia-2: A Unified Model for Human-AI Alignment in Chess. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, November 2024.

Wim J. van der Linden and Cees A. W. Glas (eds.). *Computerized Adaptive Testing: Theory and Practice*. Springer Netherlands, Dordrecht, 2000.

Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6000–6010, Red Hook, NY, USA, December 2017. Curran Associates Inc.

David J. Weiss and G. Gage Kingsbury. Application of Computerized Adaptive Testing to Educational Problems. *Journal of Educational Measurement*, 21(4):361–375, 1984.

I.-Chen Wu, Ti-Rong Wu, An-Jen Liu, Hung Guei, and Tinghan Wei. On Strength Adjustment for MCTS-Based Programs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 1222–1229, July 2019.

Ti-Rong Wu, Hung Guei, Pei-Chiun Peng, Po-Wei Huang, Ting Han Wei, Chung-Chin Shih, and Yun-Jui Tsai. MiniZero: Comparative Analysis of AlphaZero and MuZero on Go, Othello, and Atari Games. *IEEE Transactions on Games*, 17(1):125–137, March 2025.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank - Theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 1192–1199, January 2008.

Yiming Zhang, Athul Paul Jacob, Vivian Lai, Daniel Fried, and Daphne Ippolito. Human-Aligned Chess With a Bit of Search. In *The Thirteenth International Conference on Learning Representations*, October 2024.

A Detailed Training Settings for CASE

CASE consists of a board encoder, a Transformer that maintains the question-response history, and a selector that scores candidate questions. The board encoder is a residual network in the style of AlphaZero, described in Section 4.1. For a question-response pair (q_i, a_i), its value head scores the answered board with the action planes zeroed to give v_i^{before}, and the position the move leads to for v_i^{after}. Because the head reports the value estimate from the perspective of the side to move, and the side to move after the answer is the opponent, we flip v_i^{after} to the player's perspective, so that $v_i^{\text{after}} - v_i^{\text{before}}$ is the value the move gained. The encoder produces a 256-dimensional board embedding, which is projected to the Transformer's model dimension of 384 and appended to the sequence; the two value estimates are projected to the same dimension by a single fully connected layer and added to that token. The Transformer (Vaswani et al., 2017) is a 4-layer causal encoder with 6 attention heads, a feed-forward dimension of 1536 and dropout 0.1.

It uses no positional encoding: the ordering of the history is carried by the causal mask alone, so there is no learned position embedding that would have to extrapolate, and the same network accepts histories longer than those seen in training – each assessment is capped at $T_{\text{train}} = 30$ questions during training, while at test time the network is given up to $T_{\text{eval}} = 200$. Its output is the history embedding E_{H_t}, which a single fully connected layer turns into the rank distribution $P(r_i \mid H_t)$, and which also conditions the selector when it scores each candidate question.

Two implementation details of the selection loop are left out of Section 3.1. First, a question that has already been asked is never asked again: the highest-scoring candidate is checked against the history before it is given to the player. Second, the *unknown* rank the selector uses before the first question is an additional $(n{+}1)$-th row of the rank embedding table rather than one of the n ranks. The distribution the estimator produces from an empty history is an artifact of that emptiness rather than a belief about the player, so converting it into a one-hot would commit every opening question to one arbitrary rank's geometry; with a row of its own the selector learns an opening policy instead, and the first question is chosen by the selector rather than drawn at random.

Training proceeds in three stages, all from the same initialization and all using the same corpus. Stage 1 (30,000 steps) pre-trains the board encoder with the policy and value heads, so that it learns to read a position before it is asked to compare positions. Stage 2 (a further 60,000 steps, to 90,000) pre-trains the estimator to recognize relative strength, by presenting moves played by different ranks on the same board. Stage 3 (10,000 steps, to 100,000) trains the selector jointly with the estimator through simulated adaptive assessments. Those 10,000 steps correspond to 50 rounds of data generation, or $50 \times 2000 = 100{,}000$ assessment trajectories in total. The updates of each round draw from a replay buffer holding the trajectories of the 20 most recent rounds, so at steady state a batch is sampled from $20 \times 2000 = 40{,}000$ assessment trajectories rather than from the round just generated. Appendix D shows that removing Stage 2 leaves the model with no discriminative ability at all.

