Beyond the Final Board: Turn-by-Turn Skill Assessment in the Game of Go

João Marcos Alves Modesto Ramos[1, 2], **Almiro Furtado de Miranda Neto**[1], **Fabrício Aguiar Silva**[1, 2], **Leandro Soriano Marcolino**[*3]

[1] Federal University of Viçosa (UFV), Florestal Campus, Minas Gerais, Brazil
[2] Manna Team
[3] College of Engineering & Computer Science, VinUniversity, Hanoi, Vietnam
joao.m.ramos@ufv.br, almiro.neto@ufv.br, fabricio.asilva@ufv.br, leandro.sm@vinuni.edu.vn

Abstract

Estimating a user's skill as early as possible is crucial in competitive gaming scenarios, as accurately matching opponents of similar levels prevents frustrating experiences, fosters skill improvement, and reduces player churn. In the game of Go, traditional rating systems evaluate player strength only after a match concludes, which limits the potential for fair matchmaking and real-time difficulty adjustment. Estimating a player's rank dynamically during a game is a significant challenge due to the complex and highly contextual nature of individual moves. To address this, we propose a novel real-time strength estimation approach using a hierarchical meta-model. Our architecture dynamically classifies each move into distinct skill brackets, followed by specialized regressors for exact rank prediction. Our system achieves a promising Mean Absolute Error for the majority of the active player base. This study establishes a foundational framework for turn-by-turn rank estimation, that can be used in adaptive matchmaking mechanisms to enhance player retention and the overall gaming experience. This work was conducted within the scope of the Manna Ecosystem.

Introduction

In recent years, Artificial Intelligence (AI) has advanced significantly, achieving remarkable milestones in domains requiring complex decision-making, such as route planning, medical diagnosis, and strategic games. The board game Go stands out as one of the most complex computational challenges in the world, with an estimated state space of 2.1×10^{170} possibilities. Furthermore, the restrictive capture dynamics cause the board to evolve in non-linear ways, making position evaluation a difficult problem for machines.

Nevertheless, AI-based systems, such as DeepMind's AlphaGo (Silver et al. 2016), have managed to surpass the best human professional players. Since then, the Go community has benefited from the strategies proposed by these systems, leading to the discovery of unprecedented techniques over centuries of human practice (Shin, Kim, and Kim 2021). With the advancement of this research, systems focused on improving user experience have emerged, notably adaptations aimed at emulating real players at varying levels of proficiency (Fujita 2021). However, a complementary challenge lies in accurately inferring the human opponent's level. Correctly identifying this proficiency is fundamental to calibrating an adversarial model to the ideal difficulty, fostering an experience where the player understands the moves and effectively learns from the AI.

Additionally, in the context of online matches, estimating a player strength as early as possible is a practical necessity. When new users join a platform, traditional rating systems require them to complete dozens of matches to stabilize their classification. This calibration period frequently results in unbalanced games, which reduces motivation and increases churn rates (Kang, Suh, and Kim 2024). The disparity between different platforms or local clubs also hinders fair matchmaking in tournaments. Exacerbating this scenario, experienced players may deliberately conceal their true abilities, a practice known as *sandbagging* (Gibson and Sachau 2000) or *smurfing* (Mccauley 2023), aiming to gain unfair advantages against beginners.

In recent literature, works such as Chen, Shih, and Wu (2025) have proposed models capable of estimating a player's rank based on the results of a set of matches, while Kuboki et al. (2025) developed a multi-model approach to refine the accuracy of these estimations. However, a common limitation to these approaches is that inference is performed retrospectively and in an aggregated manner. Besides requiring the completion of the game to calculate the player's level, they rely on deep architectures and evaluations that are computationally expensive.

To bridge this gap and allow an intelligent agent's difficulty to be dynamically adapted during the match, it is important for the predictive system to be computationally lightweight and extremely fast to execute. A turn-by-turn evaluation requires the estimate to be updated at every move in fractions of a second, with low processing cost, ensuring that the game's fluidity is not interrupted. For this purpose, this work proposes a real-time evaluation approach based on a two-layer hierarchical meta-model structured for great efficiency.

The main contribution of this research lies in the novel

[*]Corresponding author. Author also affiliated with Lancaster University.
Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

formulation of the rank inference problem. Although architectures based on Mixture of Experts (MoE) and hierarchical meta-models already exist and are consolidated in the machine learning literature (Bishop and Svensén 2012), even recently resurging with great prominence in the scaling of Large Language Models (LLMs) (Zhang et al. 2025), their adoption for rank prediction modeling remains unexplored. To the best of our knowledge, this is the first work to apply a hierarchical expert architecture specifically for real-time, low computational cost skill prediction. Our contributions are available on GitHub[1].

In this proposal, the first layer of the system consists of an orchestrator model responsible for classifying the move into broad skill bands, routing the inference to the second layer, composed of expert models. Experimental results demonstrate that, although a rigid deterministic routing (*Hard Routing*) already outperforms previous generalist approaches (achieving 23.4% accuracy against the 11.9% *baseline*), the definitive performance leap is achieved by reformulating the problem with a **Probabilistic Strategy** (Bishop and Svensén 2012). By replacing regressors with classifiers in the expert layer and weighting the absolute joint distributions, the architecture actively mitigates cascading error. The resulting system reduces the Mean Absolute Error (MAE) and achieves 57.75% exact accuracy (and over 71% if we consider the margin of error of 1 class), establishing a robust technical foundation for the development of adaptive *matchmaking* mechanisms during gameplay.

The remainder of this paper is organized as follows: the Related Work section reviews fundamental studies in the field. The Methodology section details the proposed architecture, the data processing pipeline, and the probabilistic integration strategies. Next, the configuration of the empirical evaluations is described in the Experiments section, followed by the presentation and discussion of the data in the Results section. Finally, architectural vulnerabilities and concluding remarks are discussed in the Limitations and Conclusions and Future Work sections, respectively.

