Uncertainty-Guided Exploration for Efficient AlphaZero Training

Scott Cheng[1] **Meng-Yu Tsai**[2] **Ding-Yong Hong**[3] **Mahmut Taylan Kandemir**[1]

[1]The Pennsylvania State University, USA [2]Independent
[3]Institute of Information Science, Academia Sinica, Taiwan

[1]{ypc5394,mtk2}@psu.edu [2]adamatuno@gmail.com [3]dyhong@iis.sinica.edu.tw

Abstract

AlphaZero has achieved remarkable success in complex decision-making problems through self-play and neural network training. However, its self-play process remains inefficient due to limited exploration of high-uncertainty positions, the overlooked runner-up decisions in Monte Carlo Tree Search (MCTS), and high variance in value labels. To address these challenges, we propose and evaluate uncertainty-guided exploration by branching from high-uncertainty positions using our proposed Label Change Rate (LCR) metric, which is further refined by a Bayesian inference framework. Our proposed approach leverages runner-up MCTS decisions to create multiple variations, and ensembles value labels across these variations to reduce variance. We investigate three key design parameters for our branching strategy: where to branch, how many variations to branch, and which move to play in the new branch. Our empirical findings indicate that branching with 10 variations per game provides the best performance-exploration balance. Overall, our end-to-end results show an improved sample efficiency over the baseline by 58.5% on 9x9 Go in the early stage of training and by 47.3% on 19x19 Go in the late stage of training.

1 Introduction

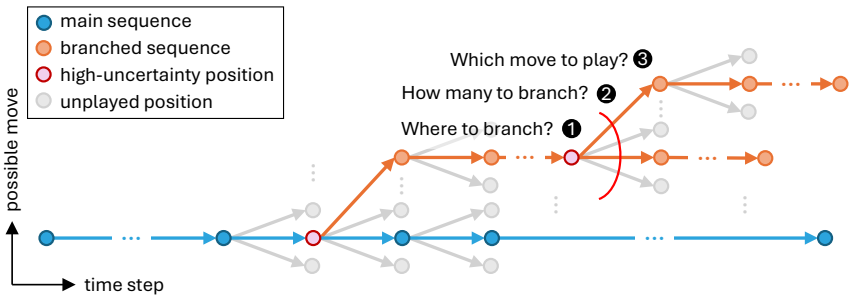


Figure 1: Our proposed method enhances exploration in self-play by branching at high-uncertainty positions to explore additional runner-up MCTS decisions, and reduces the variance of the value label by ensembling multiple game results, thus improving training efficiency.

AlphaZero [1] has demonstrated remarkable capabilities by combining Monte Carlo Tree Search (MCTS) [2, 3] and deep neural networks, achieving superhuman performance in games such as Go,

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Chess, and Shogi. It learns through self-play, where a neural network guides MCTS to generate games, and the game results are used to train the network. This process is repeated in an alternating fashion, gradually improving the playing strength. As the size and capability of the neural network increase, the cost of evaluating it during MCTS increases substantially, making self-play the dominant component of overall training time.

Despite its success, AlphaZero's self-play process remains inefficient for several reasons. First, only 40% of the positions change decisions after the tree search [4, 5], so only a few positions benefit from the search for improvement. Those high-uncertainty positions are unfortunately not reused for exploration. Second, only one MCTS decision is used per move during self-play, while other non-best decisions, especially the runner-up decisions, are discarded without exploration. Those runner-up moves can potentially be stronger than the best decision, but due to an insufficient number of simulations, they fail to overtake the current best one, even if a better move was found during the latter phase of tree search. Third, the value label in self-play training is provided by a single game result and thus exhibits high variance, especially during the opening phase of a game. Previous works [6, 7] on reusing positions to improve AlphaZero training efficiency focus mainly on early game exploration, thus failing to capture high-uncertainty positions, particularly in the mid-to-late game. Furthermore, most of the work [8–11] on efficient training of AlphaZero and its variants focuses on improving the learning target for neural network training. In contrast, this work focuses on exploring high-uncertainty positions during self-play, and thus complements these prior approaches.

To this end, we propose **uncertainty-guided self-play**, which revisits high-uncertainty but under-explored positions, as illustrated in Figure 1. This branching process is analogous to how humans learn after game plays, where people identify the last major mistake, typically not in the opening, and revisit that position to explore highly considered but initially overlooked decisions to change the game result. In particular, we aim to address the following questions in the context of branching: *i) Where to branch?, ii) How many variations*[1] *to branch?, and iii) Which move to play in each new variation?* For the first question, we branch from high-uncertainty positions by our proposed *Label Change Rate*, which quantifies uncertainty by estimating how likely the game result would differ when repeatedly replaying from a given position. We further refine this estimate using a **Bayesian inference framework** to robustly integrate both prior knowledge from search values and empirical game results. For the second question, we control the number of branches via a variation budget, balancing the degree of exploration within a single game and across multiple games. Finally, for the third question, we reuse the runner-up decisions from the original search result to play the new variation without performing additional simulations or relying on other larger models, since we observe that, when a deeper search changes the best move, the new best decision has a high chance coming from the runner-up decisions in the initial search. Moreover, since obtaining an MCTS planning result typically requires hundreds of neural network inferences, we can reuse the runner-up decisions during self-play, unlike the original method, which discards all decisions but the best result.

In summary, this paper makes the following main **contributions**:

- We introduce an uncertainty-guided exploration for AlphaZero, which enhances exploration by revisiting high-uncertainty positions and reduces the variance of the value label by ensembling multiple game results. Furthermore, we conduct a detailed analysis of the three design dimensions for uncertainty branching and experimentally show their contributions to training efficiency.
- We propose the **Label Change Rate** (LCR) metric based on our analysis on the Bernoulli distribution, along with a Bayesian refinement scheme, to robustly identify high-uncertainty positions. We experimentally show that the proposed uncertainty estimator closely matches the empirical results, with an average RMSE (root mean square error) of 0.02. This metric enables us to discover high-uncertainty positions at a later game stage via an analytical form.
- Our evaluation indicates that using 10 variations per game achieves the best balance for exploration within and between games. Moreover, branching at positions selected uniformly at random leads to a significant decrease in strength, with only a 25.5% win rate against our proposed method, highlighting the critical role of branch point selection and the insufficiency of relying solely on the value ensemble. Finally, our end-to-end results show that our method improves sample efficiency over the baseline in *both* the early and late stages of training, achieving an improvement of up to 58.5% in 9x9 Go and 47.3% in 19x19 Go.