The selector is supervised by $\mathcal{L}_{\text{pos}}$, the ListMLE loss of equation 6 with $K = 2$, together with the anchor loss $\mathcal{L}_{\text{neg}}$; the ablation in Appendix D examines both choices. The network is optimized with stochastic gradient descent (momentum 0.9, weight decay 10^{-4}) on batches of 64, with gradients clipped to a norm of 5.0. The learning rate starts at 0.02 and is halved every $40{,}000$ steps, so it has fallen to 0.005 by the end of training; the schedule runs on the global step count and therefore spans all three stages. The whole training takes roughly 50 GPU hours on NVIDIA GeForce RTX 3090 cards. Although Stage 3 accounts for only 10,000 of the 100,000 steps, it takes about half of that time, because every step requires running an assessment to generate its own training data, whereas the first two stages read from a fixed corpus. Other hyperparameters are listed in Table 1.

B Baseline Calibration

For CAT_{bin}, we use Stockfish at depth 15 with MultiPV 10 to identify the correct moves for each question, treating a move as correct if its score is within 25 centipawns of the best move. For each rank r, $P(a = \text{correct} \mid q, r)$ is estimated by averaging, across the four calibration players, the total policy probability mass assigned to the correct moves. For CAT_{nom}, each legal move is treated as a distinct response, and $P(a \mid q, r)$ is obtained by averaging the probability assigned to move a by the same four players.

Table 1: Hyperparameters for training CASE.

Parameter	Chess
Estimator	
Number of Blocks	3
Input Channel	16
Hidden Channel	256
Board Embedding Dimension	256
Transformer Dimension	384
Transformer Layers	4
Transformer Heads	6
Transformer Feed-Forward Dimension	1536
Dropout	0.1
Selector	
Number of Blocks	3
Candidates per Step	64
K in $\mathcal{L}_{\text{pos}}$	2
Training	
Optimizer	SGD (momentum 0.9, weight decay 10^{-4})
Learning Rate	0.02, halved every $40{,}000$ steps
Batch Size	64
Training Steps (Stage 1)	30,000
Training Steps (Stage 2)	60,000
Training Steps (Stage 3)	10,000
Assessment Trajectories per Round	2,000
Model Updates per Round	200
Replay Buffer (rounds)	20
Machine	
Main Memory	503GB
Central Processing Unit (CPU)	Intel Xeon Silver 4216 (2.1 GHz)
Graphics Processing Unit (GPU)	NVIDIA GeForce RTX 3090
GPU Hours	$\sim$50

