WHEN ALPHAGO THINKS: MASTERING GO WITH AN LLM-BASED REASONING SYSTEM

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning LLMs perform well on many mathematics and coding tasks, but extending these capabilities to specialized domains with scarce textual corpora remains challenging. In Go, recent models have improved move prediction while still producing inaccurate professional terminology. We develop an LLM-based reasoning system that combines a candidate sampler with a text-based reward model. Experiments across model and data scales motivate separate training for candidate generation and selection. We also introduce Term-DPO, a preference data construction method that uses progressive terminology replacement and rewriting to expand approximately 5,000 expert-annotated examples into 500,000 preference pairs. On KataGo-Eval-Levelwise, the system achieves a top-eight reference-move hit rate of 92.1%, compared with 82.8% for Single LLM. In full games, the system achieves closely matched results against specialized KataGo models. It also wins 73 of 81 games against nine professional players ranked 2 dan to 9 dan, with 30 minutes per player and three 20-second byo-yomi periods. Term-DPO achieves 87.4% terminology precision while maintaining comparable move prediction accuracy. These results demonstrate improved playing performance and terminology reliability in LLM-generated Go analyses.

1 INTRODUCTION

Recent advances in RLVR have enabled general-purpose reasoning LLMs to approach or surpass human experts on common benchmarks in mathematics, code, and agentic tasks. However, these advances remain difficult to transfer to specialized domains with scarce textual corpora.

Go provides a representative example. Specialized neural networks have mastered the game since AlphaGo (Silver et al., 2016; 2017), yet general-purpose reasoning LLMs remain substantially weaker and may fail to complete a full game without illegal moves. This contrast suggests that strong general reasoning performance does not directly imply competence in specialized tasks.

Go positions and strategic analyses are rare in general pre-training corpora, making it difficult for LLMs to acquire professional Go knowledge during pre-training. Ma et al. (2026) transfers general reasoning patterns to Go through cold-start SFT and RL, enabling an LLM to approach professional human performance.

Our training pipeline builds on this cold-start SFT and RL framework, with modifications to its training rewards. Within a candidate-generation and selection framework, we develop task-specific training objectives for the sampler and reward model, together with Term-DPO for terminology accuracy. These designs address two limitations.

First, candidate generation and final move selection pose distinct challenges. Across the tested model and data scales, generating more candidates does not consistently improve final move selection, while dedicated sampling achieves higher candidate coverage. These results motivate a dual-model system in which a sampler proposes candidate moves and a text-based reward model selects among them. The resulting system improves the benchmark hit rate over the tested single-model baseline and achieves near-even game results against KataGo-256c and 384c at 4,096 visits per move. We further analyze alternative reward model designs and the effect of explicit reasoning.

Second, the model of Ma et al. (2026) still uses professional Go terminology inaccurately. Its RL reward evaluates the final move without explicitly supervising the accompanying analysis. Expert

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.

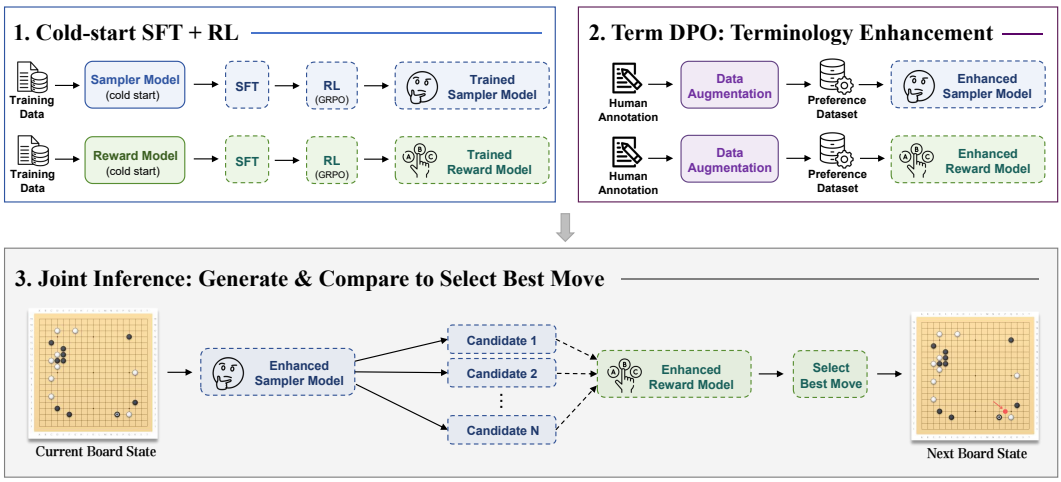


Figure 1: Dual-model framework with term-aware preference optimization. The sampler generates candidate moves, and the reward model selects the best one after cold-start SFT, RL, and Term-DPO.

annotation provides direct supervision for terminology use, but requires manual review of each analysis. We therefore propose Term-DPO, a terminology-oriented data augmentation method that expands approximately 5,000 human-annotated samples into nearly 500,000 preference pairs. Term-DPO substantially improves terminology accuracy while preserving move prediction performance.

In summary, this paper makes the following contributions:

1. We propose an LLM-based reasoning system that achieves near-even game results against KataGo-256c and 384c at 4,096 visits per move while providing board-state analyses with professional Go terminology.
2. We study candidate coverage and selection accuracy across model and data scales and evaluate separate training objectives for these tasks.
3. We propose Term-DPO, which expands a small set of human annotations into large-scale preference pairs and improves terminology accuracy while maintaining comparable move prediction accuracy.
4. We construct KataGo-Eval-Levelwise using 32 KataGo and HumanSL reference models, providing a level-wise benchmark for evaluating Go-capable LLMs.

2 Method

Figure 1 illustrates the training stage of our reasoning system and its inference strategy. Given a Go board state, the sampler model generates candidate next moves. The reward model selects one candidate as the next move.

Both the sampler and Text RM undergo cold-start SFT, RL, and Term-DPO. The following sections describe their task-specific designs and training objectives.

2.1 How to Model the Game of Go

We first briefly describe how we model a Go game. Go is a two-player, perfect-information, zero-sum game. The players take turns placing black and white stones on the board intersections, with the outcome determined by the applicable scoring rules.

In our formulation, we follow Ma et al. (2026) and represent the current position as a sequence. Specifically, for a Go position after k moves, let the preceding move sequence be denoted by $\{x_n\}_{n\leq k}$. Let s_k denote the corresponding board array. The model predicts the next move as

$$x_{k+1} = \pi_\theta(x_1, x_2, \ldots, x_k, s_k). \tag{1}$$

In the input, we use `X` and `O` to distinguish the black and white players, respectively. Meanwhile, we employ tools to represent the actual board state as a 19×19 two-dimensional array, where 1 denotes a black stone, -1 denotes a white stone, and 0 denotes an empty intersection.

Therefore, for a given game, the query takes a form similar to the following:

> **The move list is:**
> 1. X–Q16 2. O–D16 ...
> **The current board state is:**
> `[1, -1, -1, -1, 0, ...]`
> *(a 19×19 two-dimensional array)*
> **Please predict the next move.**

2.2 Data Preparation

KataGo-Eval-Levelwise We select 32 reference models (Wu, 2020), comprising 27 KataGo-HumanSL models and five specialized KataGo models, denoted by $\{\pi_i\}_{i=1}^{32}$. We sample 1 million board positions from professional games played from 1949 to the present and evaluate all 32 models on these positions.

KataGo (kata1-b28c512nbt), denoted by π^*, provides reference annotations. Let $\mathcal{A}_8(s)$ contain its eight highest-ranked moves at position s. An LLM prediction a counts as a hit if $a \in \mathcal{A}_8(s)$ and $w^*(s) - w(s,a) < 0.20$, where $w^*(s)$ and $w(s,a)$ are reference win probabilities for the best and predicted moves from the current player's perspective.

We rank the reference models by their top-eight agreement rates. To cover transition regions between adjacent capability levels, we sample positions from the corresponding hit-rate intervals and form several groups. We select 774 positions covering a range of difficulty levels to construct KataGo-Eval-Levelwise.

Professional Data Synthesis Next, we sample approximately 50 million positions from 2 million human and AI games and annotate them with the reference KataGo model π^*. Each annotated example contains the best next move, several plausible candidate moves, and corresponding analyses. The analysis includes possible continuations, the win rate associated with each candidate, and other relevant information. These data are subsequently used to construct synthetic expert data for supervised fine-tuning (SFT). Training and evaluation use disjoint game records and are fully deduplicated at the board-position level. None of the 774 benchmark positions occurs in our training data.

2.3 LLM as a Policy Model in Go

We begin by investigating the extent to which an LLM can serve as a policy model for Go.

LLM as a Sampler and Selector. We first construct a set of chain-of-thought (CoT) supervision data from KataGo annotations, and use it to fine-tune the model. Given a board position, the model is trained to first propose several candidate moves, then predict a possible continuation for each candidate, estimate the corresponding win rate, and finally select the best move. In this way, the LLM is expected to act jointly as a sampler over candidate moves and as a selector based on downstream evaluation. An example response is shown below:

> **Candidate moves:** `D4` and `Q16`
> ***Candidate 1:*** `D4`
> Predicted continuation: `D4 R16 Q4 C14 P17 Q15 M17 C3 D3 C4 D5 B6`
> Estimated White win rate: 65.40%
> ***Candidate 2:*** `Q16` ...
> **Recommended next move**
> `D4`

LLM as a Sampler Only. We next explore a decoupled strategy that separates candidate generation from move evaluation. Specifically, we decompose next-move prediction in Go into two components within an agentic system: a *Sampler* and a *Reward Model*. For the policy model, we train the LLM solely as a sampler, whose task is to generate several plausible next moves for a given game state.

2.4 LLM AS A REWARD MODEL IN GO

Reasoning LLM as a reward model Like the sampler, Text RM undergoes cold-start SFT, RL, and Term-DPO. It compares candidate moves through textual predictions of their continuations and win rates, then selects the preferred move. Training examples take the following form.

> ***Candidate 1:*** `L13...` ***Candidate 2:*** `K12 ...`
> **Recommended next move:** `L13`

LLM as a Classifier As a comparison, following prior works on other board games (Farebrother et al., 2024; Schultz et al., 2025), we add 64 special tokens to the vocabulary, each corresponding to a uniformly partitioned win-rate interval. Given a move, the reward model predicts one of these special tokens, thereby mapping the win-rate estimate to a discrete bucket.