[1]We use "branches" and "variations" interchangeably; "variation" follows the terminology used in GTP [12].

2 Background

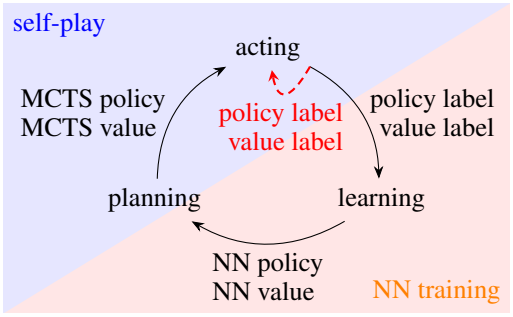


Figure 2: Interaction of policy and value among the stages of acting, planning, and learning in AlphaZero. Our method further employs policy and value labels early during acting, as indicated by the red arrow.

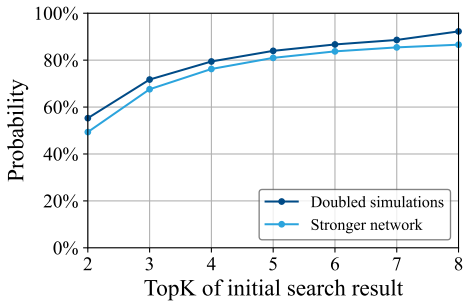


Figure 3: Probability that when doubling simulations or using a stronger network, the new best move appears within the initial top k search results, conditioned on cases where the original top-1 move changes.

2.1 AlphaZero

AlphaZero combines MCTS with a deep neural network (NN) to address complex decision-making problems. As shown in Figure 2, its training process alternates between two phases: (1) self-play and (2) neural network training. In the self-play phase, AlphaZero generates games by *acting*, with each move determined through MCTS *planning*. In the training phase, the neural network is updated by *learning* from the collected self-play games. Specifically, the neural network learning target contains (i) a **policy label**, corresponding to the root visit count distribution of MCTS, and (ii) a **value label**, representing the final game result. Moreover, during planning, the MCTS search tree is guided by the neural network policy and value to obtain the search result, typically requiring hundreds to thousands of neural network predictions to obtain a search result. Once a decision is made by MCTS planning, the resulting MCTS policy and value are used to select the next action or decide to resign. Through the cycle of acting, planning, and learning, AlphaZero continuously refines its decision-making capabilities. Furthermore, unlike the original approach, our method augments the self-play process by incorporating not only the MCTS planning results but also the neural network learning targets (the red arrow in Figure 2) to guide acting. This additional label information enables us to explore more high-uncertainty positions during acting, as we will discuss later in Section 4.2.

2.2 Motivation

In this section, we study the relationship between stronger decisions and their rank in the search result. Previous studies [4, 5] have shown that, for more than 60% of positions, the best move remains *unchanged* after MCTS. Thus, we focus on the remaining 40% of important positions where the best search result *changes*. Specifically, we examine whether the new best decision emerges from novel, out-of-distribution (OOD) decisions or from a runner-up decision already present in the initial search.

To analyze this, we generate improved decisions by increasing simulation counts or using a stronger network, and then compare the new best move to the ranking in the initial search results. Figure 3 shows that when search results change due to an increase in simulation count or a stronger network, the improved decision falls within the top 4 of the initial search results in more than 80% of the cases. In contrast, OOD decisions account for only 10%. These observations motivate us to *reuse* the initial search results when exploring potentially stronger decisions. Since most improved decisions come from runners-up of the initial search, this approach avoids the cost of additional simulations or larger networks, while still providing a high chance of improving the decision.

3 Related Work

Most prior works on improving AlphaZero's training efficiency and its variant, MuZero [13], focus on refining the training objective, such as introducing path consistency [9, 14], using contrastive

learning [15], and modifying the value target [16, 17]. Additional improvements involve enhancing prioritized experience replay [11, 18] and adopting an alternative training optimizer [8]. In contrast, our proposed method focuses on enhancing exploration *during self-play*, thus complementing these existing approaches toward efficient AlphaZero training. Furthermore, previous studies [19, 20] on uncertainty in deep reinforcement learning typically rely on neural networks, such as predicting uncertainty using neural networks [5, 21–24], planning to explore with world models [25–27], or representing uncertainty as the discrepancy or variance between neural networks [28–30]. In contrast, we propose an *analytical uncertainty metric* that can be directly estimated from the results of the MCTS search. We also discuss the distribution assumptions for our work compared to the previous works in Section 6.

On the other hand, previous work has also explored the option of solving hard exploration problems by revisiting unfamiliar states [31], or replaying trajectories starting from states of interest [6, 32] by sampling from an archive of positions. Similarly, KataGo [7] includes an option to fork from a random position to play, especially in the opening phase of a game. However, these approaches primarily focus on early game exploration, as opening positions are often, naturally, of high uncertainty. As we show later in Section 5.2, these works fail to effectively target high-uncertainty positions in the mid and late stages of a game. In contrast, our proposed uncertainty metric can identify such challenging positions across the entire game.

4 Methodology

We first establish the foundations of uncertainty estimation in Section 4.1. Then, we introduce uncertainty-guided branching for self-play to enhance training efficiency in Section 4.2.

4.1 Uncertainty Estimation

In this subsection, we quantify the reduction in training variance, introduce our uncertainty metric to guide exploration, and present a Bayesian framework for refining uncertainty estimation with empirical observations. Detailed proofs and derivations can be found in the appendix A.2.