C In-depth Analysis for CASE

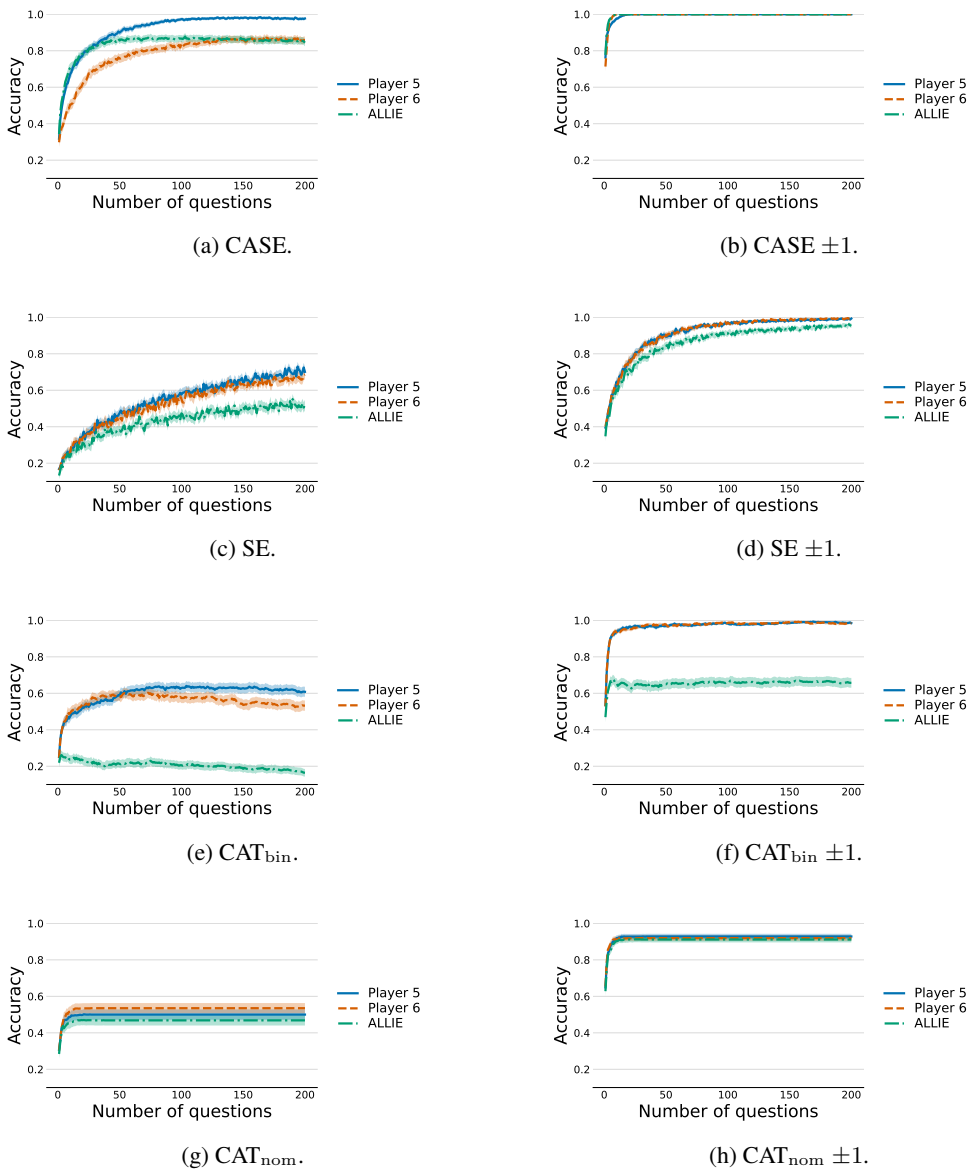
C.1 Breakdown by Evaluation Player

Section 4.1 trains six simulated players per rank, of which the first four supply the responses used to train CASE and to calibrate the baselines. The remaining two are held out from both; we refer to them as players 5 and 6, and together with ALLIE they are the three evaluation players.

Figure 5 reports each method separately on the three evaluation players, rather than averaged over them as in Figure 2. The spread is modest for CASE (97.7%, 85.8% and 84.4% after 200 questions) and for CAT_{nom} (50.0%, 53.5% and 46.8%), but very large for CAT_{bin}, which reaches 60.8% and 53.2% on players 5 and 6 yet only 16.4% on ALLIE, barely above the 12.5% of a uniform guess. The two held-out players are trained in the same way as the four used for calibration whereas ALLIE is not, so the averaged curve understates how far CAT_{bin} falls on a genuinely unfamiliar player. Its curve on ALLIE also declines from the very first question, from 21.9% to 16.4%, so on such a player its questions are not merely uninformative but actively misleading. SE behaves consistently across the three, trailing on ALLIE by about ten points but rising throughout on all of them.