At inference time, the reward model outputs a distribution over these bucket tokens. To obtain a scalar win-rate prediction, we extract the logits of all bucket tokens and normalize them into a probability distribution. The final predicted win rate is then computed as the probability-weighted average of the representative win rates of all buckets:

$$p_i = \frac{\exp(\ell_i)}{\sum_{j=1}^{N} \exp(\ell_j)}, \qquad \hat{w} = \sum_{i=1}^{N} p_i w_i, \tag{2}$$

where $N = 64$, ℓ_i denotes the logit of the i-th bucket token, p_i is the corresponding normalized probability, and w_i is the representative win rate of that bucket, defined as the midpoint of the interval.

2.5 MIXING DOMAIN KNOWLEDGE WITH NATURAL REASONING

Since the policy model in Section 2.3 is trained on synthetic data, its generated chain-of-thought is often unnatural. Following (Ma et al., 2026), we perform mixed cold-start training using long-CoT reasoning data in general domains. Specifically, we distinguish the system prompts used for synthetic data from those used for reasoning-style data, so that the model learns to produce natural and fluent reasoning under reasoning prompts.

We then optimize the model on the move prediction task using GRPO. This stage enables the model to maintain strong predictive performance while producing more natural CoT outputs.

Reward function. For the policy model and sampler, we use a task-specific reward of the form

$$R = \mathbb{I}[\text{format correct}] \cdot \big(R_{\text{rank}} - \alpha R_{\text{opt}}\big),$$

where R_{rank} rewards move quality based on reference rank, and R_{opt} penalizes suboptimal moves relative to the best move.

For the sampler, we set $\alpha = 0$ and compute R_{rank} using a piecewise function of the overlap proportion between predicted moves and $\mathcal{A}_8(s)$. Each distinct matching move is counted once, and the denominator is the required candidate count. For the policy model, which outputs a single finalized move, we instantiate R_{rank} as a rank-based reward on the selected move and define

$$R_{\text{opt}} = \frac{\beta(w^* - w)}{1 + \beta(w^* - w)},$$

Here $w, w^* \in [0, 1]$ are the reference win rates of the selected and optimal moves. For the policy model, $\alpha = 0.5$ and $\beta = 0.2$. Text RM receives the fraction of comparable candidate pairs whose predicted ordering agrees with reference KataGo, with zero reward for invalid output formats. The complete reward definitions are provided in Appendix E.2.

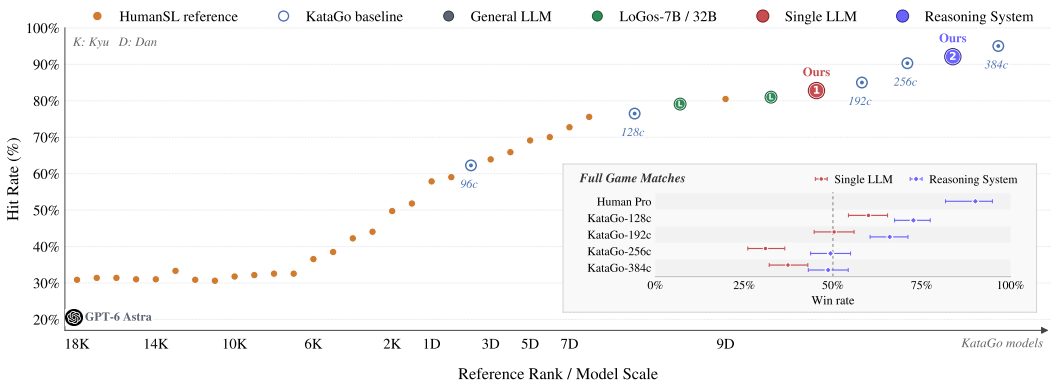


Figure 2: Performance on KataGo-Eval-Levelwise and in full games. The main plot shows position-level hit rates. The inset compares both systems over 300 games per engine matchup. Human Pro reports Reasoning System over 81 games pooled across professional players. Bars denote 95% Wilson confidence intervals, and the dashed line marks 50%.

2.6 Term-DPO for Terminology Alignment

Ma et al. (2026) proposes a framework for training a chain-of-thought reasoning LLM for Go. However, the proposed LLM remains inaccurate in its use of domain-specific Go terminology.

To this end, we recruit 50 strong amateur Go players, whose average playing strength is at the top amateur level, to edit the responses generated by the policy model, correct erroneous terminology, and add missing expressions. In total, this process yields approximately 5,000 manually annotated examples. We use these expert corrections as the basis for constructing additional terminology preferences.

We propose **Term-DPO**, a data-augmentation-based DPO method that expands these 5,000 manually annotated examples into approximately 500,000 synthetic preference pairs. This augmented dataset substantially improves the model's accuracy in generating Go-specific terminology.

For an expert-edited response containing n replaceable Go terms, let the response before terminology perturbation be denoted by $x^{(0)}$. We maintain a vocabulary of approximately 300 common Go terms, denoted by $\{T_i\}_{i=1}^{N}$, and replace the terms in each sample one by one according to a fixed order, yielding a sequence

$$x^{(0)} \to x^{(1)} \to \cdots \to x^{(n)},$$

where $x^{(k)}$ denotes the sample after k terms have been replaced. We refer to this sequence as a **progressive replacement chain**. For $i < j$, $x^{(i)}$ contains fewer term replacements than $x^{(j)}$ and is expected to retain more accurate terminology. To preserve fluency after replacement, we further rewrite each $x^{(k)}$ using GPT-5.1 and denote the rewritten sample by $\tilde{x}^{(k)}$. We use the replacement order to assign preference labels to the rewritten sequence

$$\tilde{x}^{(0)} \to \tilde{x}^{(1)} \to \cdots \to \tilde{x}^{(n)}.$$

Any pair $\left(\tilde{x}^{(i)}, \tilde{x}^{(j)}\right)$ with $i < j$ can then be used to construct a preference pair, where $\tilde{x}^{(i)}$ is treated as the preferred response and $\tilde{x}^{(j)}$ as the dispreferred one. Each annotated sample yields $\binom{n+1}{2}$ possible pairs. We downsample the resulting pool to approximately 500,000 pairs for DPO training. Pairs from the same source provide related training comparisons.

3 Experiments

3.1 Experimental Setups

Dataset For data collection, we gather a total of 2 million human and AI game records. We randomly sample board states from these games and annotate them using the kata1-b28c512nbt

Inference Mode	Mean Tokens	First-candidate hit ↑	Oracle@2 ↑	System Acc. ↑
No Reasoning	8	75.84	84.40	80.32
Short CoT	189	80.49	92.25	89.79
Full CoT	1,044	**82.80**	**94.83**	**92.10**
Single LLM (Full CoT), self-selection				82.80
Single LLM (Full CoT), Text RM selection				87.72
Single LLM (Full CoT), Oracle@2				90.44

Table 1: Effect of sampler reasoning on candidate coverage and final selection. The upper rows use the dedicated sampler; the lower rows share two candidates from one Single LLM response. Oracle@2 reports their coverage; other rows report final selection accuracy. Hits follow Section 2.2.

model, resulting in annotated data for 50 million board states. For reasoning data, we mainly distill part of the reasoning data from LoGos-7B introduced by Ma et al. (2026), and further mix reasoning data from sources such as GLM-5.1-Reasoning-1M-Cleaned (Kassadin88, 2026) and KIMI-K2.5-1000000x to supplement the model's reasoning ability.

Training Details We use XTuner (Contributors, 2023) for SFT and VeRL (Sheng et al., 2024) with GRPO for RL. SFT uses a learning rate of 2×10^{-5}, a global batch size of 64, and sequence packing up to 16,384 tokens. RL uses 8 rollouts per input, a batch size of 16, a maximum generation length of 16,384 tokens, a learning rate of 10^{-6}, and clip ratios $[0.2, 0.28]$. Term-DPO uses $\beta_{\text{DPO}} = 0.1$, a learning rate of 10^{-6}, and a global batch of 64 preference pairs for one epoch (Appendix F). Base models are from the Qwen3 series (Yang et al., 2025), with 1B, 2B, 4B, and 8B parameters.

3.2 Performance of the Reasoning System

Single LLM denotes the Qwen3-8B policy model after cold-start SFT, RL, and Term-DPO. Reasoning System combines a separately trained sampler and Text RM. Figure 2 compares them with LoGos, a general-purpose reasoning baseline, and the reference engines on benchmark and full-game performance.

3.2.1 Benchmark Details

LLM hit rates require top-eight membership and a reference win-rate loss below 20 percentage points. Horizontal positions organize the reference models by their designated ranks or network sizes. KataGo-HumanSL models provide reference ranks from 18K to 9D, where K and D denote kyu and dan. We also include KataGo-96c, 128c, 192c, 256c, and 384c. HumanSL ranks label imitation models and do not assign human ranks to the evaluated LLMs. Appendix C details benchmark construction and supplementary top-one and score-loss comparisons.

3.2.2 Benchmark and Tournament Results

The reasoning system improves the hit rate from 82.8% for Single LLM after Term-DPO to 92.1%. This gain comes with additional inference cost, while two independent Single LLM generations provide little improvement despite using more tokens and time than the system (Table 7).

Full-game matches show that the improvement extends to playing performance. The system wins 65.7% of 300 games against Single LLM. Against KataGo opponents using 4,096 visits per move, it wins a majority of games against 128c and 192c and achieves near-even results against 256c and 384c (Figure 2).

Games against professional players provide a further test of playing strength. The system wins 73 of 81 games (90.1%) against nine professionals ranked 2 dan to 9 dan. These games use Chinese rules without handicap, random colors, and 30 minutes per player with three 20-second byo-yomi periods. Appendix C.3 reports individual player records, tournament protocols, and Elo ratings.