Variance Reduction. First, to reduce variance in the training target, we ensemble multiple game results. To formalize this, let w be the win rate at a given position in a game, and let $Z \sim \text{Bernoulli}(w)$ be the random variable representing the result of a single game (winning or not). Consider n independent and identically distributed (i.i.d.) samples $Z_1, \ldots, Z_n \sim Z$. Then, the variance of the sample mean $\bar{Z} = \sum_{i=1}^{n} Z_i/n$ is given by:

$$\text{Var}[\bar{Z}] = w(1-w)/n. \tag{1}$$

In AlphaZero, the average game result $\bar{Z}$ over n games played from a given position serves as the value label during training. Hence, by increasing the degree of exploration from a given position, we can reduce the variance of the value label with more game results.

Uncertainty Estimator. Next, to quantify the uncertainty of the game results from a given position s, we introduce **the Label Change Rate (LCR)**, denoted by $\theta = \text{LCR}(s)$, defined as the probability that two independently sampled game results, Z_i and Z_j for $i \neq j$, are different as follows:

$$\text{LCR}(s) := \Pr(Z_i \neq Z_j) = 2w(1-w). \tag{2}$$

This metric also matches our intuition, since the uncertainty is highest when the win rate is 50% and approaches zero when the result is nearly determined; that is, when the win rate is close to 0 or 1. In practice, this metric enables an online uncertainty estimation during self-play by using the search value v to estimate θ.[2] Moreover, we will show later in Section 5.2 that this analytical approach to uncertainty estimation aligns closely with the empirical results. In general, collecting more states with greater epistemic uncertainty can increase the degree of exploration and lead to more efficient training [19, 30, 33].

Bayesian Inference. When multiple game results are available for a given position, Bayesian inference can provide a more robust estimate of the LCR since simply employing the search value

[2]If the search value $v \in [0, 1]$, we obtain an estimate $\hat{\theta} = 2v(1-v)$; otherwise, we rescale v to interpret it as a win rate before computing $\hat{\theta}$.

v to estimate w in Equation 2 ignores the uncertainty about v itself. To address this, we model the LCR by treating the underlying label change probability θ as a random variable. Thus, each observed label change is represented as $X_i \sim \text{Bernoulli}(\theta)$, where $X_i = 1$ indicates that two independently sampled outcomes from the same position differ, and $X_i = 0$ otherwise. Given a sequence of i.i.d. observations $D = \{X_1, X_2, \ldots, X_m\}$, we perform Bayesian inference over θ, and define the LCR as the posterior expectation $\mathbb{E}[\theta \mid D]$. In practice, we can construct the observations as $X_i = \mathbf{1}_{Z_{2i} \neq Z_{2i-1}}$ using paired game outcomes for $i \in [1, m]$ and $m = \lfloor n/2 \rfloor$. Alternative empirical choices are discussed in the appendix A.2.4. Let $s = \sum_{i=1}^{m} X_i$ denote the number of observed label changes. Since the conjugate prior for the Bernoulli likelihood is the Beta distribution, we assume the prior $\theta \sim \text{Beta}(\alpha, \beta)$, and thus the posterior distribution becomes $\theta \sim \text{Beta}(\alpha + s, \beta + m - s)$. Consequently, the corresponding Bayesian estimate of the posterior LCR is as follows:

$$\mathbb{E}[\theta \mid D] = \frac{\alpha + s}{\alpha + \beta + m}, \tag{3}$$

where the prior LCR $= \mathbb{E}[\theta] = \frac{\alpha}{\alpha+\beta}$.

As an illustrative example, suppose that the MCTS search value for a position is $v = 0.6 \in [0, 1]$. Then, the prior LCR is $2 \cdot 0.6 \cdot (1 - 0.6) = 0.48$, according to Equation 2. If we perform four empirical trials ($m = 4$) and observe one label change ($s = 1$), the posterior estimate becomes $\text{LCR} = \frac{\alpha+s}{\alpha+\beta+m} = \frac{4.8+1}{10+4} \approx 0.414$, where $(\alpha, \beta) = (4.8, 5.2)$ and $\frac{\alpha}{\alpha+\beta} = 0.48$. As expected, the posterior LCR decreases slightly, as the observed change rate (1 in 4) is lower than the prior estimate. The scale of α, β can be adjusted to reflect the confidence in the prior probability. In the next section, we will discuss how to integrate this estimation and its update into self-play to guide exploration.

4.2 Uncertainty-Guided Branching

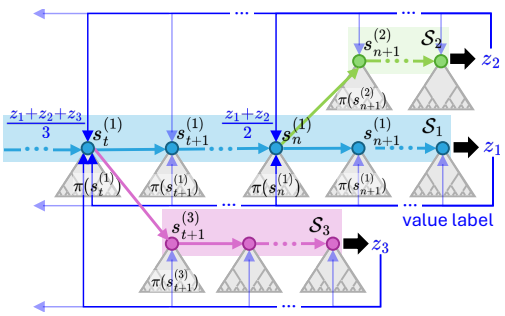


Figure 4: An example of an uncertainty-guided self-play game, where $\mathcal{S}_i$ is the set of positions in variation i, with $\mathcal{S}_1$ as the main sequence. $s_t^{(i)}$ is the position at time t, $\pi(s_t^{(i)})$ is the policy label obtained from MCTS, and z_i is the game result. The value label at each position is computed as the average of all subsequent game results from that position, as indicated by the blue arrows.

Algorithm 1: Uncertainty-Guided Branching

Input: G number of states to collect
Output: $\mathcal{G}$ ={(state, policy label, value label)}

$\mathcal{G} \leftarrow \emptyset$
while $|\mathcal{G}| < G$ **do**
  $(s, a) \leftarrow$ initial state and action
  for $i \leftarrow 1$ **to** V **do**
    $\{(s', \pi(s'), z_i) \mid s' \in \mathcal{S}_i\} \leftarrow$
      play episode from (s, a)
    Compute LCR and sampling weights
     u (defined in Equation 4)
    $s \sim Y(u)$ (defined in Equation 5)
    $a \sim \pi(\cdot \mid s)$
    while (s, a) has been played **do**
      $s \leftarrow$ preceding state of s
      $a \sim \pi(\cdot \mid s)$
  $\mathcal{S} := \bigcup_{i=1}^{V} \mathcal{S}_i$
  $\mathcal{G} \leftarrow \mathcal{G} \cup \left\{ \left(s', \pi(s'), \frac{\sum_{i:\, s' \in \mathcal{S}_i} z_i}{|\{i:\, s' \in \mathcal{S}_i\}|} \right) \mid s' \in \mathcal{S} \right\}$
return $\mathcal{G}$