C.2 Ranking-agreement procedure

We refer to one recorded run of the CASE procedure – a single player answering a sequence of adaptively selected questions from an empty history – as a *session*, and the axes of Figure 4a are



(a) CASE. (b) CASE ± 1.

(c) SE. (d) SE ± 1.

(e) CAT_{bin}. (f) CAT_{bin} ± 1.

(g) CAT_{nom}. (h) CAT_{nom} ± 1.

Figure 5: Accuracy of each method broken down by evaluation player, rather than averaged over the three as in Figure 2. Exact accuracy and accuracy within one rank are shown for each pair of methods in turn.

labeled accordingly. Figure 4a is computed from recorded sessions rather than from assessments generated afresh for this analysis. We run the standard CASE procedure on all six simulated players per rank and on ALLIE, collecting 2,560 sessions from each, and record, at the state reached after 50 questions, the selector score that each session assigns to every question in a held-out probe set of 262,144 positions. Including the four training players costs nothing here, because the analysis concerns how the selector re-ranks questions as its own estimate changes rather than how accurate that estimate is. The probe set is byte-identical across all sessions, so any difference between two sessions comes from their histories and not from the questions they happen to see.

Sessions are bucketed by the rank the estimator currently predicts, $\hat{r}_t$, rather than by the player's true rank, because $\hat{r}_t$ is what the selector actually receives. The main text describes the comparison in terms of two players of different strengths; the matrix realizes that comparison over current esti-

mates, which is what its axes report. The eight buckets hold 17,920 sessions in total, the smallest 834.

For each cell of the matrix we perform 200 independent comparisons. In each comparison we draw one session from each of the two buckets, without replacement within a cell, and sample a fresh set of 1,024 probe questions. Of these we keep the 102 (top 10%) with the highest score under an empty history; these 102 questions form the candidate set $\mathcal{Q}$ of the main text, so $m = 102$. The remaining questions sit almost exactly on the anchor, so their relative order is noise at the 10^{-5} scale and including them compresses the whole matrix toward zero. Both sessions rank the same 102 questions by selector score, and we report the Spearman correlation between the two orderings, averaged over the 200 comparisons. The standard error within a cell is 0.008 off-diagonal and 0.001 on the diagonal.

Diagonal cells compare two *different* sessions that share the same current estimate, so they measure how much the ordering moves when only the history changes. They are therefore a noise floor rather than an identity, and any off-diagonal cell should be read against the diagonal cells of its own row and column.

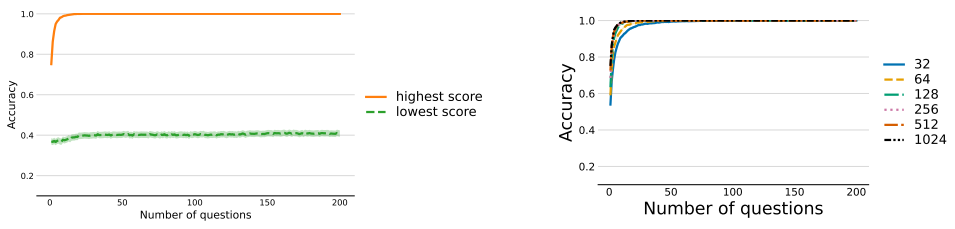
C.3 Question-preference procedure

The question pool is the 1,000 positions with the highest selector score under an empty history. Because that score is taken as the minimum over the eight ranks, it is independent of any particular rank, so the pool is selected for general informativeness rather than for suiting one strength. The pool is chosen without regard to the side to move, and splits into 556 White-to-move and 444 Black-to-move positions; Figures 4b and 4c show the two sides separately, but the nominations below are collected once over the whole pool. The uninformative reference set is drawn separately for each side and subsampled to 500 positions per side.