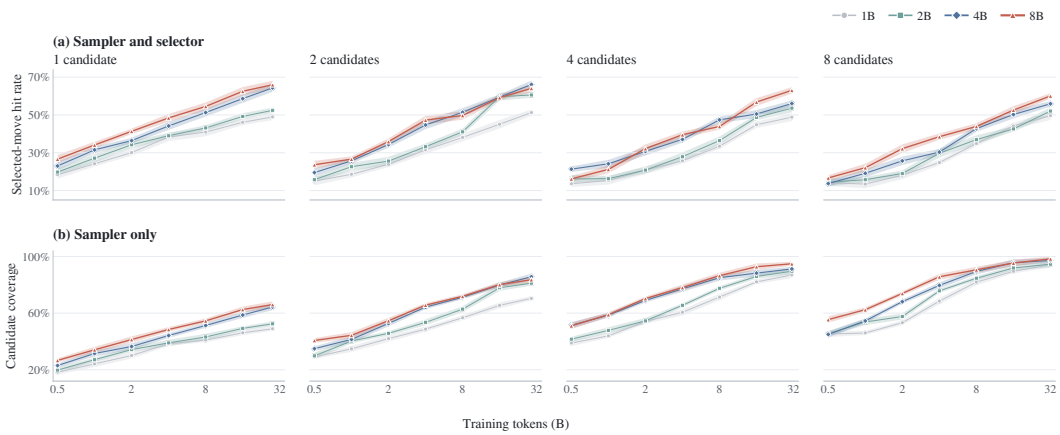


Figure 3: Scaling with training data and model size. Panel (a) measures the final selected-move hit rate; panel (b) measures candidate coverage. Curves show means over three independent training runs; shaded bands denote pointwise 95% t confidence intervals. Both metrics follow Section 2.2.

3.3 Scaling Experiments on the Policy Model

We vary model size, training data, and candidate count to compare joint generation and selection with candidate generation alone. Both evaluations use the hit criterion in Section 2.2.

3.3.1 LLM as a Sampler and Selector

The model proposes candidates, analyzes their continuations and win rates, and selects a move within one response. We score the selected move. Larger models generally achieve higher hit rates, but increasing the candidate count does not consistently improve final selection (Figure 3(a)).

3.3.2 LLM as a Sampler Only

The model generates candidates without selecting among them. We measure coverage, counting a hit when any candidate satisfies the criterion. This bounds the accuracy attainable by a selector restricted to these candidates. Coverage increases with multiple candidates (Figure 3(b)), motivating separate training for generation and selection.

Effect of explicit reasoning. We vary the final 8B sampler's reasoning budget on the same evaluation set under greedy decoding. Each response contains two candidates. No Reasoning outputs only the moves, Short CoT limits analysis to 256 tokens, and Full CoT uses the standard setting. Token counts include the candidate coordinates.

Table 1 shows higher candidate coverage and system accuracy with longer reasoning outputs. Short CoT recovers most of the gain over direct prediction, with a smaller further increase under Full CoT. These settings vary reasoning length and computation together.

3.4 Modeling Reward with LLMs

Using a subset of the training data, we compare Qwen3-8B Text RM and Bucket RM trained on equal numbers of board states. Both select from identical candidate pairs across sampler checkpoints (Figure 4). Policy First Hit takes the first candidate, while Oracle accepts either candidate satisfying the hit criterion.

Text RM outperforms Bucket RM, which remains close to taking the first candidate. It also improves on Single LLM's self-selection when given the same candidate pairs (Table 1). Text RM remains similarly close to the coverage ceiling for both generators. The dedicated sampler offers higher coverage and, with Text RM, the highest final hit rate.

Method	Coverage ↑	Precision ↑	Valid/Resp. ↑	Major Error ↓	Move Acc. ↑
External baseline					
LoGos-7B	29.0	44.2	1.02	23.0	79.10
Qwen3-8B variants					
Qwen3-8B	3.0	4.5	0.04	3.0	2.58
+ CS	24.0	40.8	0.31	22.0	52.07
+ CS + RL	25.0	42.6	0.34	25.0	**82.91**
+ SFT-5K	27.0	44.4	0.60	22.0	81.40
+ DPO-5K	36.0	53.3	1.03	18.0	82.65
+ DPO-Repeat	37.0	55.1	1.35	17.0	81.92
Term-DPO ablations					
+ Term-DPO w/o rewriting	60.0	62.4	2.05	3.0	82.60
+ Term-DPO, original-only	45.0	54.1	1.50	16.0	81.80
+ Term-DPO	**99.0**	**87.4**	**5.46**	**1.0**	82.80

Table 2: Terminology reliability on 100 positions and move accuracy on KataGo-Eval-Levelwise. Coverage and Major Error measure responses with any valid term or major error, respectively (%). Precision measures valid terms among those used (%); Valid/Resp. is their mean count per response. Repeated terms count once per response. CS denotes cold start. SFT-5K uses corrected responses. DPO-5K pairs expert corrections with original model responses; DPO-Repeat resamples these pairs to match Term-DPO's updates. No rewriting omits GPT rewriting. Original-only retains rewriting but pairs the original response only with each perturbed version.

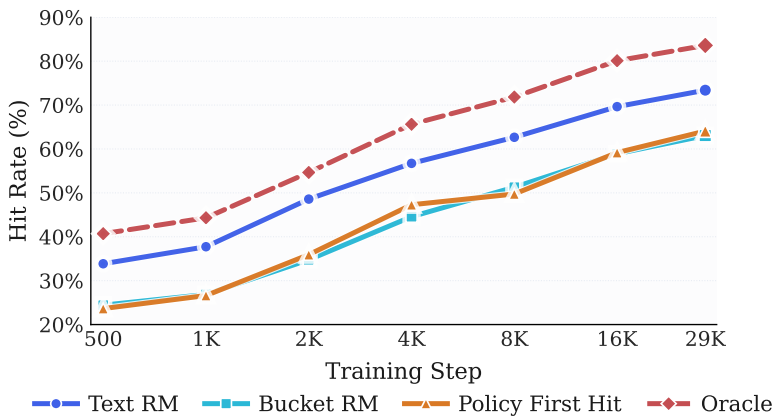


Figure 4: Comparison of text-based and classifier-based reward models across sampler training stages in an experiment using a subset of the training data.

3.5 Term-DPO for Reliable Terminology Use

We evaluate Term-DPO through Single LLM ablations; the full system also applies it to both the sampler and Text RM. We assess terminology reliability on 100 board states using 50 independently recruited amateur Go players, separate from the Term-DPO training annotators. All Go terms in each response are evaluated for correctness and relevance, with model identities hidden. Appendix D details the voting protocol. SFT-5K and DPO variants start from CS+RL. Both ablations match Term-DPO's hyperparameters and updates.

Term-DPO achieves 87.4% terminology precision while maintaining move accuracy, outperforming SFT-5K and both conventional DPO baselines (Table 2). Removing rewriting or restricting preferences to original-versus-perturbed pairs reduces precision and coverage and increases major errors, supporting both components of preference construction. A separate label check finds that ten professional or amateur players prefer the positive response in 87.5% of 200 judgments over 100 pairs.

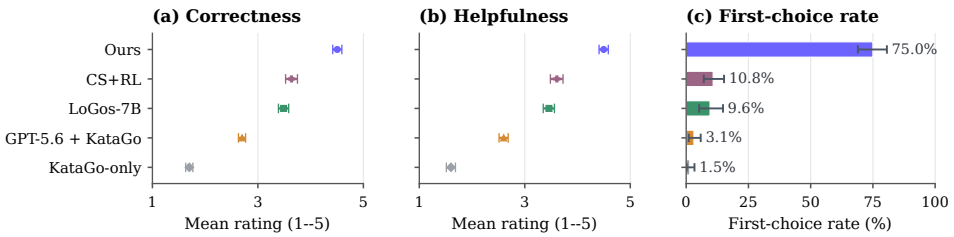


Figure 5: Human evaluation of analysis correctness, helpfulness, and preference. Ours denotes Single LLM after Term-DPO; CS+RL denotes cold-start SFT followed by RL. Error bars show 95% bootstrap confidence intervals.

3.6 Human Evaluation of Analysis Utility

We assess analysis utility with 12 professional Go players on 40 positions. With model identities hidden, they rate correctness and helpfulness from 1 to 5 and select their preferred output. Correctness requires both reasonable variations and sound Go terminology and positional analysis. Term-DPO improves both ratings over CS+RL and receives 75.0% of first choices across five methods (Figure 5), connecting terminology gains with the usefulness of the full analysis. Appendix D details evaluation and bootstrap procedures.

4 Related Work

4.1 Reinforcement Learning with Verifiable Rewards

GRPO (Shao et al., 2024) supports RL training for reasoning LLMs. RLVR has improved mathematical, coding, and general reasoning capabilities (Hui et al., 2024; Shen et al., 2025; Team et al., 2025; Yang et al., 2025), with extensions to tool interaction (Qian et al., 2025; Feng et al., 2025; Song et al., 2025; Li et al., 2025a;b) and optimization variants targeting efficiency and stability (Yu et al., 2025; Zheng et al., 2025; Zhao et al., 2025).

4.2 LLMs for Games

LLM agents have been studied in Minecraft (Wang et al., 2023) and Pokemon (Hu et al., 2024), while Feng et al. (2023) train language models for chess. Schultz et al. (2025) compare external search and internal reasoning in board games. In Go, Ma et al. (2026) transfer general reasoning capabilities through RLVR. We extend this direction by separately training candidate generation and evaluation, and by supervising the terminology used in accompanying analyses.

4.3 Verification and Preference Learning

Generating candidates and selecting them with a learned verifier is an established approach (Cobbe et al., 2021). Our system applies this framework to Go through coverage rewards for the sampler and pairwise ranking rewards for Text RM. DPO learns directly from preference pairs (Rafailov et al., 2023), while Step-DPO uses individual reasoning steps as preference units (Lai et al., 2024). Term-DPO retains the DPO objective and constructs response-level preferences from progressively perturbed terminology. Its contribution lies in using expert corrections to form comparisons across replacement stages, with rewriting to preserve fluency.

5 Conclusion

We present an LLM-based reasoning system that produces natural-language analyses and achieves near-even game results against KataGo-256c and 384c at 4,096 visits per move. Candidate coverage and selection results across the tested scales motivate separate sampler and reward models. Term-DPO improves terminology accuracy under scarce textual supervision without materially degrading move prediction, while KataGo-Eval-Levelwise provides a difficulty-controlled evaluation of Go-capable LLMs.

AI USE STATEMENT

Generative AI tools were used to assist with language polishing, presentation, and feedback on experimental reporting; all research decisions, results, and AI-assisted edits were independently reviewed and verified by the authors, who take full responsibility for the final content.