To improve training efficiency, this subsection integrates the previously discussed strategies for reducing variance in value labels and increasing exploration at high-uncertainty positions into our method. Figure 4 illustrates an example game obtained by Algorithm 1, which contains a main branch $\mathcal{S}_1$ and two variations, $\mathcal{S}_2$ and $\mathcal{S}_3$, where branching occurs at positions $s_n^{(1)}$ and $s_t^{(1)}$, respectively. In particular, for the first branching points, two different moves are sampled from $\pi(s_n^{(1)})$, resulting in two variations starting from positions $s_{n+1}^{(1)}$ and $s_{n+1}^{(2)}$, which in turn lead to game results z_1 and z_2, respectively. With more variations branched, we obtain more game results, allowing the value label of $s_n^{(1)}$ to be computed as the average of z_1 and z_2, rather than only relying on a single result. This reduces label variance and likewise reduces variance for all earlier positions that lead to these variations.

Algorithm 1 shows uncertainty-guided branching during self-play to produce policy and value labels for the training target. It begins by playing the main sequence from an initial state and obtains the training target for the first game variations $\mathcal{S}_1$. We use "play episode from(s, a)" to denote playing a complete game from the state s starting with the action a, and the game playing process remains the same as in the original AlphaZero algorithm. After each completed episode, we calculate the posterior LCR for each position based on Equation 2 and 3. Then, we select a high-uncertainty position based on sampling, and the weight assigned to each position is defined as follows:

$$u\left(s_t^{(i)}\right) = \begin{cases} t, & \text{if LCR}\left(s_t^{(i)}\right) \geq \text{LCR}_0 \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

where LCR_0 represents the uncertainty threshold. This hyperparameter is typically determined based on the initial win rate w_0 so that we can discard positions whose win rates fluctuate around or remain close to the initial win rate. In addition, positions are weighted by their time step because we want to capture the win rate turning point in the later game phase. We then sample the position to branch using a categorical distribution over the set of all positions $\mathcal{S}$:

$$Y \sim \text{Categorical}(p), \quad \text{where} \quad p(s) = \frac{u(s)}{\sum_{s' \in \mathcal{S}} u(s')}. \tag{5}$$

After selecting a position $s \sim Y$ to branch from, we sample an action from the current position and play from there. If the sampled action has already been explored, especially in cases where only one legal move is available, we backtrack to an earlier position and repeat the sampling process. Once an unexplored action is found, we proceed with the play. If no such action can be found, we terminate the loop and return all collected variations early, which is unlikely, and hence we omit this case from Algorithm 1 for simplicity.

Moreover, as discussed in Section 2.2, the runner-up decisions often have a high likelihood of being the best moves. Thus, we select the branching move by resampling from the same policy distribution π, which also aligns with the i.i.d. assumption introduced in the previous section. Overall, we employ the value label in the Bayesian inference and the policy label for action selection, as illustrated earlier in Figure 2. This branching process continues until a predefined variation limit V is reached. In particular, when $V = 1$, our method reduces to the original AlphaZero algorithm without branching. Finally, the self-play process finishes when G states are collected, where G is the same hyperparameter as in the original algorithm.

5 Evaluation

We first provide our experimental setting in Section 5.1. Then, we evaluate and compare empirical and analytical label change rates in Section 5.2. Following that, we evaluate the design space of our proposed approach in Section 5.3. Finally, we conclude by evaluating the end-to-end training efficiency in Section 5.4.

5.1 Experimental Setting

We evaluated our proposed method on 9x9 and 19x19 Go. To demonstrate that our approach improves efficiency for both early and late stages of training, we train the 9x9 Go model from scratch, while for 19x19 Go, we use a model pretrained with 100M samples. Our evaluations employ Elo rating [34] against the state-of-the-art KataGo program [7], and the Elo rating is converted to the win rate to reflect relative playing strength. Detailed hyperparameters for self-play and training are provided in the appendix A.3.

Our experiments are mainly conducted on 2 NVIDIA A100 GPUs with 768 GB system memory. For the empirical analysis in Section 5.2, we collected 20,000 self-play game positions, requiring 10 A100-hours. In Section 5.3, each evaluation on 19x19 Go requires 100 A100-hours, whereas the evaluations on 9x9 Go require 16 A100-hours each. Our end-to-end training experiments in Section 5.4 are conducted on a cluster of 5 nodes, each consisting of 8 NVIDIA V100 GPUs and 768GB system memory. Each training run requires 800 V100-hours with an additional 135 V100-hours for each evaluation. Both our method and the original one require about 23 GB of runtime memory, primarily due to the size of the replay buffer.

5.2 Label Change Rate

In this section, we show that our proposed uncertainty estimator, defined by Equation 2, closely aligns with the uncertainty observed in the empirical label changes. To compute the empirical LCR for a given position, we perform two independent game plays from the same position and compare whether their results differ. Then, the empirical LCR $\bar{X}$ for a position is the ratio of games with changed results, averaged over 20,000 positions. As shown in Figure 5, the empirical mean is closely in agreement with the mean of the analytical estimators, with an RMSE (root mean square error) of 0.023. This suggests that our uncertainty metric provides a reliable analytical estimate of uncertainty without requiring repeated empirical experiments at each position. Moreover, unlike empirical sampling, which yields only binary results per trial, we can now estimate the uncertainty for *each position*, denoted $\hat{\theta}$. Therefore, we adopt our analytical LCR estimator as a practical substitute for empirical uncertainty measurements in the following evaluations. In addition, we perform similar evaluations across different simulations in the appendix A.1.5.