Related Work

Traditional Rating Systems and AI Agents

Matchmaking systems widely adopted in online gaming platforms, such as Elo (Elo 2008) and Glicko-2 (Glickman 2012), operate exclusively based on the outcome of the match (win, loss, or draw). These algorithms update the player's skill estimate retrospectively, without ever analyzing the board state or the tactical quality of individual moves. On the other hand, with the advent of deep learning, modern Go engines (such as AlphaGo (Silver et al. 2016), Leela Zero (Pascutto 2017), and KataGo (Wu 2020)) have become capable of accurately evaluating the win probability (*win rate*) and the point advantage of any game state. However, the primary goal of these artificial intelligences is to find the optimal move. Although a system like KataGo can identify a suboptimal move made by a human, it was not designed to model the player's profile, as it does not classify whether a specific mistake is characteristic of a beginner (e.g., 30 *kyu*, the traditional amateur scale where higher numbers indicate lower proficiency) or an intermediate amateur player (10 *kyu*). This dichotomy reinforces the need for analytical systems focused purely on modeling human proficiency in real time.

Hierarchical Mixture of Experts (HMoE)

Predictive modeling in high-dimensional environments frequently requires dividing the input space into more tractable subdomains. The Hierarchical Mixture of Experts (HMoE) paradigm has established itself as a robust architectural solution for this purpose. Grounded in probabilistic approaches, such as the Bayesian model proposed by (Bishop and Svensén 2012), the HMoE architecture allows the problem's complexity to be modularly partitioned through nested layers of routing functions and expert subnetworks.

Historically, this architecture has been adapted under various topologies to mitigate interference between different data patterns. (Irsoy and Alpaydin 2018) explored the use of regularization in tree structures, where the internal nodes act as restricted routing functions (*gating functions*) and the leaves as the definitive expert models, a topology aligned with the orchestration screening proposed in this work. More recently, the HMoE paradigm has evolved beyond classical decision trees, being actively integrated into deep neural architectures to promote generalization and efficiency in learning highly complex tasks (Li et al. 2025).

Currently, the core concept of conditional partitioning in Mixture of Experts (MoE) is experiencing a highly relevant resurgence at the forefront of Artificial Intelligence. The technique has become the backbone for the efficient scaling of Large Language Models (LLMs), enabling a massive increase in parameter capacity without a proportional increase in the computational cost of inference (Zhang et al. 2025).

However, despite the empirical and theoretical success of these architectures in consolidated domains such as Natural Language Processing and hardware synthesis, the literature lacks their transposition to dynamic skill estimation modeling. Although the HMoE architecture is optimized to capture nuances in data subpopulations, its direct application to turn-by-turn skill prediction remains unexplored, positioning this work as an innovation at the intersection of hierarchical ensemble architectures and skill estimation analysis.

Strength Estimation

The work of Chen, Shih, and Wu (2025) introduces a novel strength evaluation system for Go, comprising a Strength Estimator (SE) and an SE-based Monte Carlo Tree Search (SE-MCTS). The estimator calculates strength scores and predicts player ranks from game records without direct human interaction. Subsequently, the SE-MCTS utilizes these scores to dynamically adjust the artificial agent's playing strength and style. Through their experiments, the authors achieved over 80% accuracy in predicting ranks by observing a history of only 15 full games, a significant improvement over previous methods that required up to 100 games to reach 49% accuracy.

[1]https://github.com/raitocan/Turn-by-Turn-Skill-Assessment-in-the-Game-of-Go

Building upon this logic, (Kuboki et al. 2025) focused on the observation that human behavioral tendencies vary according to their proficiency. To improve estimation accuracy, the authors combined traditional strength scores with policies derived from neural networks, which were trained on match data from players across different skill levels. Their proposed method achieved an 80% accuracy in strength estimation when observing 10 matches, increasing to 92% when analyzing a 20-game history.

However, a fundamental limitation of these state-of-the-art approaches is their reliance on retrospective, aggregated data. In both studies, strength prediction is performed only after multiple matches have concluded, utilizing the entirety of the data collected throughout the games. Instead of relying on post-game analysis, our work addresses this gap by focusing on dynamic, real-time rank estimation, evaluating the player's skill turn-by-turn while the match is still unfolding.

Methodology

Let X be the multidimensional feature space and Y be the target variable space, corresponding to the player's ordinal rank. The objective of the system is to approximate a global predictive function $F : X \to Y$.

For each meta-class $m_i \in M$, there is a subset of labels $L_i \subset Y$ representing the rank levels exclusively contained within that category. These subsets are mutually exclusive and cover the entire classification spectrum, such that:

$$\bigcup_{i=1}^{K} L_i = Y \quad \text{and} \quad L_i \cap L_j = \emptyset, \forall i \neq j$$

Due to the high behavioral variance in the game of Go, mapping X directly to Y with a single model introduces considerable statistical noise. To mitigate this complexity, the target domain is partitioned. A set of meta-classes $M = \{m_1, m_2, \ldots, m_K\}$ is defined, where K represents the total number of macroscopic skill categories.

The *Hierarchical Mixture of Experts* (HMoE) architecture adopted in this work consists of two independent functional components: an Orchestrator model and a set of Experts. The Orchestrator acts as a routing function $G : X \to M$, mapping an input state $x \in X$ to the estimated meta-class $\hat{m}_i$. Concurrently, a set of experts $E = \{E_1, E_2, \ldots, E_K\}$ is defined, in which each expert $E_i : X \to L_i$ is trained and optimized exclusively on the training instances whose true label y belongs to the subdomain L_i.