REPRODUCIBILITY STATEMENT

The method and training procedure are described in Section 2, with further implementation and evaluation details provided in the appendix. We will release the data and model checkpoints.

REFERENCES

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

XTuner Contributors. Xtuner: A toolkit for efficiently fine-tuning llm. https://github.com/InternLM/xtuner, 2023.

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, Aviral Kumar, and Rishabh Agarwal. Stop regressing: Training value functions via classification for scalable deep rl, 2024. URL https://arxiv.org/abs/2403.03950.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025.

Xidong Feng, Yicheng Luo, Ziyan Wang, Hongrui Tang, Mengyue Yang, Kun Shao, David Mguni, Yali Du, and Jun Wang. Chessgpt: Bridging policy learning and language modeling, 2023. URL https://arxiv.org/abs/2306.09200.

Sihao Hu, Tiansheng Huang, and Ling Liu. Pokellmon: A human-parity agent for pokemon battles with large language models, 2024. URL https://arxiv.org/abs/2402.01118.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Kassadin88. Glm-5.1-1000000x: One million reasoning traces distilled from glm-5.1, 2026. URL https://huggingface.co/datasets/Kassadin88/GLM-5.1-1000000x.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms, 2024. URL https://arxiv.org/abs/2406.18629.

Peiji Li, Jiasheng Ye, Yongkang Chen, Yichuan Ma, Zijie Yu, Kedi Chen, Ganqu Cui, Haozhan Li, Jiacheng Chen, Chengqi Lyu, et al. Internbootcamp technical report: Boosting llm reasoning with verifiable task scaling. *arXiv preprint arXiv:2508.08636*, 2025a.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*, 2025b.

Yichuan Ma, Linyang Li, Yongkang Chen, Peiji Li, Jiasheng Ye, Qipeng Guo, Dahua Lin, and Kai Chen. Mixing expert knowledge: Bring human thoughts back to the game of go, 2026. URL https://arxiv.org/abs/2601.16447.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2023. URL https://arxiv.org/abs/2305.18290.

John Schultz, Jakub Adamek, Matej Jusup, Marc Lanctot, Michael Kaisers, Sarah Perrin, Daniel Hennes, Jeremy Shar, Cannada Lewis, Anian Ruoss, Tom Zahavy, Petar Veličković, Laurel Prince, Satinder Singh, Eric Malmi, and Nenad Tomašev. Mastering board games by external and internal planning with language models, 2025. URL https://arxiv.org/abs/2412.12119.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, et al. Vlm-r1: A stable and generalizable r1-style large vision-language model. *arXiv preprint arXiv:2504.07615*, 2025.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL https://arxiv.org/abs/2305.16291.

David J. Wu. Accelerating self-play learning in go, 2020. URL https://arxiv.org/abs/1902.10565.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Yuzhong Zhao, Yue Liu, Junpeng Liu, Jingye Chen, Xun Wu, Yaru Hao, Tengchao Lv, Shaohan Huang, Lei Cui, Qixiang Ye, Fang Wan, and Furu Wei. Geometric-mean policy optimization, 2025. URL https://arxiv.org/abs/2507.20673.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025. URL https://arxiv.org/abs/2507.18071.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs, 2024. URL https://arxiv.org/abs/2312.07104.

A LIMITATIONS

Our experiments focus on Go. Applying Term-DPO to another domain would require a domain-specific terminology resource, expert judgments of correctness and relevance, and checks that perturbation and rewriting preserve the intended preference relation. Such transfer has not been evaluated here. The ablations examine rewriting and comparisons between replacement stages, while sensitivity to the replacement order remains to be studied. Our results against specialized models apply to the evaluated search and decoding settings. The human-match win rate is pooled across the tested professional players. Professional judgments cover variations, terminology, and positional analysis. Numerical win-rate calibration and automatic move-by-move verification of generated variations remain unevaluated.

B DEMOS

B.1 PROMPT DEMONSTRATIONS

We next briefly present the data formats used to construct different model components. We first show the system prompts used by different models, including the prompts provided to the policy model, the sampler model, and the reward model. In practice, the prompts we use are written in Chinese. For ease of understanding, we translate them into English for presentation here.

Prompt for the Reward Model

```
You are a professional Go player, and you are skilled at making
    accurate positional judgments. Based on the given position, you
    need to analyze the strengths and weaknesses of the next move in a
    structured format.

In the game record, "X" denotes black stones and "O" denotes white
    stones. The board size is 19x19. Each move coordinate consists of
    one letter and one number. The letters range from A to T, skipping
    I, and correspond to the board from left to right. The numbers
    range from 1 to 19 and correspond to the board from bottom to top.

Please analyze the resulting position after each candidate move in a
    structured format, and provide the score lead and win rate.
```

Prompt for Next-Move Prediction

```
You are a professional Go player. You are now playing a game of Go.
    Based on the given board information, you need to make a
    structured prediction for the next move. Your task is to analyze
    the current position according to the given game record and select
    the single most likely next move.

In the given board, "X" denotes black stones and "O" denotes white
    stones. The board size is 19x19. Each move coordinate consists of
    one letter and one number. The letters range from A to T, skipping
    I, and correspond to the board from left to right. The numbers
    range from 1 to 19 and correspond to the board from bottom to top.

You should first analyze and reason about the current position, and
    then provide the most recommended next move. Your output format
    should be:

<reasoning>
Analysis and reasoning about the current position.

1. Move coordinate
</reasoning>
```

```
<answer>
1. Move coordinate
</answer>
```

Prompt for the Sampling Model

```
You are a professional Go player. You are now playing a game of Go.
    Based on the board information, you need to make vivid and
    interesting predictions, analysis, and reasoning about the next
    move. Your task is to analyze the current position according to
    the given game record and select two possible candidates for the
    next move.

In the given board, "X" denotes black stones and "O" denotes white
    stones. The board size is 19x19. Each move coordinate consists of
    one letter and one number. The letters range from A to T, skipping
    I, and correspond to the board from left to right. The numbers
    range from 1 to 19 and correspond to the board from bottom to top.

You should first provide reasonable analysis and reasoning about the
    current position, then predict, simulate, and analyze the
    following moves, and finally select the two most suitable
    next-move candidates. Your output format should be:

<reasoning>
Analysis and reasoning about the current position.

Move coordinate
</reasoning>

<answer>
1. Move coordinate
2. Move coordinate
</answer>
```

Input Example (for the Reward Model)

```
The current game record is as follows:

1.X-Q16, 2.O-D16, 3.X-R4, 4.O-D3, 5.X-D5, 6.O-P4, 7.X-N3, 8.O-R3,
    9.X-Q3, 10.O-Q4, 11.X-P3, 12.O-O3, 13.X-O4, 14.O-O2, 15.X-R2,
    16.O-S3, 17.X-P2, 18.O-R5, 19.X-N2

The current board state is:
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, -1, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 1, -1, -1, 1, 0, 0], [0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0,
    0, 1, -1, 1, 1, -1, -1, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

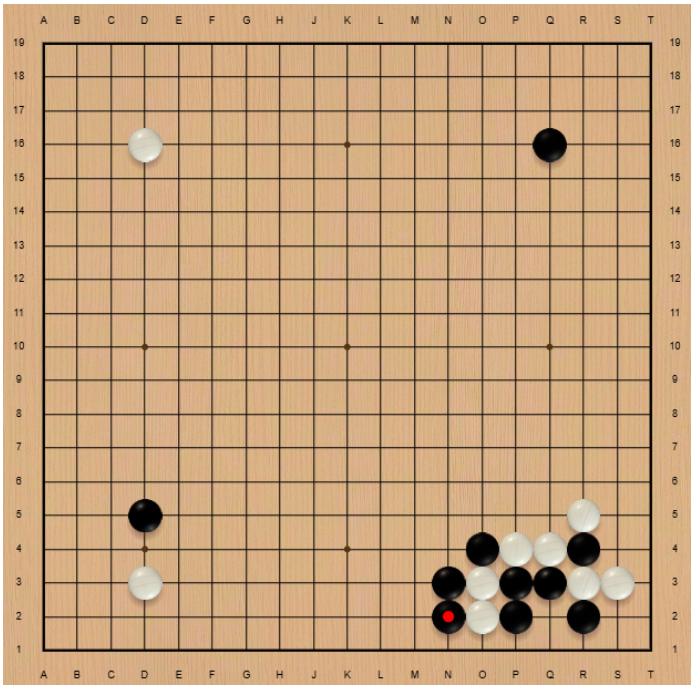


Figure 6: Board state relating to the input example.

```
    1, -1, 1, 0, 1, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0]]

Here, 1 denotes black stones, -1 denotes white stones, and 0 denotes
    empty points.

The candidate next moves are as follows:
1. 20.O-C5; 2. 20.O-F3

Please simulate each candidate move separately, provide the key
    subsequent variations with alternating black and white moves,
    report the corresponding win rate and score lead, and finally
    summarize the analysis and choose the best move.
```

The board position corresponding to the input example above is shown in Figure 6.

B.2 Response Demonstrations

Next, we will present the responses generated by different models in detail and show the corresponding board positions.

B.2.1 Reward Model Responses

Structured Textual Reward Model The first response below is a structured response generated by the reward model. The corresponding board state is shown in Figure 6.

Structured Response from the Reward Model

```
For the current position, we compare the candidates from White's
    perspective:
1) Candidate C5
The subsequent variation is: 20.O-C5 -> 21.X-O5
```

```
White is estimated to lead by 1.94 points, with a win rate of 77.50\%.
2) Candidate F3
The subsequent variation is: 20.O-F3
White is estimated to lead by 1.54 points, with a win rate of 73.93\%.
Summary: Considering both the win rate and the score lead, the best
    next move is C5.
<answer>
\boxed{Best next move: C5}
\boxed{Score lead: 1.94}
\boxed{Win rate for the next move: 77.50\%}
</answer>
```

Reasoning Textual Reward Model The example below illustrates the reward model's comparison of Q13 and S13 for the board state in Figure 7.

Illustrative Reward Model Response

```
<reasoning>
Analysis of candidate moves:

1. 53.X-Q13

This move may look ordinary, but it contains several subtle ideas:

- It first ensures the safety of Black's own connection.
- It creates a countermeasure against White's extension at P12.
- It prepares opportunities for the subsequent central fight.