Nominations are collected as follows. For each of the eight possible values of $\hat{r}_t$ we generate 25,000 selection situations. In each situation we sample 200 of the 1,000 pool questions, condition the selector on a session that has answered 50 questions and is currently estimated at that rank, and record the 10 highest-scoring questions. This gives each question roughly 40,000 opportunities to be nominated. A question is colored by the vote-weighted mean of the ranks that nominated it, and only questions nominated at least 30 times are colored, since fewer votes do not estimate that mean reliably.

The two-dimensional layout is a t-SNE of the 256-dimensional board embeddings produced by the trained board encoder, computed jointly over the pool and the reference set with perplexity 50.

C.4 Accuracy Within One Rank for the Selector Analyses



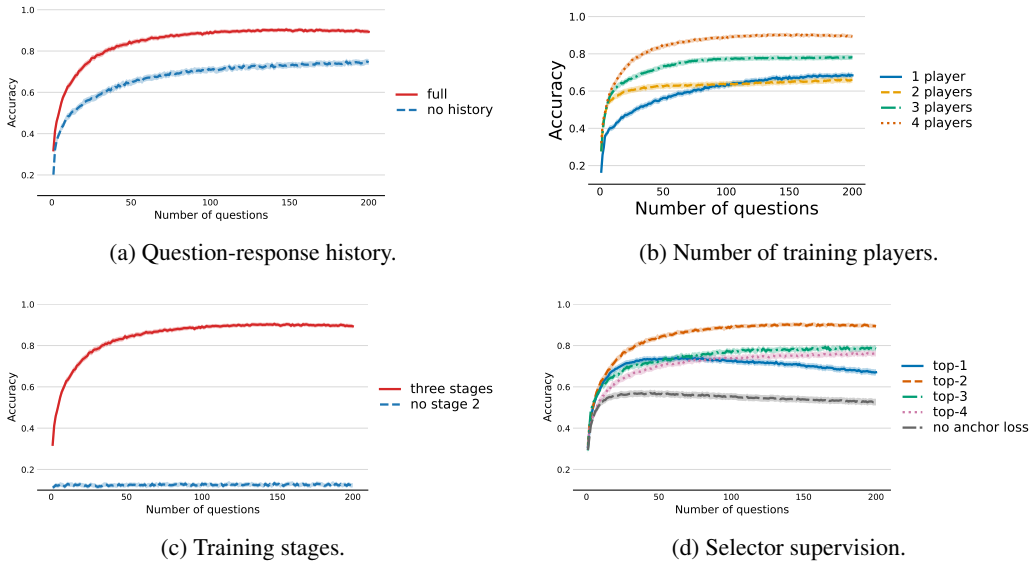
(a) Selection by question score.

(b) Different number of candidate questions.

Figure 6: Accuracy within one rank for the two analyses of Section 4.3.

Reversing the selector (Figure 6a) keeps the estimate near chance ($22/64 \approx 0.344$) throughout, matching the exact-accuracy result. The size of the candidate pool leaves it untouched (Figure 6b): every pool from 32 to 1024 candidates reaches 100.0%, so enlarging the pool buys exact accuracy without changing how far the estimate can stray.

D Further Ablation Results



(a) Question-response history.

(b) Number of training players.

(c) Training stages.

(d) Selector supervision.

Figure 7: Further ablations, reported as exact accuracy against the number of questions asked.

We ablate four design choices, each changing one setting and leaving everything else as described in Section 4.1. The first two concern what the model is given: whether the selector sees the question-response history at all, and how many simulated players per rank supply the responses it is trained on. The third concerns the training schedule, namely whether the estimator pre-training of Stage 2 can be dropped. The fourth concerns the selector objective: how many candidates $\mathcal{L}_{\text{pos}}$ ranks, and whether the anchor loss $\mathcal{L}_{\text{neg}}$ is needed at all. Every variant is evaluated under the protocol of Section 4.1, so each curve can be read directly against the full model's 89.3%.