In the inference phase, the final architecture prediction $F(x)$ for a new state x is determined by the integration method between the orchestrator model output $G(x)$ and the set of experts E. The exact formulation of $F(x)$ depends fundamentally on how the uncertainty and probabilities of the initial classification are handled by the system. In this work, instead of assuming a single decision pathway, three distinct ways of performing this routing were explored:

1. Hard Routing: The architecture deterministically selects the path of highest probability, ignoring residual uncertainty. The index of the winning meta-class is given by $\hat{\imath} = \arg\max_{i \in \{1,\ldots,K\}} w_i(x)$, where w is the probability of the meta-class i, and the final prediction is the isolated output of that expert:

$$F_{hard}(x) = E_{\hat{\imath}}(x)$$

2. Soft Routing: Under the classic Mixture of Experts paradigm, the final prediction is a linear combination of all regressors. The output of each expert is weighted by the probability of the respective meta-class:

$$F_{soft}(x) = \sum_{i=1}^{K} w_i(x) E_i(x)$$

3. Probabilistic Method: In this approach, the regressor expert models E_i are replaced by classifier experts C_i. Each classifier C_i outputs a conditional probability distribution over the exact rank labels belonging to its subdomain, denoted by $P(y \mid x, m_i)$.

The foundation for the global probability of the model is based on the product rule of probability combined with the hierarchical structure of the classes. If we define event A as the prediction of the specific rank (y) and event B as the prediction of its corresponding meta-class ($m_{k(y)}$), the joint probability of them occurring simultaneously is given by:

$$P(A \cap B) = P(A \mid B) \cdot P(B)$$

Since a specific rank y belongs mutually exclusively to a single meta-class $m_{k(y)}$, the occurrence of rank y necessarily implies that it is contained within its meta-class (i.e., event A is a subset of B). As a logical consequence, the probability of event A alone is equal to the joint probability of A and B:

$$P(A) = P(A \cap B)$$

Translating this logic into the context of the model, we have that the absolute probability of a given rank is equivalent to the joint probability of the rank and its meta-rank:

$$P(\text{rank}) = P(\text{rank} \cap \text{meta-rank})$$

$$P(\text{rank}) = P(\text{rank} \mid \text{meta-rank}) \cdot P(\text{meta-rank})$$

In this formulation, the prior probability of the meta-rank $P(\text{meta-rank})$ corresponds to the confidence weight $w_{k(y)}(x)$ output by the global model, and the conditional probability $P(\text{rank} \mid \text{meta-rank})$ corresponds to the output of the expert model $P(y \mid x, m_{k(y)})$.

Thus, applying the Law of Total Probability under this local conditional independence, the final prediction selects the rank class with the highest absolute global probability:

$$F_{prob}(x) = \arg\max_{y \in Y} \left[w_{k(y)}(x) \cdot P(y \mid x, m_{k(y)}) \right]$$

To illustrate this concept practically: suppose the Orchestrator assigns a 60% probability that a given move belongs to the Single-digit kyu (SDK) meta-class. Within that subdomain, the SDK Expert classifier estimates a 30% probability that the specific rank is 5k. The absolute joint probability for the 5k label evaluates to 18% (0.60×0.30). The system computes this joint probability for all possible ranks across

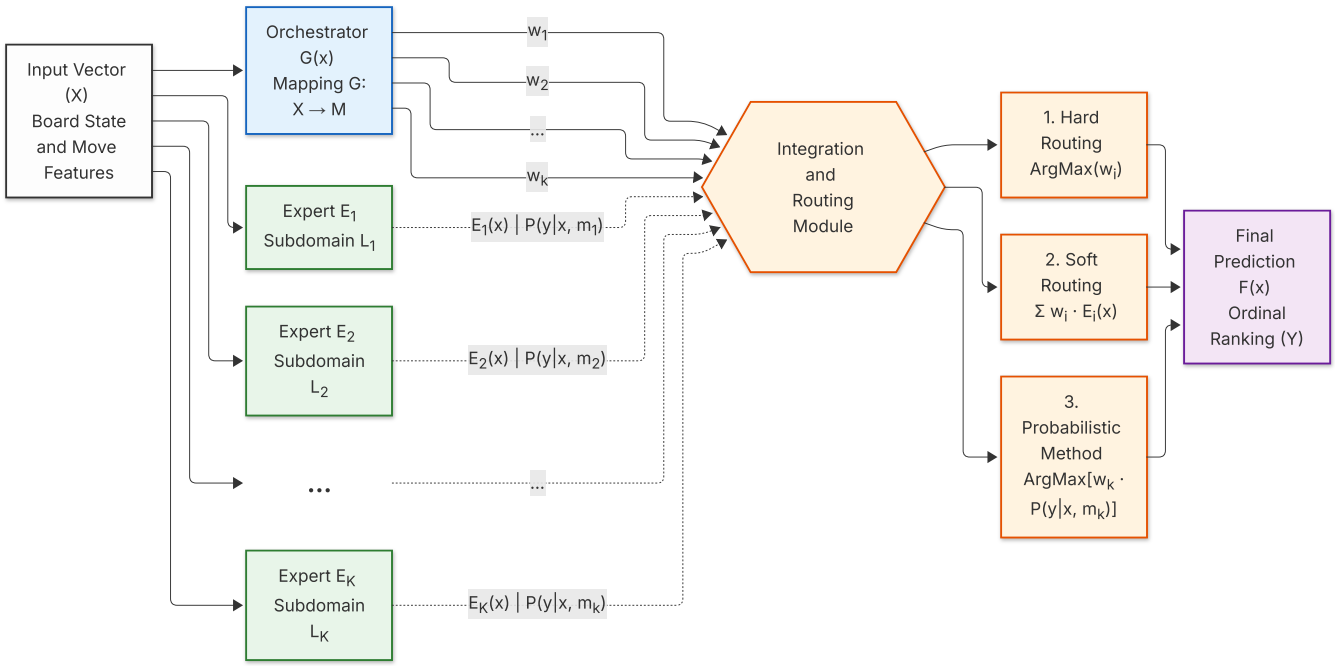


Figure 1: Architecture of the Hierarchical Mix of Experts

all experts, ultimately outputting the specific rank that maximizes this global confidence.

A summary of this architecture can be seen in Figure 1. To evaluate the feasibility and effectiveness of the proposed architecture, the development pipeline was structured from data collection and formatting to the design of the meta-model. Two distinct datasets were utilized: a public repository (provided by Kuboki et al. (2025)) and a primary dataset extracted directly from an online gaming platform.