A possible variation is:

53.X-Q13 magari (bend; Black reinforces its own shape)
54.O-P13 magari (bend; White continues to damage Black's shape)
55.X-Q12 tsugi (connection; Black completes the repair of the cut)

In this variation, Black's win rate is 53.8\%, making it a solid
    choice.

2. 53.X-S13

This move can be regarded as achieving three goals at once:

- It effectively restrains White's expansion in the upper-right corner.
- It protects the integrity of Black's shape on the right side.
- It creates conditions for a later counterattack.

The simulated variation is:

53.X-S13 hane (Black blocks White in the upper-right corner)
54.O-C12 placement (White may turn to the left side to open a new
    battlefield)
55.X-D12 nobi (stretch; Black strengthens the outside influence on the
    left)
56.O-B14 hane (White seeks territory in the upper-left corner)
57.X-C13 solid connection (Black connects into a unified group)

In this variation, Black's win rate is 51.4\%. Although the advantage
    is not large, the game is still long.

Q13 has the higher estimated win rate and is selected as the next move.
</reasoning>

<answer>
```

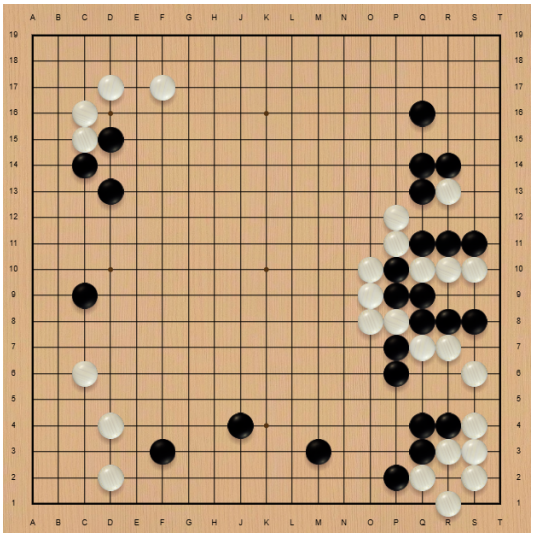


Figure 7: Board state relating to the response example from the reasoning reward model.

```
Q13
</answer>
```

The English translations of Go-specific terms in the following examples follow common usage in Go terminology references, including Go4Go, Sensei's Library, and GoMagic.[1]

B.2.2 Policy Model Responses

We next present the outputs of the policy model. Similar to the reward model, the policy model supports two output formats: structured outputs following a predefined format, and natural reasoning outputs in a reasoning-style format.

Structured Textual Policy Model. We first show a policy model response that follows the structured output format. The board state of this game is shown in Figure 8.

Structured Response from the Policy Model

```
<reasoning>
In the current position, the recommended next move is: N17
Recommended move: N17
Predicted subsequent variation: N17 C11 N18 G14 G13 F16 F15
At this point, Black's score lead is: -10.02 points.
Under the current variation, Black's win rate is: 1.35\%.
</reasoning>
<answer>
\boxed{Next move color: Black}
\boxed{Next move position: N17}
\boxed{Win rate for the next move: 1.35\%}
</answer>
```

Reasoning Policy Model. We then show a policy model response in the natural reasoning format. The board state is shown in Figure 9.

[1]We refer to Go4Go Chinese Go Terms: https://www.go4go.net/go/chinese_go_terms, Sensei's Library Go Terms: https://senseis.xmp.net/?GoTerms=, and GoMagic Glossary of Go Terms: https://gomagic.org/glossary-of-go-terms/.

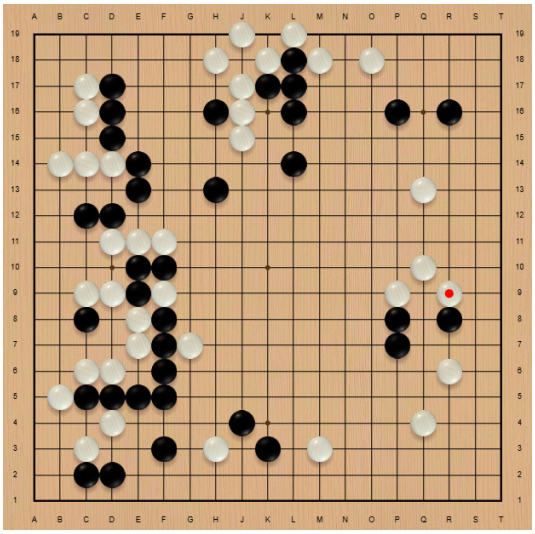


Figure 8: Board state relating to the response example from the structured policy model.

Reasoning Response from the Policy Model

```
<reasoning>
Let us carefully examine the current position. The game has reached
    move 109, and Black has just connected at S15, securing the eye
    space and connection of the large dragon in the upper-right area.
    As White, we now need to find a key point that can both apply
    pressure and take the whole-board situation into account.
1. 110.O-N18 descent:
This move can be regarded as a good point that achieves three goals at
    once. First, it coordinates with stones such as O17 and P18 to
    build thickness. Second, it restricts Black's development in the
    upper-right area. Most importantly, it prepares a follow-up
    invasion at M18. If Black responds inaccurately, White can even
    consider a hane at the head at T18, which would make Black rather
    uncomfortable.
The simulated variation is:
110.O-N18 descent (good shape!)
111.X-M15 jump out (expanding toward the center)
112.O-M18 block at the head (strong)
113.X-T10 placement inside the eye (tricky)
114.O-S10 connection (calm)
115.X-K15 warikomi (separating)
116.O-J15 hane (counterattack)
117.X-L15 stretch (heading out)
118.O-J14 support (making eye shape)
119.X-L13 cut (decisive move)
In this variation, White's win rate is 99.1\%, and the position is
    quite promising. Although Black can still live, Black's overall
    position will be severely compressed.
</reasoning>
<answer>
\boxed{Next move color: White}
\boxed{Next move position: N18}
\boxed{Win rate for the next move: 99.14\%}
</answer>
```

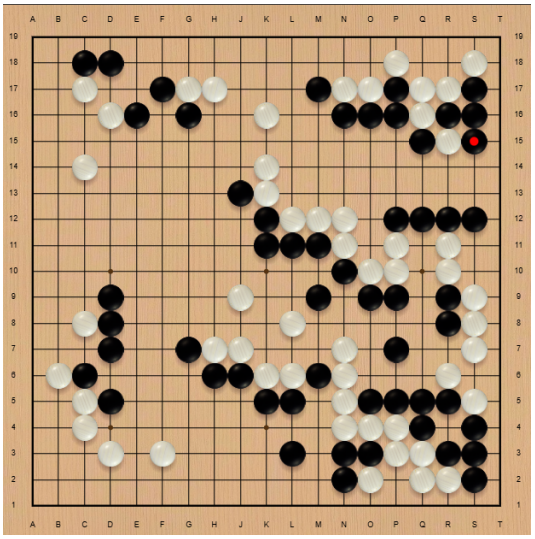


Figure 9: Board state relating to the response example from the reasoning policy model.

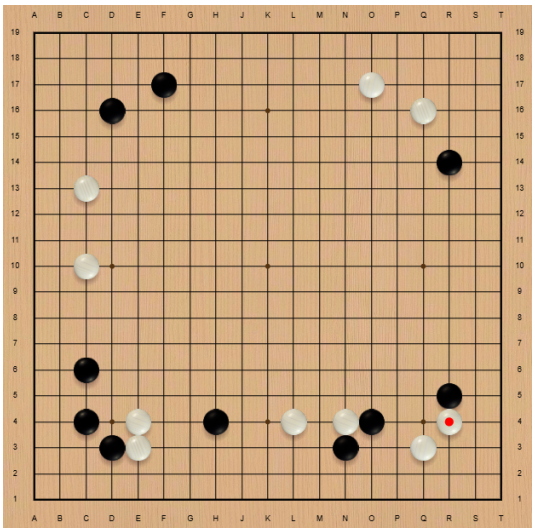


Figure 10: Board state relating to the example of Bucket RM.

B.2.3 BUCKET RM

Finally, we present the response of the bucket-based reward model mentioned in Section 2.4. Specifically, we first modify the tokenizer of the model by adding a set of special tokens, such as <WR_BIN_0>, <WR_BIN_1>, ..., and <WR_BIN_63>. Each special token represents a win-rate interval. With 64 equally divided bins, <WR_BIN_0> corresponds to the win-rate interval from 0.00% to 1.56%, while <WR_BIN_63> corresponds to the interval from 98.44% to 100.00%.

Therefore, for the position shown in Figure 10, the model prediction is as follows:

Response from the Bucket RM

```
<WR_BIN_19>
```

Here, <WR_BIN_19> corresponds to the win-rate interval from 29.69% to 31.25%.

B.3 Preference Pair Demos in Term-DPO

The following illustrative pair isolates a terminology replacement. The rejected response substitutes "ladder" for "atari", while all other text, moves, and win-rate estimates remain identical.

Chosen Response in Term-DPO

```
<reasoning>
White's move 84 is an atari at L2. Black must choose a response.

Black currently has two key candidate moves. The first choice is K3
    connection. This move directly responds to White's central
    pressure and coordinates with Black's stone at K2 from a distance.
    If White answers with a connection at M2, Black can take the
    opportunity to cut at G7 and gradually penetrate the center.
    Although the current win-rate estimate is 44.5\%, this is the
    strongest means of resistance.

The second choice is D5 connection. Although this move reinforces the
    lower-left area, it is somewhat passive. White can ignore it and
    occupy key points such as K3, causing Black's win rate to drop to
    38\%.

A possible variation is: 85.X-K3, 86.O-M2, 87.X-G7, 88.O-G6, 89.X-H5.
    In this way, Black can build a certain base in the center while
    keeping the potential in the upper-right area.
</reasoning>

<answer>
\boxed{Next move color: Black}
\boxed{Next move position: K3}
\boxed{Win rate for the next move: 44.5\%}
</answer>
```

Rejected Response in Term-DPO

```
<reasoning>
White's move 84 is a ladder at L2. Black must choose a response.