In addition, Figure 5 shows that the LCR remains as high as 50% during the early stages of a game, especially before move 60, leaving limited opportunities to capture additional high-uncertainty positions at this stage. In particular, the original method is equivalent to branching at the first positions of a game. In contrast, we emphasize that uncertain positions continue to appear in a game's mid-to-late stages. The figure shows the distribution of uncertain positions in the self-play games, represented by orange scatter points. Although the average LCR decreases as it gets closer to the end-game, we still observe that several games have an LCR above 40% in the mid-to-late game, even if most positions have nearly determined results and thus have a low LCR due to less uncertainty. Hence, our method focuses on capturing those high-uncertainty positions, which are often buried by a massive amount of low-uncertainty positions. As a result, we use LCR not only as an uncertainty indicator but also as a metric to evaluate how effectively different strategies explore high-uncertainty positions in a game.

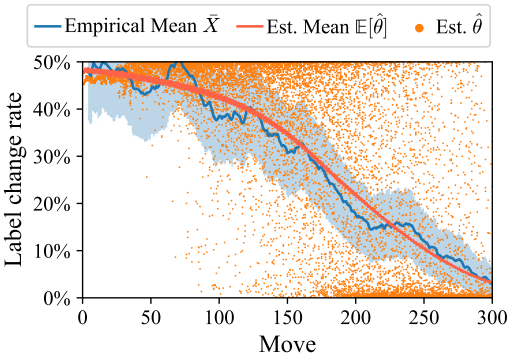


Figure 5: Label change rate (LCR) across moves in 19x19 Go. The empirical mean $\bar{X}$ (blue line) is the average of observed label changes across positions, with a 95% confidence interval shown as the shaded blue region. Analytical estimates $\mathbb{E}[\hat{\theta}]$ (orange line) are averaged from individual LCR estimators $\hat{\theta}$, shown as orange scatter points.

5.3 Design Space Exploration

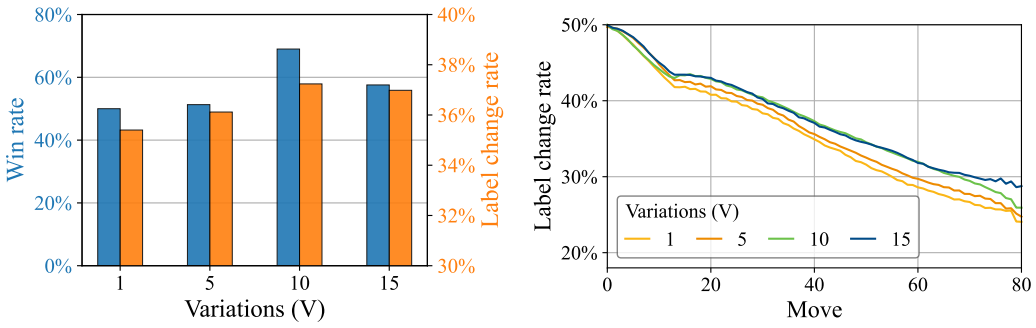


Figure 6: Win rate and label change rate under different variation limits V when training on the same number of samples in 9x9 Go. The $V = 1$ setting serves as the win rate comparison baseline, without uncertainty-guided branching. Reported results have a standard deviation of 1%.

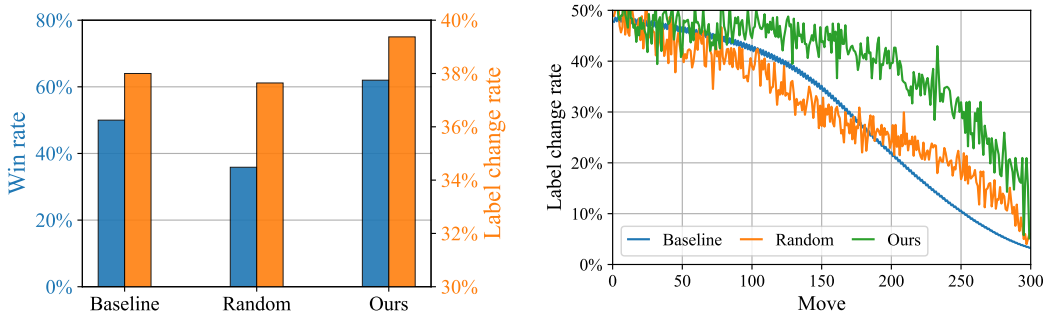


Figure 7: Win rate and label change rate under different branching methods when training under the same number of samples in 19x19 Go. *Baseline* denotes the original algorithm without branching and serves as the win rate comparison baseline; *Random* applies branching at positions selected uniformly at random; *Ours* refers to the proposed uncertainty-guided branching method. Reported results have a standard deviation of 1%.

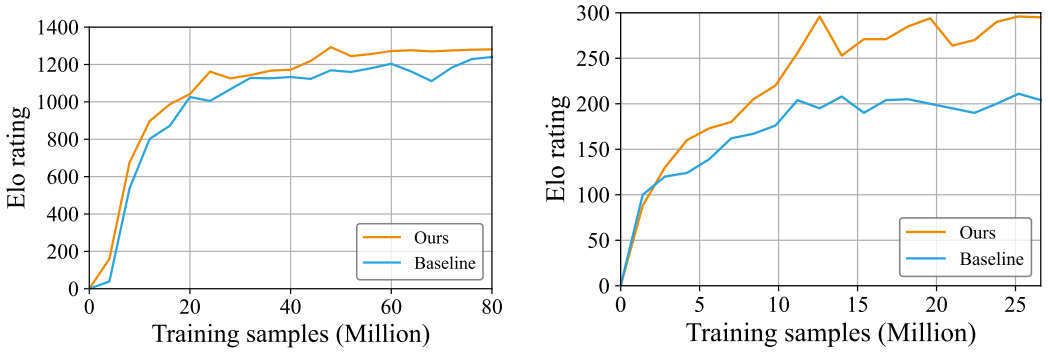
To determine the degree of exploration, we evaluate the efficiency of training across different variation limits V after training in 12M positions. Figure 6a shows that $V = 10$ produces the highest strength and outperforms the $V = 1$ baseline by a 69% win rate under the same number of training samples, thus achieving the best training efficiency among different variations. Comparing $V = 5$ with the baseline, we find that the average LCR increases slightly and the strengths are close because there is only a limited amount of additional exploration. Furthermore, Figure 6b reveals that, with more variations explored, self-play, in general, contains more high-uncertainty positions. However, as V increases further, the degree of exploration begins to saturate. In particular, we observe that the LCRs for $V = 10$ and $V = 15$ are close. Since we collect a fixed number of positions per self-play iteration, increasing the number of variations per game can reduce the diversity of opening variations across games. As a result, while $V = 10$ and $V = 15$ both saturate the LCR within a game, $V = 10$ can achieve a higher win rate since it can explore more diverse opening positions. Based on these observations, we adopt $V = 10$ in the following evaluations.