Data Collection

The first dataset comes from the public repository of the Fox[2] server, used analogously to the studies by Chen, Shih, and Wu (2025) and Kuboki et al. (2025). From this repository, a sample of matches of players ranked between 5 Kyu and 9 Dan was extracted. 5,000 matches were selected for each of the 11 *ranking* subgroups, totaling 55,000 unique matches which, when analyzed from the individual perspective of each player, result in 110,000 game instances.

The second data collection front was conducted on the Online-Go (OGS) platform[3], a global environment where users play real-time Go matches. The platform provides records in the standard text tree format known as *Smart Game Format* (SGF). The data *scraping* process began with the top-ranked player in the official 19x19 category. From this initial node, the algorithm collected the player's match history and recursively added their opponents to the collection queue. This recursive collection was conducted in strict compliance with the OGS Terms of Service, carefully adhering to its API guidelines and rate limits. Respecting the platform's privacy guidelines, games configured as private were summarily ignored. This process culminated in a dataset of 157,856 matches, comprising interactions from approximately 33,000 players of all levels.

Preprocessing

The match history describes the complete journey of the player *ranking* evolution. To ensure data quality, rigorous filtering was applied: matches with fewer than 10 moves were removed (avoiding premature victories due to abandonment), along with games containing corrupted SGF records and instances where a player's *ranking* was not explicitly classified.

Once cleaned, the SGF data were divided turn by turn, inverting the perspective so that each move exclusively contained the attributes and board view of the current player. For feature extraction, an approach inspired by the vector representation of KataGo (Wu 2020) was adopted. The board state was encoded into four essential binary matrices: the positions of white stones, black stones, all occupied intersections, and empty intersections. In parallel, a numerical vector representing tactical and contextual properties was extracted to provide a lightweight representation of the board state. Specifically, the numerical features comprise: the (X, Y) board coordinates of the current move, the current turn number, the number of captured stones by both black and white players, the estimated instantaneous score difference, and the handicap level (compensation points, Komi).

Finally, the target variable (the ranking in *string* format) underwent ordinal encoding, being mapped to a continuous

[2]https://www.foxwq.com/
[3]https://online-go.com/

integer scale where higher values represent greater levels of mastery (e.g., 25k was mapped to 0, while 3 Dan was mapped to 28).

One of the main problems in board game modeling is data leakage. If a single player turns were distributed randomly, the model might memorize that user style rather than generalizing the patterns of their ranking. To mitigate this risk, the splitting of the primary dataset (80% training, 20% testing) was conducted strictly at the unique player identifier (*Player ID*) level. Thus, it was ensured that the final evaluation was performed solely on individuals whose moves had never been exposed to the model during training.

To structure the prediction, the traditional Go *ranking* system was segmented into four broad meta-classes: Beginners (*Beginner* - B, up to 25k), *Double-digit kyu* (DDK, 24k to 10k), *Single-digit kyu* (SDK, 9k to 1k), and *Dan* (D, encompassing all professional and amateur mastery levels). The volumetric distribution of the segmented instances is outlined in Table 1.

Name of the Meta-Class	Quantity
D	102,580
SK	256,113
DDK	462,014
B	142,812
Total	**963,519**

Table 1: Frequency Distribution by Meta-Class

Model Architecture

There are two main approaches to developing this architecture: through a tree structure, where internal nodes act as routing functions and leaves are the expert models (Irsoy and Alpaydin 2018); or in deep neural architectures, involving nested blocks across different network layers (Li et al. 2025). The architecture proposed in this work consists of a hierarchical meta-model designed to mitigate the difficulty of predicting a wide range of continuous rankings. This architecture operates in two distinct phases:

1. **Meta-Classifier (Orchestrator):** The input move is initially processed to route the instance to the correct meta-class. To mitigate severe class imbalance, class weights were applied during training.
2. **Expert Models:** Four distinct models based on the *XGBoost Regressor* algorithm were trained, one for each meta-class. Once the Orchestrator defines the move's meta-class, it is routed to the corresponding expert, according to the chosen strategy, who then makes the final prediction of the numerical ranking.

Since the objective is to predict the player's ranking, the initial idea would be to use accuracy as a comparison metric; however, considering the high number of existing classes, it is valuable to understand the model's behavior and account for predictions that fall close to the true value, penalizing them less severely. With this in mind, the chosen loss function was the Mean Absolute Error (MAE) for the regression models, and Accuracy for the Meta-classifier. For the comparative evaluation, *Top-k Accuracy* was also used, which computes the number of times the correct label is among the k labels with the highest predicted probability.

As illustrated in Figure 1, the input is first submitted to an Orchestrator Meta-Classifier. To select the model, a test was conducted with a sample of the database, which can be seen in Table 2. Based on these results, the *XGBoost Classifier* (Chen and Guestrin 2016) with class balancing weights was chosen, which estimates the probabilistic skill band for that turn. Depending on the adopted routing strategy, the instance is forwarded to the subordinate expert models (*XGBoost Regressors*), trained exclusively to predict the exact *ranking* within their respective *Kyu* or *Dan* subdomain.

Metric	GradientBoosting	XGBoost	LightGBM
Accuracy	28.30%	37.15%	27.98%
Top-3 Acc	66.31%	73.20%	68.02%
MAE	1.282	1.074	1.229

Table 2: Model Performance Results

To ensure reproducibility and a fair comparison among the evaluated algorithms, hyperparameters were carefully calibrated and standardized during the training phase, as we can see Table 3. Specifically, for the winning configuration, the expert layer utilizes XGBoost Classifiers to output the conditional probability distributions. The number of sequential trees (`n_estimators` or `max_iter`) was fixed at 300 across all baseline models, orchestrators, and these classifier experts to establish a common training horizon. Structural parameters, such as maximum tree depth and maximum leaf nodes, were constrained to prevent overfitting while maintaining sufficient expressive capacity to map the high dimensionality of the board states. Furthermore, hardware acceleration was extensively employed to meet the low-latency requirements of the study: multi-core CPU utilization (`n_jobs=-1`) was enabled across compatible models, and GPU processing (`device="cuda"`) coupled with histogram-based tree methods (`tree_method="hist"`) was utilized to optimize the training time of both the XGBoost regressors and classifiers.