Black currently has two key candidate moves. The first choice is K3
    connection. This move directly responds to White's central
    pressure and coordinates with Black's stone at K2 from a distance.
    If White answers with a connection at M2, Black can take the
    opportunity to cut at G7 and gradually penetrate the center.
    Although the current win-rate estimate is 44.5\%, this is the
    strongest means of resistance.

The second choice is D5 connection. Although this move reinforces the
    lower-left area, it is somewhat passive. White can ignore it and
    occupy key points such as K3, causing Black's win rate to drop to
    38\%.

A possible variation is: 85.X-K3, 86.O-M2, 87.X-G7, 88.O-G6, 89.X-H5.
    In this way, Black can build a certain base in the center while
    keeping the potential in the upper-right area.
</reasoning>

<answer>
\boxed{Next move color: Black}
\boxed{Next move position: K3}
\boxed{Win rate for the next move: 44.5\%}
</answer>
```

Category	Models
KataGo-HumanSL	KataGo-HumanSL-{18K, 17K, 16K, 15K, 14K, 13K, 12K, 11K, 10K} KataGo-HumanSL-{9K, 8K, 7K, 6K, 5K, 4K, 3K, 2K, 1K} KataGo-HumanSL-{1D, 2D, 3D, 4D, 5D, 6D, 7D, 8D, 9D}
KataGo	kata1-b6c96-s175395328-d26788732
KataGo	kata1-b10c128-s1141046784-d204142634
KataGo	kata1-b15c192-s1672170752-d466197061
KataGo	kata1-b20c256x2-s5303129600-d1228401921
KataGo	kata1-b18c384nbt-s6827143680-d3483991074

Table 3: Models used in KataGo-Eval-Levelwise.

C Analysis of KataGo-Eval-Levelwise

Next, as this is the most important experimental benchmark in this work, we briefly discuss several properties of KataGo-Eval-Levelwise.

C.1 Models Used

We list the 32 models used to construct the benchmark, comprising 27 KataGo-HumanSL models and five specialized KataGo models. Here, KataGo-HumanSL denotes the KataGo series of models trained to imitate human players at different skill levels. In the model names, K denotes kyu and D denotes dan. The skill level increases from 18K to 1K, then from 1K to 1D, and further from 1D to 9D. The KataGo models are specialized Go models with different sizes and capability levels, all trained using the algorithm proposed by Wu (2020).

C.2 Level-wise Results

To construct the level-wise split, we measure the difficulty of each board position by the number of reference models whose predicted move belongs to $\mathcal{A}_8(s)$, the reference KataGo model's top eight candidates. Positions solved by fewer models are considered harder, while positions solved by more models are considered easier. We then sort all positions by this difficulty score and uniformly divide them into nine levels, denoted as L0 to L8. Therefore, L0 contains the hardest positions, and L8 contains the easiest positions.

Figure 11 shows the level-wise hit rates of different models on KataGo-Eval-Levelwise. Overall, the benchmark exhibits a clear and stable ranking among models. For KataGo-HumanSL models, hit rates generally increase from kyu levels to higher dan levels. The improvement is especially clear on the middle levels, where low-rank HumanSL models still have limited accuracy, while high-dan HumanSL models achieve much stronger performance.

The specialized KataGo models further improve over HumanSL models. As the model size increases from b6c96 to b18c384nbt, the overall hit rate rises substantially, and the strongest model achieves high accuracy across almost all levels. Meanwhile, the easiest levels are nearly saturated for most models, while the hardest levels remain discriminative. This suggests that KataGo-Eval-Levelwise can provide a meaningful fine-grained evaluation of Go-playing ability across both human-level imitation models and specialized Go models. These results characterize agreement with the reference engine across difficulty levels. Full-game results provide a separate measure of playing performance.

Additional move-quality metrics. Table 4 complements top-eight hit rates with top-one agreement and mean score loss. The reasoning system improves both metrics over Single LLM, indicating that its gains extend to exact agreement and the reference engine's score estimates. KataGo score losses are evaluated on the available reference-analysis subsets, whose sizes are reported alongside the results.

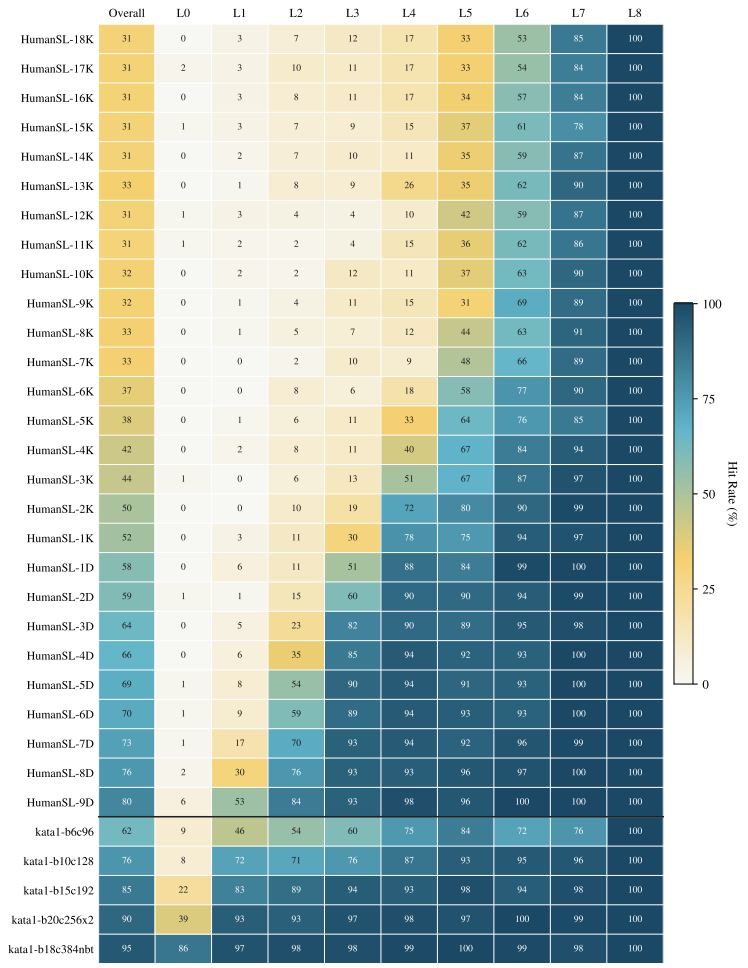


Figure 11: Level-wise hit rates of different KataGo-HumanSL and KataGo models on KataGo-Eval-Levelwise. Darker colors indicate higher hit rates.

Model	Top-1 hit rate (%) ↑	Mean score loss (points) ↓	Score-loss positions
KataGo-96c	19.90	1.42	638
KataGo-128c	29.97	1.40	714
KataGo-192c	40.31	1.10	752
KataGo-256c	45.74	1.02	765
KataGo-384c	54.39	0.37	772
Single LLM	45.35	0.97	774
Reasoning System	51.29	0.82	774

Table 4: Top-1 agreement and move quality on 774 benchmark positions. Top-1 compares the selected move with the reference engine's first-ranked move. Score loss is the reference score lead for the best move minus that for the selected move, from the current player's perspective, averaged across positions. Both moves are evaluated by the same reference engine. Top-1 uses all 774 positions. Score-loss means use the positions counted in the last column, where reference evaluations of both moves are available.

C.3 Full Game Tournament

Tournament protocol The tournament includes 27 KataGo-HumanSL models, five specialized KataGo models, Single LLM, and Reasoning System. The tournament contains 300 games for each

Player ID	System wins–losses
2–7 dan	
Player1	9–0
Player2	9–0
Player3	9–0
Player4	8–1
Player5	8–1
Player6	8–1
Player7	7–1
8–9 dan	
Player8	8–1
Player9	7–3
Total	73–8

Table 5: Reasoning System match results against anonymized professional Go players.

of the 561 model pairs, giving 168,300 games in total. Colors are assigned independently before each game through random nigiri. All games follow Chinese rules with 7.5 komi and start from an empty board. All KataGo and KataGo-HumanSL models use a search budget of 4,096 visits per move. LLM moves are generated with temperature 0.9 and a maximum generation length of 8K tokens. An illegal move results in an immediate loss.

Direct match results Table 6 reports the complete matchup results. Reasoning System wins a majority of games against Single LLM and KataGo-128c and 192c, while its results against KataGo-256c and 384c are near even. The 95% Wilson intervals for the latter two matchups include 50%, consistent with closely matched performance under this protocol. All intervals are pointwise and treat individual games as independent observations.

Professional player matches We additionally evaluate Reasoning System over 81 games against nine professional players ranked 2 dan to 9 dan. Human players are not permitted to use AI assistance. All moves played by the system in these 81 games are legal. Games are played without handicap under Chinese rules with 7.5 komi, with colors assigned by random nigiri. Each side has 30 minutes of main time followed by three 20-second byo-yomi periods. Pooling games across these opponents, the system wins 73 of 81 games, giving a win rate of 90.1% and a pointwise 95% Wilson confidence interval of $[81.7\%, 94.9\%]$ under the independent-game assumption. Seven opponents are ranked 2 dan to 7 dan, and two are ranked 8 dan to 9 dan. The system wins 58 of 62 games (93.5%) against the 2–7 dan group and 15 of 19 games (78.9%) against the 8–9 dan group. Table 5 reports the individual match results.

Elo estimation We estimate relative playing strength using a Bradley Terry model fitted to the observed win and loss counts. Let r_i be the Elo rating of model i. Its probability of winning against model j is