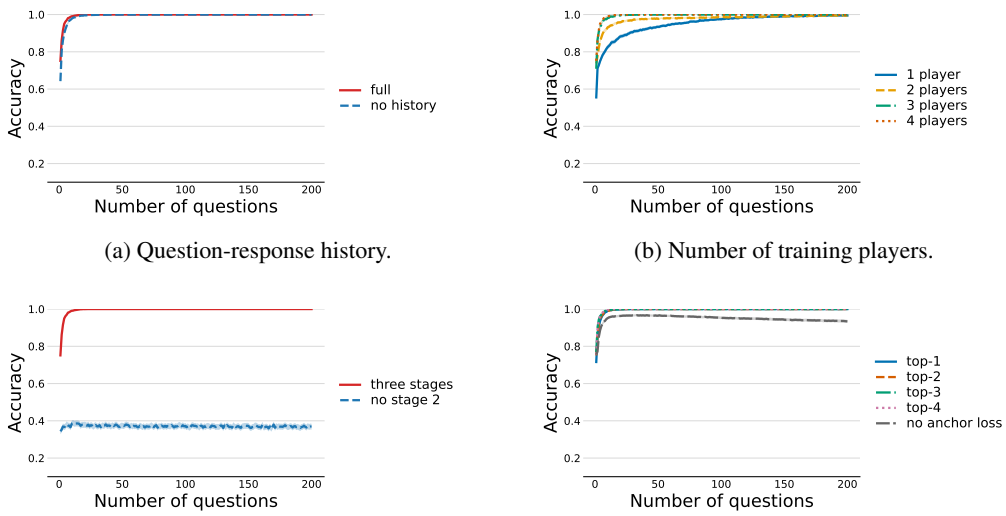
Figure 7a examines the effect of withholding the Transformer's history embedding E_{H_t} from the selector. The difference is substantial: accuracy rises from 74.6% to 89.3%. This also explains why the diagonal of Figure 4a is not 1 – the same estimated rank produces a different ordering of the same candidates once the question-response history differs.

Figure 7b examines how many simulated players per rank supply the responses used in training. Using more players generally generalizes better, rising from 68.4% with one player to 78.0% with three and 89.3% with four, which suggests that the model learns the distribution of playing strength more faithfully. The two-player variant remains the exception at 66.0%, marginally below the single-player one.

Figure 7c examines the effect of removing the second stage (Stage 2: pre-training the estimator). The result is unambiguous: without it the model has no discriminative ability at all, sitting at 12.4% after 200 questions, indistinguishable from a uniform guess among the eight ranks, with a curve that is flat throughout, so that asking two hundred questions is worth no more than asking one. Pre-training the estimator to recognize the relative strength of moves on the same board is therefore indispensable to the training as a whole.

Figure 7d examines K, the number of items ranked by $\mathcal{L}_{\text{pos}}$, and the role of the anchor loss $\mathcal{L}_{\text{neg}}$. Neither more nor fewer ranked items is uniformly better: top-2 is the best at 89.3%, while top-1 falls to 67.1% and top-4 to 76.1%. This suggests that the selector needs only the best few candidates rather than a full ordering, since ranking more items spreads the loss over questions that would never be asked, whereas ranking a single item gives no relative signal at all. Removing the anchor loss is the worst of the five, peaking at 56.6% and falling to 52.6%, so the steady pull away from the uninformative questions remains necessary.

Figure 8 repeats these four ablations as accuracy within one rank. Withholding the question-response history costs nothing on this measure (100.0%), and neither does training on fewer players (99.5%



(a) Question-response history.

(b) Number of training players.

(c) Training stages.

(d) Selector supervision.

Figure 8: The ablations of Figure 7, reported as accuracy within one rank.

with one or two players, 100.0% with three). Removing Stage 2 is again the exception: it ends at 36.9%, so without it the model is not merely imprecise but unable to place the player even to within a neighboring band. The number of ranked items behaves like the first two, with top-1 through top-4 all reaching 100.0% despite differing by 22 percentage points in exact accuracy, but removing the anchor loss ends at 93.4%.