Model	Hyperparameters
XGBoost Classifier	`n_estimators`: 300, `max_depth`: 4, `n_jobs`: -1
XGBoost Regressor	`n_estimators`: 300, `max_depth`: 4, `tree_method`: 'hist', `device`: 'cuda', `n_jobs`: -1

Table 3: Hyperparameters for Evaluated Models

Experiments

The first experiment established a direct comparison with the *baseline* from Chen, Shih, and Wu (2025), utilizing the public repository of Fox matches [4]. In this dataset, the sample made available by the work of Kuboki et al. (2025) was

[4]https://github.com/featurecat/go-dataset

collected, and the performance of both models was evaluated on the test data.

Given that the work by Chen, Shih, and Wu (2025) predominantly classifies high-level players, the comparison dataset was divided according to the available rankings. Had the previous division been applied, only two classes would have been covered: Dan and Single-Digit Kyu. Therefore, a new division focusing on high-level play was established, splitting the rankings into 3 classes:

- Kyu - Corresponds to all values up to 1 Dan.
- Low Dan - Corresponds to values between 1 and 4 Dan.
- High Dan - Corresponds to values greater than 4 Dan.

The second experiment consisted of a comprehensive ablation study designed to evaluate both feature dependencies and the most effective routing strategy for the HMoE architecture. To understand the model's reliance on different data types, the input space was evaluated under constrained scenarios, isolating spatial board geometry (*Only Board*) and contextual metadata (*No Board*). Concurrently, the performance of a baseline Singular model was compared against three hierarchical routing methods: *Hard Routing*, *Soft Routing*, and a *Probabilistic* strategy. The evaluation across all configurations was based on Mean Absolute Error (MAE), exact Accuracy (Top-1), and Accuracy with a one-level tolerance margin (Top-2).

Results

To evaluate the effectiveness of the proposed architecture, the results are structured around two complementary analyses. The first consists of a detailed internal analysis through an ablation study, dedicated to investigating the impact of different feature subsets and routing strategies on the hierarchical meta-model. This step justifies the architectural decisions adopted to handle high-dimensionality scenarios and mitigates cascading errors. Following this internal validation, the second analysis focuses on external validation, establishing a direct comparison with state-of-the-art approaches for classifying players in high-level matches to demonstrate the system's competitive advantage in end-to-end rank estimation.

Ablation Study

This experiment consisted of an ablation study to define the best routing strategy for the HMoE architecture. Performance was evaluated based on the Mean Absolute Error (MAE), Accuracy (Top-1), and Accuracy with a one-level tolerance margin (Top-2). Scaled tests with up to 2 million instances consolidated the superiority of the probabilistic strategy, detailed in Table 4.

The comparative analysis reveals fundamental differences regarding both feature dependency and routing methods. Using the singular model yields an MAE of 2.152; however, its low accuracy (13.06%) highlights the limitation of non-hierarchical architectures in mapping the high dimensionality of Go to exact predictions.

The feature ablation scenarios further highlight how the model interprets the game. When spatial features are completely removed (*No Board*), performance falls slightly across all metrics. This proves that the tactical geometry of the board is the most importat factor of skill prediction. Conversely, relying exclusively on the board (*Only Board*) yields a slightly degraded performance compared to the full-feature baseline (*Singular Model*). This dynamic indicates that while the raw board state contains the vital features required by the models, contextual metadata is needed to properly frame the match and correctly route the initial prediction.

Strategy	**MAE**	**Accuracy (%)**	**Top-2 Acc (%)**
Only Board	2.220	12.45	37.25
Only Metadata	2.541	10.68	31.45
Singular model	2.152	13.06	38.76
Hard Routing	2.172	18.78	53.36
Soft Routing	2.147	13.35	38.91
Probabilistic	2.030	40.60	57.08

Table 4: Comparison of Routing Strategies in the Ablation Study in 2 milion instances

Upon integrating the hierarchical meta-model with the complete feature set, a divergence is observed between continuous error optimization and exact precision. The *Soft Routing* strategy acts as a regularization mechanism: by diluting the prediction through the weighted average of all experts, it manages to slightly suppress the error distance (MAE 2.147) but fails to provide significant accuracy gains (13.35%). In contrast, *Hard Routing* exhibits the opposite behavior. By deterministically delegating the prediction to a single expert, the architecture slightly sacrifices the MAE (which rises to 2.172) due to the inevitable cascading error effect—when the orchestrator fails the class screening, the error of the expert outside its domain is severe. However, this sacrifice pays off by isolating the specific heuristics of the correct class, drastically raising the Accuracy (18.78%) and more considerably the accuracy when considering a 1-class error margin (53.36%).

Despite these individual trade-offs, the most disruptive breakthrough lies in the Probabilistic strategy, which actively mitigates the flaws of rigid routing and consolidates itself as the superior architecture. By weighting the absolute joint probabilities, this approach bypasses the cascading error trap. As a result, it not only minimizes the global prediction deviation (obtaining the best MAE of the suite, at 2.030) but also achieves a monumental leap to 40.60% in Accuracy and 57.08% in Top-2. Thus, it is concluded that joint probabilistic analysis is the most effective mechanism when dealing with massive datasets, justifying the adoption of the probabilistic approach as the definitive routing strategy of the proposed model.

To evaluate the practical viability of the proposed architecture for real-time applications, we conducted a latency analysis of the inference pipeline. Table 5 details the inference time in milliseconds (ms) per move. While the baseline Singular Model requires approximately 260 ms to evaluate a move, the hierarchical approach introduces an expected structural overhead. Consequently, the end-to-end latency

for the Probabilistic strategy totals 1340 ms per move. Despite this increase, a response time of approximately 1.3 seconds remains highly efficient and perfectly suitable for turn-by-turn dynamic adjustment without disrupting the fluidity of the game.