$$p_{ij} = \frac{1}{1 + 10^{(r_j - r_i)/400}}. \tag{3}$$

For each pair $i < j$, let w_{ij} and w_{ji} denote the numbers of wins by the two models. We maximize

$$\mathcal{L}(\mathbf{r}) = \sum_{i<j} \left[w_{ij} \log p_{ij} + w_{ji} \log(1 - p_{ij}) \right], \qquad \frac{1}{34} \sum_{i=1}^{34} r_i = 1500. \tag{4}$$

The mean rating fixes the arbitrary additive offset. Ratings are fitted jointly and do not depend on game order. Benchmark scores are not used in the fit, and these Elo values are relative to the evaluated model pool rather than an external human rating scale.

We compute 95% percentile confidence intervals from 1,000 bootstrap replicates. Within each matchup, we resample its observed game outcomes with replacement, preserving its sample size of 300 games, and refit all ratings, fixing the mean to 1500 in every replicate. The intervals quantify uncertainty from the observed game outcomes under the fixed evaluation settings. They do not measure variation across training runs or alternative inference budgets.

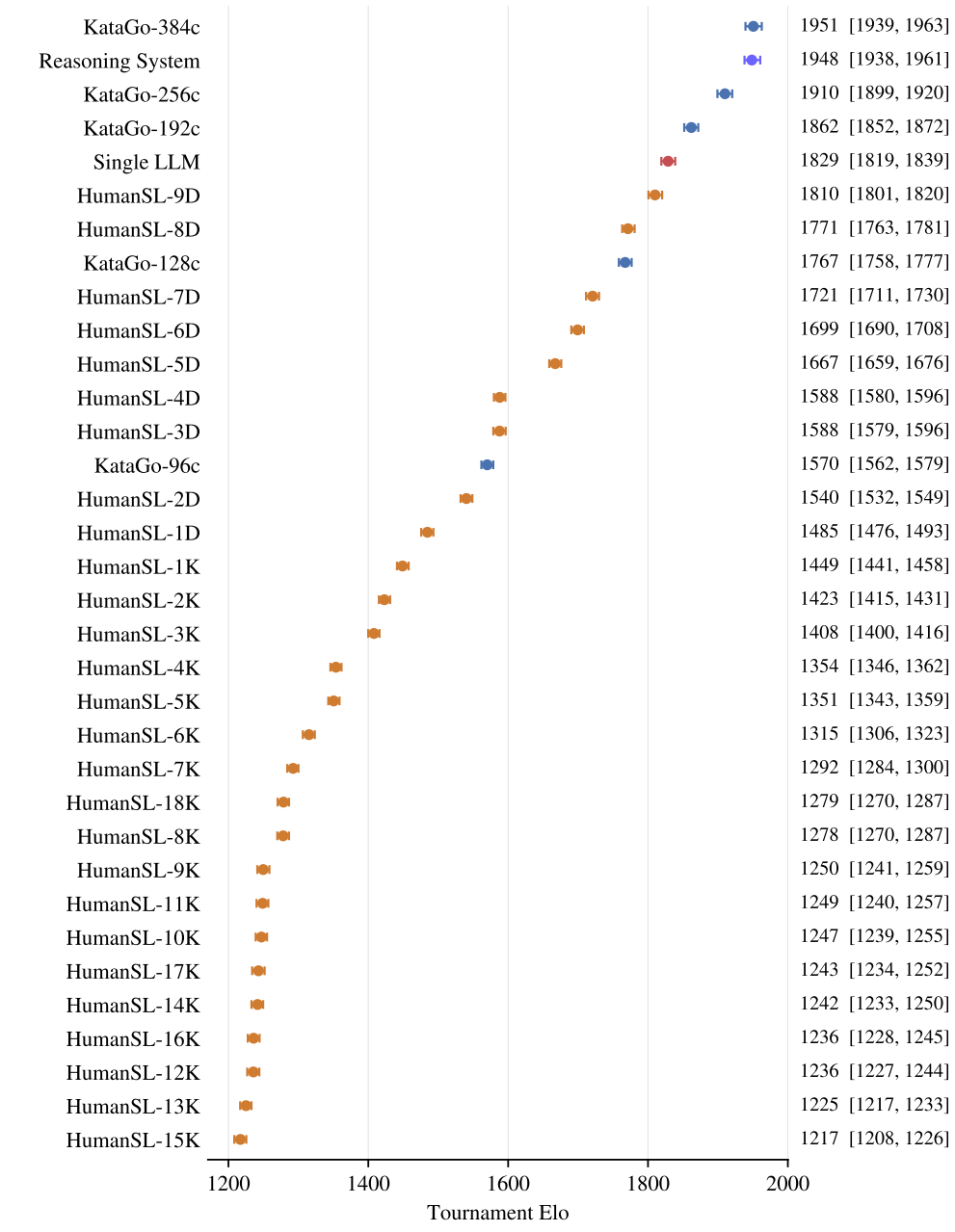


Figure 12: Tournament Elo ratings fitted from all 168,300 games. Points show maximum likelihood estimates and bars show pointwise 95% bootstrap confidence intervals. The values on the right give the estimates and intervals. The mean rating is fixed to 1500.

Figure 12 reports the ratings and their intervals. Reasoning System has an estimated Elo of 1948.45 with a 95% interval of $[1938.04, 1960.51]$, compared with 1828.75 and $[1819.13, 1838.89]$ for Single LLM. KataGo-384c has the highest fitted rating in the evaluated pool. Figure 13 provides the full pairwise win rate matrix. The ratings summarize performance across the opponent pool, whereas direct win rates describe individual matchups. A pooled Elo difference need not match the observed win rate of a particular pair, and changing the opponent pool can change the fitted ratings.

	Single LLM		Reasoning System	
Opponent	Wins	Losses	Wins	Losses
KataGo-384c	112	188	146	154
KataGo-256c	93	207	148	152
KataGo-192c	151	149	198	102
HumanSL-9D	153	147	200	100
HumanSL-8D	186	114	210	90
KataGo-128c	180	120	218	82
HumanSL-7D	215	85	237	63
HumanSL-6D	206	94	244	56
HumanSL-5D	198	102	232	68
HumanSL-4D	239	61	271	29
HumanSL-3D	225	75	265	35
KataGo-96c	231	69	278	22
HumanSL-2D	236	64	278	22
HumanSL-1D	264	36	283	17
HumanSL-1K	275	25	286	14
HumanSL-2K	274	26	290	10
HumanSL-3K	278	22	294	6
HumanSL-4K	293	7	290	10
HumanSL-5K	281	19	292	8
HumanSL-6K	284	16	288	12
HumanSL-7K	289	11	297	3
HumanSL-18K	292	8	294	6
HumanSL-8K	287	13	296	4
HumanSL-9K	288	12	293	7
HumanSL-11K	292	8	296	4
HumanSL-10K	290	10	294	6
HumanSL-17K	293	7	299	1
HumanSL-14K	295	5	298	2
HumanSL-16K	291	9	297	3
HumanSL-12K	286	14	298	2
HumanSL-13K	290	10	297	3
HumanSL-15K	287	13	300	0

Table 6: Complete results of Single LLM and Reasoning System against the 32 reference models. Each matchup contains 300 games. In their mutual matchup, Reasoning System records 197 wins and 103 losses.

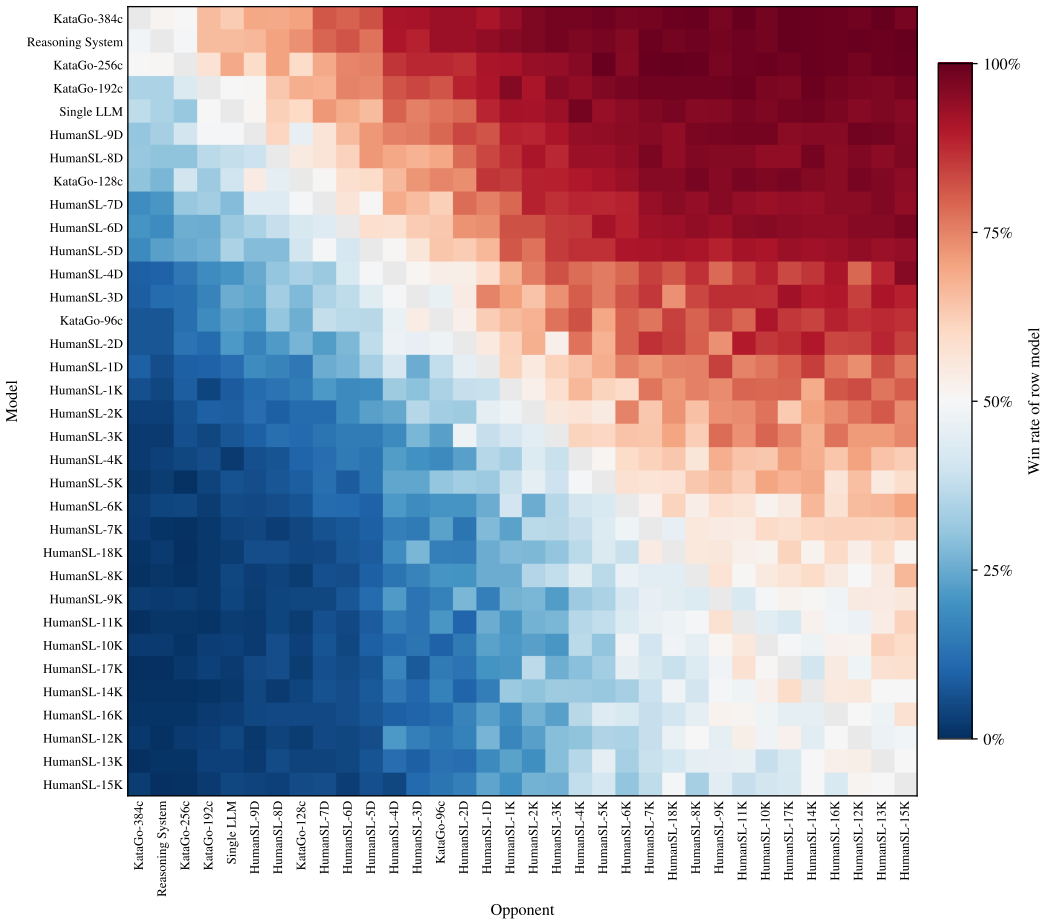


Figure 13: Win rates for all tournament matchups. Each cell gives the win rate of the row model against the column model, using all 300 games in that matchup. Models are ordered by fitted Elo. Diagonal cells are not evaluated.



Figure 14: Overview of the Go Annotation Platform.

D Details of Annotation

We independently design a Go annotation platform. Specifically, the interface of the annotation platform is shown in Figure 14. The left side shows the current game board, while the upper-right part contains an editable text box. Initially, this text box displays the original response generated by the model. The Go players are asked to revise the response in this box, replacing imprecise expressions with professional terminology and accurate board-state understanding. The lower-right part records statistics on the number and types of revisions made by the annotator. After finishing the annotation of one sample, the annotator clicks the submit button in the lower-right corner to proceed to the next sample.

Human evaluation protocol For the evaluation in Table 2, each response is initially assessed by three or four Go players. Players judge the correctness and relevance of every Go term appearing in the response, including terms outside the approximately 300-term vocabulary used for training data construction. We aggregate judgments by majority vote for each criterion and count a term as valid only if both criteria pass. Major-error judgments are also aggregated by majority vote. In the event of a tie, judgments from players of higher rank take precedence. If the tie remains unresolved, an additional Go player evaluates the item, and the majority decision is used. A major error is a clearly incorrect statement involving Go terminology or analysis of the board position. The major-error rate is the proportion of responses containing at least one such error.

Analysis utility evaluation The evaluation in Figure 5 contains 2,400 output ratings and 480 five-way first-choice selections from 12 professional Go players evaluating the same 40 positions. Each of the five methods receives 480 ratings. The correctness rating jointly assesses the proposed variations and the accompanying Go-specific descriptions, including terminology and positional analysis. A response is judged correct only when both components are reasonable for the given position. These judgments are expressed on the 1–5 rating scale. Helpfulness rates the response's usefulness to the player. Ours denotes the reasoning output of the final Single LLM after Term-DPO, which is also evaluated in Table 2. For the GPT-5.6 + KataGo baseline, GPT-5.6 receives structured analysis results from KataGo and is asked to polish the wording and explain the analysis. We compute 95% percentile confidence intervals from 10,000 bootstrap replicates. Each replicate independently resamples positions and players with replacement, retaining their crossed observations and using the same samples for all systems to preserve paired comparisons.

E Preliminaries and Reward Design

We next briefly introduce the specific design of the algorithms used in our experiments, and provide a quantitative description of the reward design in the RL stage.

E.1 PRELIMINARIES: DPO AND GRPO

Direct Preference Optimization. Direct Preference Optimization (DPO) is a preference optimization algorithm that directly trains the policy model on preference pairs without explicitly fitting a separate reward model. Given a prompt x, a preferred response y_w, a dispreferred response y_l, a policy model π_θ, and a reference model π_{ref}, DPO optimizes the following objective:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta) = -\mathbb{E}_{(x,y_w,y_l)}\left[\log\sigma\left(\beta_{\text{DPO}}\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta_{\text{DPO}}\log\frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right], \quad (5)$$

where β_{DPO} controls the strength of the KL constraint to the reference policy. Intuitively, DPO increases the relative probability of the preferred response while decreasing that of the dispreferred response, with the reference model serving as a regularization anchor. In our setting, DPO is used as the basic preference-learning objective for improving professional Go-related expressions.