On the other hand, to demonstrate that the label change rate is an effective metric for branching, we compared three methods: the baseline (no branching), uniformly random branching, and our proposed uncertainty-guided method, as shown in Figure 7. For uniform branching, branching positions are selected uniformly at random with total variations $V = 10$, and the new variation is played by resampling from the MCTS policy, similarly to our method.

Figure 7a shows that our method achieves a 62% win rate against the baseline after training on 12M positions in 19x19 Go. In contrast, random branching results in a 35% win rate against the baseline and a 25.5% win rate against our method, along with the lowest average LCR among all approaches. As shown in Figure 7b, compared to the baseline, the LCR of random branching is lower during the mid-game, while significantly higher in the end-game. This trend reflects the inherent complexity and uncertainty of mid-game positions, where uniformly branching is unable to distinguish between low and high uncertainty positions. Conversely, typically there are fewer choices of moves during the end-game, so high-uncertainty positions can have a chance to be sampled with uniformly random exploration. Overall, uniform random branching explores more lower-uncertainty positions due to the lack of a metric to identify high-uncertainty positions, thus resulting in lower performance. This highlights the limitations of naive branching strategies and the insufficiency of relying solely on the value ensemble over multiple game results. In contrast, our uncertainty-guided branching consistently explores higher uncertainty positions throughout the game, as shown in Figure 7b, leading to more effective exploration and, eventually, to greater training efficiency.

5.4 End-to-End Training Efficiency

In this section, we evaluate the sample efficiency of our method compared to the baseline without branching in the 9x9 and 19x19 Go games with the same amount of compute resources. As shown in Figure 8, our method consistently outperforms the baseline in *both* settings, indicating improved



(a) 9x9 Go.

(b) 19x19 Go.

Figure 8: End-to-end evaluation of sample efficiency. In 9x9 Go, models are trained from scratch, while in 19x19 Go, the performance is relative to the pretrained model. Our method consistently achieves higher strength than the baseline with fewer samples.

training efficiency under the same training samples, and the standard deviation is about 2.1%. In 9x9 Go, our method reaches the same performance as the baseline in 40M and 80M samples using only 29.6M and 50.5M training samples, corresponding to 34.9% and 58.5% improved sample efficiency, respectively. On average, our method requires only 59.6% of training samples to reach the same strength as in the baseline. In particular, the gap between our method and the baseline becomes evident early and persists throughout, with our approach leading by an average of 99.3 Elo rating under the same number of samples. In addition, we visualize the LCR distribution for self-play games in the appendix A.1.1.

Similarly, in 19x19 Go, our method matches the baseline's performance at 5M and 10M samples with only 3.9M and 6.8M samples, yielding 29.0% and 47.3% improvements in sample efficiency. This shows that, even in the later phases of training, our proposed method maintains a high sample efficiency. On average, our method achieves the same strength using only 66.4% of the samples required by the baseline, and our approach maintains a consistent lead to the baseline by 75 Elo rating under the same number of samples. Overall, our end-to-end evaluations highlight better sample efficiency when adopting our proposed uncertainty-guided exploration during self-play.

6 Discussion

Our LCR analysis requires the game result to be binary or discrete since we assume the Bernoulli distribution for the result. This sets our approach apart from many prior works [20, 23, 28] on uncertainty in deep reinforcement learning, which typically assume more general or continuous distributions such as Gaussian. Our Bernoulli setting allows us to derive a closed-form LCR metric and significantly simplifies uncertainty analysis. Interestingly, in this case, there is a coincidence for the Bernoulli distribution that the value variance in Equation 1 and the LCR in Equation 2, share a similar form, and this relationship does not hold for general distributions. Moreover, to extend the Bernoulli setting to continuous outcomes, one can define a Bernoulli random variable X based on whether the outcome exceeds a threshold: $Y := \mathbf{1}_{(X>s)}$, where s is a threshold score. This formulation may also be useful in hard-exploration tasks since completing a sub-goal can be modeled as a Bernoulli event. Alternatively, the LCR can be generalized by defining it as $\Pr(|X_i - X_j| > t)$, where $t > 0$ is the threshold of measuring significant differences between outcomes.

7 Concluding Remarks

In this work, we proposed an uncertainty-guided exploration framework that addresses key inefficiencies in AlphaZero training. By branching out from high-uncertainty positions and exploring multiple variations, our method achieves two complementary goals: (1) reducing the variance of the value label by ensembling multiple game results in a tree-structured self-play setting, and (2) exploring previously overlooked yet promising decisions that may contain potential improvements. In general, our method is applicable to episodic reinforcement learning environments that allow the algorithm to return to a specific state during acting. In our work, AlphaZero inherently assumes access to a perfect

simulator. However, such a simulator is not strictly required, as previous work [31] has already shown that returning to earlier states can also be accomplished by *restoring* memory states. Our future work includes extending the LCR analysis to continuous outcomes or a generalized form, as discussed in the previous section, or applying LCR to multiple value heads [35] to guide exploration and improve interpretability for uncertainty. We believe that our analytical formulation for LCR opens up a promising direction for future research on uncertainty exploration in reinforcement learning.

Acknowledgments and Disclosure of Funding

This research is supported in part by DOE grant DE-SC0023186, NSF grants 2453653 and 2211018, and National Science and Technology Council of Taiwan project number NSTC114-2221-E-001-019-MY3. The opinions expressed in this paper are those of the authors and not necessarily the views of the NSF or the DOE. We thank the Taiwan National Center for High-performance Computing (NCHC) for providing computational and storage resources. The authors would also like to thank Cyan Subhra Mishra for helpful discussions and the anonymous reviewers for their valuable comments.