Strategy	Latency (s)
Singular Model (Baseline)	0.26
Hard Routing	1.49
Soft Routing	1.60
Probabilistic	1.34

Table 5: Inference Latency per Move by Routing Strategy

Baseline Comparison

It is important to emphasize the fundamental difference in the evaluation scope between the models. While the baseline methodologies (Chen, Shih, and Wu 2025; Kuboki et al. 2025) compute accuracy by retrospectively analyzing the aggregated data of complete matches or game histories, our HMoE approach infers the rank exclusively from the static context of individual, isolated turns. Achieving competitive accuracy under this strict turn-by-turn constraint highlights the efficiency of the hierarchical strategy, as it operates with only a fraction of the temporal information available to the baselines.The comparative performance results of the architecture against the state-of-the-art are presented in Table 6.

	Top-K Accuracy	
Method	**K = 1**	**K = 3**
Chen - SE	11.9	29.6
Chen - SL	12.5	34.1
Singular model	5.03	12.95
Soft Routing	15.2	43.1
Hard Routing	23.4	61.4
Probabilistic	60.3	70.2

Table 6: Top-K Accuracy Results

The results demonstrate the compelling effectiveness of the proposed architecture and reveal the challenging nature of turn-by-turn prediction. Initially, it is observed that training a solution using a Singular model achieves only 5.03% in Accuracy, a performance inferior to the baseline (11.9% and 12.5%), which utilizes the aggregated history of complete matches. This decline is not unexpected: attempting to map the vastness of board states directly to a specific *ranking*, evaluating only the static features of a single move, and optimizing a global loss function generates an extreme level of statistical noise. The singular model fails when trying to simultaneously reconcile the error patterns of beginner players with those of professional players within a single representation space.

It is precisely at this point of failure that the hierarchical architecture proves its value. By segmenting the problem and routing the prediction to expert models via *Hard Routing*, the accuracy jumps to 23.4%. This result not only circumvents the collapse of the singular model but also practically doubles the baseline performance. Additionally, *Hard Routing* outperforms *Soft Routing* (15.2%), indicating that, in the absence of the match's historical context, deterministically delegating the inference to a highly calibrated expert in its tactical subdomain is a much more robust strategy than diluting the prediction through probabilistic averages.

However, the most significant breakthrough of the study is revealed by the **Probabilistic** strategy, which achieves 60.3% in the Accuracy metric and 70.2% in Top-3. This gain is explained by the direct mitigation of the main vulnerability of meta-models: cascading error. Instead of forcing rigid routing through a single expert, which penalizes the system if the orchestrator fails the initial screening, this strategy evaluates the absolute joint probability. By weighting the internal distributions of each classifier expert by the global confidence of the orchestrator, the model is able to identify the most probable ranking. This result demonstrates that joint probabilistic analysis is the key to enabling accurate skill evaluation in real time and with low computational cost.

To validate the hypothesis that the hierarchical architecture enables the learning of specific heuristics for each skill band, we proceeded to extract the Feature Importance from each trained model. The Relative Gain metric was used, which quantifies the average improvement in the accuracy of decision tree splits provided by each attribute. To allow for a fair comparison, the gain values were normalized (from 0 to 1) relative to the highest-impact variable of each model. Table 7 presents the most relevant variables extracted from the system.

Model	Feature	Norm.
Orchestrator	Handicap	1.000
Orchestrator	Pos (15, 15)	0.103
Orchestrator	Pos (3, 15)	0.097
Orchestrator	Pos (15, 3)	0.089
Orchestrator	Pos (3, 3)	0.071
Exp. Kyu	Handicap	1.000
Exp. Kyu	Pos (6, 9)	0.273
Exp. Kyu	Pos (6, 11)	0.251
Exp. Kyu	Pos (7, 10)	0.247
Exp. Kyu	Pos (6, 10)	0.247
Exp. Low Dan	Komi	1.000
Exp. Low Dan	Pos (7, 4)	0.826
Exp. Low Dan	Pos (11, 0)	0.801
Exp. Low Dan	Pos (9, 4)	0.801
Exp. Low Dan	Pos (12, 11)	0.800
Exp. High Dan	Komi	1.000
Exp. High Dan	Pos (3, 15)	0.796
Exp. High Dan	Pos (15, 3)	0.681
Exp. High Dan	Pos (3, 3)	0.676
Exp. High Dan	Pos (15, 16)	0.664

Table 7: Top 5 Highest Impact Features per Model (Normalized Gain)

As detailed in Table 7, there is a clear divergence in the distribution of feature importance, reflecting the inherent

rules of skill progression in Go. The Orchestrator model utilizes the *Handicap* variable as the feature of highest importance, and assigns residual weights (less than 0.11) to the board positioning variables. The most notable aspect, however, is the geometric identity of these spatial positions: the coordinates *Pos (15, 15)*, *Pos (3, 15)*, *Pos (15, 3)*, and *Pos (3, 3)* correspond exactly to the four star points of the board's corners (*Hoshi*). Given that high *handicap* values, placed on these exact canonical points, are exclusive indicators of matches involving amateur players, this behavior is perfectly consistent with the dynamics of the game. Following this logic, the expert focused on beginners (*Exp. Kyu*) maintains the *Handicap* as its main variable to differentiate the various learning levels.

On the other hand, when observing the high-proficiency expert regressors (*Exp. Low Dan* and *Exp. High Dan*), a drastic paradigm shift occurs. The *Handicap* completely disappears from the top variables—since *Dan*-level matches are predominantly played on equal terms—being replaced by *Komi* (compensation points) as the dominant structural variable. Additionally, the relative weight of spatial positioning variables (such as *Pos (7, 4)*, *Pos (11, 0)*, *Pos (3, 15)*, etc.) rises in these elite models, approaching the maximum value (with normalized gains between 0.66 and 0.82). In the *Exp. High Dan* model, the continuing relevance of the board's corners, now accompanied by critical adjacent positions (such as *Pos (15, 16)*), does not stem from artificial advantages, but reflects the extreme importance of studying opening theory (*Joseki*). This internal behavioral variation proves that the segmentation of the input space avoids excessive generalization, forcing each expert model to adapt exclusively to the patterns and heuristics of its respective class.