Group Relative Policy Optimization. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is a reinforcement learning algorithm proposed as a memory-efficient variant of PPO. Unlike PPO, which usually trains an additional value model to estimate the baseline, GRPO samples a group of responses for the same prompt and estimates the relative advantage within the group. Specifically, for each prompt q, GRPO samples a group of responses $\{o_i\}_{i=1}^G$, obtains their rewards $\{r_i\}_{i=1}^G$, and normalizes the rewards within the group:

$$A_i = \frac{r_i - \text{mean}(\{r_j\}_{j=1}^G)}{\text{std}(\{r_j\}_{j=1}^G)}. \quad (6)$$

The resulting advantage A_i is then used to update the policy with a PPO-style clipped objective and a KL regularization term. Since the baseline is computed from group-level rewards, GRPO avoids training a separate critic model and reduces memory cost, making it suitable for RLVR-style training on reasoning tasks. In our experiments, we use GRPO in the RL stage to improve the model's Go-playing ability.

E.2 REWARD DESIGN

For the policy model and sampler, the RL reward takes the form

$$R = \mathbb{I}[\text{format correct}] \cdot \big(R_{\text{rank}} - \alpha R_{\text{opt}}\big),$$

where $\mathbb{I}[\text{format correct}]$ is an indicator function that equals 1 only when the model response follows the required output format. This term prevents the model from receiving positive reward when the generated answer cannot be parsed correctly.

Sampler reward For a board state s, let $P(s)$ be the set of distinct moves predicted by the sampler, and let K be the required number of candidates. The reference set $\mathcal{A}_8(s)$ contains the top eight candidate moves from the reference KataGo model. We define the overlap proportion and the sampler reward as

$$c(s) = \frac{|P(s) \cap \mathcal{A}_8(s)|}{K}, \qquad R_{\text{rank}}(s, P) = f(c(s)), \quad (7)$$

where

$$f(c) = \begin{cases} 0, & c = 0, \\ 0.3, & 0 < c < 0.5, \\ 0.6, & 0.5 \le c < 1, \\ 1.0, & c = 1. \end{cases} \quad (8)$$

We set $\alpha = 0$ for the sampler. Its format check also requires exactly K candidate entries. Duplicate moves contribute at most one hit, while the denominator remains K. With $K = 2$, zero, one, and two distinct matching moves receive rewards of 0, 0.6, and 1.0, respectively. Invalid output formats or incorrect candidate counts receive zero reward.

Policy reward For the policy model, the term R_{rank} is a rank-based reward for the selected move. We use the strongest reference Go model to obtain a ranked list of candidate moves for each board state. Formally, let $\text{rank}_{\text{ref}}(a \mid s)$ denote the rank of the model-selected move a in the candidate list produced by the reference model under board state s. We define R_{rank} as the following piecewise function:

$$R_{\text{rank}}(s,a) = \begin{cases} 1.0, & \text{rank}_{\text{ref}}(a \mid s) = 1, \\ 0.8, & 2 \le \text{rank}_{\text{ref}}(a \mid s) \le 3, \\ 0.6, & 4 \le \text{rank}_{\text{ref}}(a \mid s) \le 7, \\ 0.4, & 8 \le \text{rank}_{\text{ref}}(a \mid s) \le 10, \\ 0, & \text{otherwise.} \end{cases}$$

Moves ranked higher by the reference model receive larger rewards. This design provides a dense learning signal while still encouraging the model to match high-quality Go moves.

The second term R_{opt} penalizes the gap between the selected move and the best move:

$$R_{\text{opt}} = \frac{\beta(w^* - w)}{1 + \beta(w^* - w)},$$

where w^* denotes the win rate of the best move estimated by the reference model, and w denotes the win rate of the move selected by the policy model. We represent both win rates as floating-point values in $[0, 1]$ and set $\alpha = 0.5$ and $\beta = 0.2$ for the policy model. The penalty increases with the win-rate gap, and α controls its weight relative to the rank-based reward.

Text RM reward For a board state s and K distinct candidate moves $a_1, \ldots, a_K$, let v_i and $\hat{v}_i$ denote the reference KataGo win rate and the Text RM prediction for a_i, respectively. All win rates use the perspective of the player choosing the move. We define the set of comparable candidate pairs as

$$\mathcal{E}(s) = \{(i,j) \mid 1 \le i < j \le K,\ v_i \ne v_j\}. \tag{9}$$

The reference and predicted comparison directions are

$$y^*_{ij} = \text{sign}(v_i - v_j), \qquad \hat{y}_{ij} = \text{sign}(\hat{v}_i - \hat{v}_j). \tag{10}$$

Text RM receives the proportion of correctly ordered pairs, subject to the output format check,

$$R_{\text{RM}}(s) = \mathbb{I}[\text{format correct}] \cdot \frac{1}{|\mathcal{E}(s)|} \sum_{(i,j)\in\mathcal{E}(s)} \mathbb{I}[\hat{y}_{ij} = y^*_{ij}]. \tag{11}$$

Each candidate pair is counted once. Reference ties are excluded, while predicted ties on comparable pairs count as incorrect. The format check requires a valid win-rate prediction for every candidate. Positions with $\mathcal{E}(s) = \emptyset$ are excluded from RL training. For two candidates with a strict reference ordering, the reward is 1 for a valid response with the correct ordering and 0 otherwise.

F Computational Resources

SFT and RL each use 32 H800 GPUs, with XTuner and VeRL as the respective training frameworks (Contributors, 2023; Sheng et al., 2024). Inference uses 16 H800 GPUs with SGLang (Zheng et al., 2024).

Term-DPO training configuration Each component starts from its CS+RL checkpoint and uses a frozen copy as the reference model. We perform full-parameter DPO with $\beta_{\text{DPO}} = 0.1$, distinct from the policy reward coefficient $\beta = 0.2$. AdamW uses a learning rate of 10^{-6}, moment coefficients $(0.9, 0.999)$, $\epsilon = 10^{-8}$, and zero weight decay. Full Term-DPO runs for one epoch over its approximately 500,000 preference pairs with a global batch of 64 preference pairs, cosine learning-rate decay, and 3% warmup. We use BF16, a gradient norm limit of 1.0, and seed 42. Dropout and label smoothing are disabled. Each concatenated prompt and response is limited to 16,384 tokens. Response log probabilities are summed over tokens, excluding prompt tokens. Both ablations use the same CS+RL initialization, training hyperparameters, and optimizer update count as full Term-DPO. The original-only variant constructs approximately 80,000 distinct preference pairs from the same 5,000 annotated examples. We resample these pairs to match the total number of training pairs processed by full Term-DPO.

Method	Models	Hit rate (%)	Tokens/position	Time/position (s)
Single LLM	1×8B	82.8	1,301	1.4
Single LLM (Oracle@2)	1×8B	83.0	2,508	2.7
Reasoning System	2×8B	92.1	2,093	2.3

Table 7: Hit rates and mean inference costs on KataGo-Eval-Levelwise. Single LLM (Oracle@2) uses two generations at temperature 1.2 and counts a hit if either move satisfies the top-eight and win-rate-loss conditions in Section 2.2. Oracle@2 in Table 1 instead evaluates two candidates within one response. Other rows score the final selected move. Costs include both generations for Oracle@2 and both sampler and Text RM for Reasoning System. Time spans receipt of a position to completion of both predictions or final move selection, respectively.

Scaling experiment repetitions. Figure 3 reports three independent training runs with seeds 3407, 5813, and 9209, evaluated on the same 774 positions. At each setting, we average the three run-level hit rates and construct a pointwise 95% Student's t interval with two degrees of freedom, using the sample standard deviation divided by $\sqrt{3}$ as the standard error. The intervals summarize variation across training runs on this fixed evaluation set.

G Broader Impacts

Training game records come from publicly available Internet sources and contain moves and game-outcome annotations. Human annotations concern Go analyses and terminology and contain no private information.