References

[1] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[2] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *Computers and Games: 5th International Conference, CG 2006, Turin, Italy, May 29-31, 2006. Revised Papers 5*, pp. 72–83, Springer, 2007.

[3] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Machine Learning: ECML 2006: 17th European Conference on Machine Learning Berlin, Germany, September 18-22, 2006 Proceedings 17*, pp. 282–293, Springer, 2006.

[4] S. Cheng, M. T. Kandemir, and D.-Y. Hong, "Speculative monte-carlo tree search," *Advances in Neural Information Processing Systems*, vol. 37, pp. 88664–88683, 2024.

[5] L.-C. Lan, T.-R. Wu, I.-C. Wu, and C.-J. Hsieh, "Learning to stop: Dynamic simulation monte-carlo tree search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 259–267, 2021.

[6] A. Trudeau and M. Bowling, "Targeted search control in alphazero for effective policy improvement," in *Adaptive Agents and Multi-Agent Systems*, 2023.

[7] D. J. Wu, "Accelerating self-play learning in go," *arXiv preprint arXiv:1902.10565*, 2019.

[8] Y. Mei, J. Gao, W. Ye, S. Liu, Y. Gao, and Y. Wu, "Speedyzero: Mastering atari with limited data and time," in *The Eleventh International Conference on Learning Representations*, 2023.

[9] D. Zhao, S. Tu, and L. Xu, "Efficient learning for alphazero via path consistency," in *International Conference on Machine Learning*, pp. 26971–26981, PMLR, 2022.

[10] D. Willemsen, H. Baier, and M. Kaisers, "Value targets in off-policy alphazero: a new greedy backup," *Neural Computing and Applications*, vol. 34, pp. 1801 – 1814, 2021.

[11] D. J. Soemers, E. Piette, M. Stephenson, and C. Browne, "Manipulating the distributions of experience used for self-play learning in expert iteration," in *2020 IEEE Conference on Games (CoG)*, pp. 245–252, IEEE, 2020.

[12] G. Farnebäck, "GTP - Go Text Protocol." "`https://www.lysator.liu.se/~gunnar/gtp/`", 2002.

[13] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.

[14] D. Zhao, S. Tu, and L. Xu, "Generalized weighted path consistency for mastering atari games," *Advances in Neural Information Processing Systems*, vol. 36, pp. 50346–50357, 2023.

[15] W. Ye, S. Liu, T. Kurutach, P. Abbeel, and Y. Gao, "Mastering atari games with limited data," *Advances in neural information processing systems*, vol. 34, pp. 25476–25488, 2021.

[16] S. Wang, S. Liu, W. Ye, J. You, and Y. Gao, "Efficientzero v2: Mastering discrete and continuous control with limited data," in *International Conference on Machine Learning*, 2024.

[17] A. Borges and A. Oliveira, "Combining off and on-policy training in model-based reinforcement learning," *arXiv preprint arXiv:2102.12194*, 2021.

[18] Y. Oh, J. Shin, E. Yang, and S. J. Hwang, "Model-augmented prioritized experience replay," in *International Conference on Learning Representations*, 2022.

[19] Y. Jiang, J. Z. Kolter, and R. Raileanu, "Uncertainty-driven exploration for generalization in reinforcement learning," in *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

[20] S. Lahlou, M. Jain, H. Nekoei, V. I. Butoi, P. Bertin, J. Rector-Brooks, M. Korablyov, and Y. Bengio, "DEUP: Direct epistemic uncertainty prediction," *Transactions on Machine Learning Research*, 2023.

[21] V. Mai, K. Mani, and L. Paull, "Sample efficient deep reinforcement learning via uncertainty estimation," in *International Conference on Learning Representations*, 2022.

[22] S. Aravindan and W. S. Lee, "State-aware variational thompson sampling for deep q-networks," in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, p. 124–132, International Foundation for Autonomous Agents and Multiagent Systems, 2021.

[23] W. R. Clements, B. Van Delft, B.-M. Robaglia, R. B. Slaoui, and S. Toth, "Estimating risk and uncertainty in deep reinforcement learning," *arXiv preprint arXiv:1905.09638*, 2019.

[24] M. Fortunato, M. G. Azar, B. Piot, J. Menick, M. Hessel, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg, "Noisy networks for exploration," in *International Conference on Learning Representations*, 2018.

[25] Y. Sun, F. Gomez, and J. Schmidhuber, "Planning to be surprised: Optimal bayesian exploration in dynamic environments," in *International conference on artificial general intelligence*, pp. 41–51, Springer, 2011.

[26] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak, "Planning to explore via self-supervised world models," in *International conference on machine learning*, pp. 8583–8592, PMLR, 2020.

[27] P. Shyam, W. Jaśkowski, and F. Gomez, "Model-based active exploration," in *International conference on machine learning*, pp. 5779–5788, PMLR, 2019.

[28] I. Osband, J. Aslanides, and A. Cassirer, "Randomized prior functions for deep reinforcement learning," *Advances in neural information processing systems*, vol. 31, 2018.

[29] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," *Advances in neural information processing systems*, vol. 29, 2016.

[30] R. Y. Chen, S. Sidor, P. Abbeel, and J. Schulman, "Ucb exploration via q-ensembles," *arXiv preprint arXiv:1706.01502*, 2017.

[31] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune, "First return, then explore," *Nature*, vol. 590, no. 7847, pp. 580–586, 2021.

[32] A. Tavakoli, V. Levdik, R. Islam, C. M. Smith, and P. Kormushev, "Exploring restart distributions," *arXiv preprint arXiv:1811.11298*, 2018.

[33] C. E. Luis, A. G. Bottero, J. Vinogradska, F. Berkenkamp, and J. Peters, "Model-based uncertainty in value functions," in *International Conference on Artificial Intelligence and Statistics*, pp. 8029–8052, PMLR, 2023.

[34] R. Coulom, "Computing "elo ratings" of move patterns in the game of go," *ICGA journal*, vol. 30, no. 4, pp. 198–208, 2007.