Generalization to Another Dataset

For this dataset, a player natural progression was used as a reference to define subdivisions based on the learning stages of a Go player. A beginner, as they advance, becomes an amateur at the Double-digit kyu (DDK) level, then progresses to a casual player at the Single-digit kyu (SDK) level, and ultimately achieves a Dan (D) level.

For this experiment, a player with a rank up to 25k was classified as a beginner, a player between 24k and 10k as a Double-digit kyu (DDK), a player from 9k to 1k as a Single-digit kyu (SDK), and all Dan-level players were grouped into a single class.

Table 8 presents a performance comparison of the experts. The superiority of the classification approach over regression can be observed across all evaluated subdomains. Analyzing the scope of the meta-classes, variations in the predictions are noted according to the player skill level, which can be explained by the imbalance in the dataset.

End-to-End Rank Estimation

Table 9 presents the evaluation of the Mean Absolute Error (MAE) by meta-class, comparing the performance of a direct predictor (*Singular model*) against different hierarchical *ensemble* strategies (*Hard Routing*, *Soft Routing*, and *Probabilistic*). The MAE represents the average distance, in ranking levels, between the system's final prediction and the player's actual ranking.

Expert	Type	MAE	Accuracy	Top-2 Acc
Exp. B	Regressor	0.68	0.66	0.86
Exp. B	Classifier	0.02	0.99	0.99
Exp. DDK	Regressor	1.20	0.33	0.71
Exp. DDK	Classifier	0.14	0.97	0.98
Exp. SK	Regressor	10.93	0.02	0.06
Exp. SK	Classifier	7.26	0.62	0.76
Exp. D	Regressor	9.65	0.03	0.08
Exp. D	Classifier	4.60	0.82	0.91

Table 8: Individual Performance of Each Expert

Meta-Class	Singular model	Hard Routing	Soft Routing	Probabilistic
B	10.7028	15.8495	9.6626	7.3158
DDK	26.4213	34.7500	26.4608	13.4019
SK	13.1935	10.9735	13.0449	8.4552
D	13.1510	14.8324	12.8259	7.7527

Table 9: Summary of Mean Absolute Error (MAE) by Meta-Class according to Strategy

The results clearly demonstrate the cascading error effect inherent to rigid hierarchical architectures. Observing the *Hard Routing* strategy, the *Single-digit kyu* (SK) class achieves its best relative performance, obtaining an MAE of 10.97. However, for the other skill bands, especially the *Double-digit kyu* (DDK) class, the MAE of 34.75 shows a considerably worse performance than using a single model.

This elevated error in *Hard Routing* is predominantly due to failures in the first routing layer: when the orchestrator incorrectly classifies a DDK or Beginner move, the instance is forwarded to an expert model that was never exposed to those rankings. Consequently, the expert model is forced to predict a rank within its own restricted training window, instantly generating a deviation of dozens of levels. In contrast, the *Probabilistic* strategy proves to be an effective solution for this architectural vulnerability. By weighting global uncertainties and merging the distributions, it mitigates cascading errors and achieves the best performance across all classes, reducing the DDK MAE to 13.40 and reaching 7.31 and 7.75 (for B and D, respectively).

Strategy	MAE	Accuracy	Top-2 Acc (± 1)
Singular model	14.2	0.014	0.045
Hard Routing	14.1	0.025	0.061
Soft Routing	14.0	0.015	0.046
Probabilistic	8.7	0.578	0.711

Table 10: Performance Comparison of Routing Strategies

Table 10 consolidates the overall evaluation of the system, contrasting the performance of different routing approaches

in the hierarchical ensemble against a single predictive model. The analysis of these general metrics reaffirms the impact of choosing the inference aggregation mechanism.

The Singular model, *Hard Routing*, and *Soft Routing* approaches present statistically similar and markedly unsatisfactory performances. The Mean Absolute Error (MAE) stagnates around a distance of 14 levels, and the exact accuracy is practically null (below 3%). This scenario proves that the simple deterministic partitioning of the problem (*Hard Routing*) or the weighted average of the regressions (*Soft Routing*) fail to mitigate the uncertainty inherent in evaluating a single, isolated turn.

In contrast, the *Probabilistic* strategy promotes a drastic qualitative leap in the performance. By treating predictions as conditional probability distributions and appropriately unifying them, the approach reduces the global MAE to 8.65. The impact on precision is also considerable: the exact prediction accuracy improves from 1.4% in the base model to 57.75%, and considering a margin of error of ± 1 level, it reaches a robust 71.06%.

These results show that the effectiveness of the architecture in the domain of skill prediction does not lie solely in the partitioning and specialization of the leaf models, but fundamentally in the probabilistic method's ability to weight the meta-classes and cross-reference this information with the joint distribution of the final classifiers.

Limitations

The fundamental objective of this work is the development of a lightweight predictive model capable of estimating the player ranking at each turn with fast execution and low computational cost. This architectural constraint guided methodological choices and defines the main limitations of the study.

Because it focuses on instantaneous inference, the model relies on a single isolated turn. In the game of Go, the quality of a move may depend on a long-term strategic plan; therefore, the absence of historical or sequential context makes it difficult for tree-based algorithms to capture the true tactical intention of the move. An opening, for example, can be positionally identical regardless of whether it is executed by a beginner or a professional player.

It is also this efficiency requirement that justifies the selection of the baseline. The comparative validation was restricted to the work of Chen, Shih, and Wu (2025), whose complexity suits the scope of fast inference. In contrast, although the methodology of Kuboki et al. (2025) presents superior results in strength estimation, its architecture relies on *prior* scores extracted from imitation models and on the calculation of advantage loss derived from deep Artificial Intelligence engines, such as KataGo (Wu 2020) and LeelaZero (Pascutto 2017). This dependency results in a increase of processing cost, making it turn-by-turn execution unfeasible in a scenario that demands fast responses.

From a structural standpoint, the adoption of the hierarchical meta-model introduces a vulnerability to cascading errors. If the orchestrator fails in the preliminary identification of the skill range, the instance is routed to an expert model trained on a subset of data that is completely incompatible with the player's tactical reality, ensuring a substantial deviation in the final prediction.

Finally, this architectural susceptibility is aggravated by the natural imbalance of player demographics in the data collection. The dense concentration of instances at the intermediate amateur level biases the training, severely limiting the model ability to generalize patterns to the extremities of the learning curve.

Conclusions and Future Work

This work proposed an innovative approach for the dynamic estimation of rankings in the game of Go, in contrast to traditional retrospective methodologies that rely on processing complete games after they end. Through the implementation of a two-layer hierarchical meta-model, the computational feasibility of performing accurate turn-by-turn inferences was demonstrated. Fundamentally, the ablation study revealed that reformulating the problem from regression to classification, combined with the adoption of a probabilistic routing strategy, vastly outperforms both rigid partitioning approaches (*Hard* and *Soft Routing*) and the baseline for comparison (Chen, Shih, and Wu 2025).

By weighting absolute joint distributions, the architecture actively mitigated the main vulnerability of these hierarchical systems, which consists of cascading error. The probabilistic method achieved the lowest Mean Absolute Error (MAE) of the conducted tests (reduced to 8.65) and the highest accuracy, reaching 57.75%. These results demonstrate the effectiveness of this architecture for the real-time rank estimation problem.

As future work, transitioning from the analysis of static instances to modeling through sliding windows of moves is recommended, allowing the model to capture the tactical flow of the match. Additionally, exploring multimodal architectures presents itself as a promising vector. As suggested by Kuboki et al. (2025), incorporating embeddings derived from deep models designed to play Go (such as KataGo or AlphaGo) can substantially enrich board representations, providing the predictor with greater resilience against unconventional opening strategies.

In the long term, the methodological evolution of this research enables the deployment of the predictive model as the core engine of an online platform, assisting in real-time estimation. This empowers the system to actively adapt its complexity and pedagogical style at each move to mitigate *sandbagging* practices, while simultaneously visually justifying its tactical decisions on the board, providing an immersive learning experience perfectly calibrated to the user throughout their entire journey to mastery.

Acknowledgments

The authors would like to acknowledge the support of the Brazilian Ministry of Science, Technology and Innovations (MCTI), the Manna Team, Fundação Araucária, Softex, CNPq (Grant Number 421548/2022-3), and the Research Support Foundation of the State of Minas

Gerais (FAPEMIG), and CAPES. This work is also financially supported by VinUniversity under Grant No CECS.UNI.2026.DA014. The authors would also like to thank Haoran Peng for useful comments.

References

Bishop, C. M.; and Svensén, M. 2012. Bayesian Hierarchical Mixtures of Experts. *CoRR*, abs/1212.2447.

Chen, C.-J.; Shih, C.-C.; and Wu, T.-R. 2025. Strength Estimation and Human-Like Strength Adjustment in Games. In *International Conference on Learning Representations*.

Chen, T.; and Guestrin, C. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, 785–794. New York, NY, USA: Association for Computing Machinery. ISBN 9781450342322.

Elo, A. E. 2008. *The Rating of Chessplayers: Past and Present*. New York: Ishi Press International, 2nd edition. ISBN 978-0923891275.

Fujita, K. 2021. AlphaDDA: Strategies for Adjusting the Playing Strength of a Fully Trained AlphaZero System to a Suitable Human Training Partner. *arXiv e-prints*, arXiv:2111.06266.

Gibson, B.; and Sachau, D. 2000. Sandbagging as a Self-Presentational Strategy: Claiming to be Less than You Are. *Personality and Social Psychology Bulletin*, 26(1): 56–70.

Glickman, M. E. 2012. Example of the Glicko-2 system. Technical report, Boston University.

Irsoy, O.; and Alpaydin, E. 2018. Dropout Regularization in Hierarchical Mixture of Experts. *CoRR*, abs/1812.10158.

Kang, H.; Suh, C.; and Kim, H. K. 2024. Match experiences affect interest: Impacts of matchmaking and performance on churn in a competitive game. *Heliyon*, 10(3): e24891.

Kuboki, K.; Ogawa, T.; Hsueh, C.-H.; Yen, S.-J.; and Ikeda, K. 2025. Policies of Multiple Skill Levels for Better Strength Estimation in Games. arXiv:2505.00279.

Li, W.; Wang, D.; Ding, Z.; Sohrabizadeh, A.; Qin, Z.; Cong, J.; and Sun, Y. 2025. Hierarchical Mixture of Experts: Generalizable Learning for High-Level Synthesis. arXiv:2410.19225.

Mccauley, B. 2023. Smurfs, silvers & CS:GO: Understanding smurfing as prosumers. In *Proceedings of the Annual Hawaii International Conference on System Sciences*. Hawaii International Conference on System Sciences.

Pascutto, G.-C. 2017. Leela Zero. https://github.com/leela-zero/leela-zero.

Shin, M.; Kim, J.; and Kim, M. 2021. Human Learning from Artificial Intelligence: Evidence from Human Go Players' Decisions after AlphaGo. In *Proceedings of the Annual Meeting of the Cognitive Science Society*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587): 484–489.

Wu, D. J. 2020. Accelerating Self-Play Learning in Go. arXiv:1902.10565.

Zhang, D.; Song, J.; Bi, Z.; Song, X.; Yuan, Y.; Wang, T.; Yeong, J.; and Hao, J. 2025. Mixture of Experts in Large Language Models. arXiv:2507.11181.