[35] T.-R. Wu, I.-C. Wu, G.-W. Chen, T.-H. Wei, H.-C. Wu, T.-Y. Lai, and L.-C. Lan, "Multilabeled value networks for computer go," *IEEE Transactions on Games*, vol. 10, no. 4, pp. 378–389, 2018.

[36] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[38] I. Danihelka, A. Guez, J. Schrittwieser, and D. Silver, "Policy improvement by planning with gumbel," in *International Conference on Learning Representations*, 2022.

[39] D. J. Wu, "Networks for kata1." "`https://katagotraining.org/networks/`", 2020.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction are aligned with the paper's method and evaluation results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitation and the scope of the work in Section 6 and 7.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Theory proofs and analysis are provided in Section 4.1 and appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Detailed experimental settings are provided in Section 5.1 and appendix, and all experiments are results of multiple evaluations.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The model checkpoints are provided in `https://huggingface.co/chengscott/ugb_zero`.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Detailed experimental settings are provided in Section 5.1 and appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Experiment statistical significance are provided in Section 5.1 and in each evaluation.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Experiment compute resources are provided in Section 5.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: All authors conform with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The paper focuses on improving training efficiency.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper focuses on improving training efficiency.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All assets are properly credited and cited in the paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs.

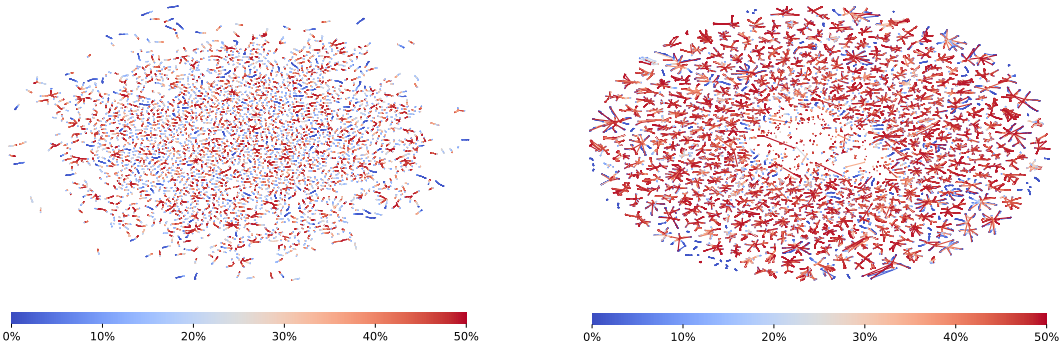
Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Appendix / Supplemental Material

A.1 Additional Evaluation

A.1.1 Visualization



(a) Baseline. (b) Ours.

Figure 9: Visualization of the LCR distribution among self-play games. The heatmap color indicates the LCR, with red representing higher values. Each line connects adjacent moves within the same game variation.

Figure 9 shows the visualization of the LCR distribution for both the baseline (without branching) and our proposed method with $V = 10$ on 9x9 Go after training on 60M samples. We project self-play game positions into two dimensions using t-SNE [36], connect successive positions from the same game variation with lines, and color them according to their LCR values.

Figure 9a shows the LCR distribution of the baseline method, which consists of unbranched lines, each representing a game. Note that some lines may overlap due to the projection from a high-dimensional space. This visualization matches the earlier result shown in Figure 5, where for most of the games, the LCR is high during the game's early stage and gradually decreases toward the end-game; i.e., one end of the line is red and the other end is blue. Interestingly, some lines remain entirely blue, indicating persistently low uncertainty throughout those games. These positions likely correspond to early losses in games, resulting from exploring obviously losing positions. On the other hand, Figure 9b presents the LCR distribution of our proposed method, where most games form a spike-shaped cluster due to branching. Overall, we observe that our self-play games contain more red lines than the baseline, indicating a higher number of high-uncertainty positions. In particular, we do not branch from positions that are entirely blue (i.e., low uncertainty).

A.1.2 LCR Threshold Sensitivity

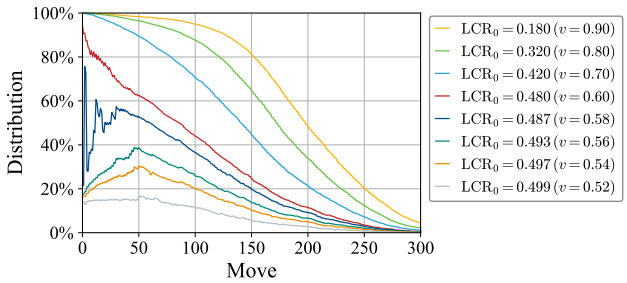


Figure 10: Percentage of positions whose LCR exceeds a LCR threshold (LCR_0) for each move. Each curve corresponds to a different threshold with its corresponding value in parentheses. In particular, $v = 0.6$ is the initial value for an empty position.

To demonstrate the robustness of various LCR thresholds, Figure 10 shows the distribution of positions whose LCR exceeds a given threshold. As the threshold decreases, the percentage of

selected positions gradually increases at each move, which aligns with our intuition: highly uncertain positions often occur near other uncertain positions, whether preceding or following. Additionally, since $v = 0.58$ is close to the initial win rate (0.6), the LCR of early-game positions often lie near the LCR threshold, leading to a greater variability in the distribution. These results suggest that the LCR threshold is a robust hyperparameter, as slight changes in the threshold often result in only a small number of neighboring uncertain positions and do not cause abrupt changes in the selection volume.

A.1.3 High-Uncertainty Position Distribution

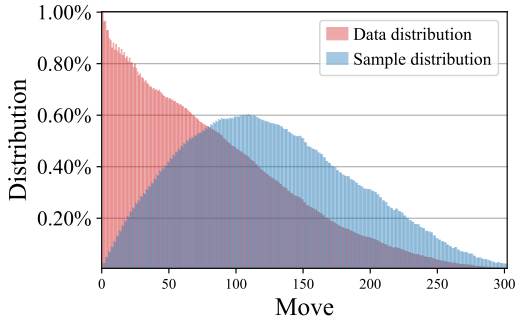
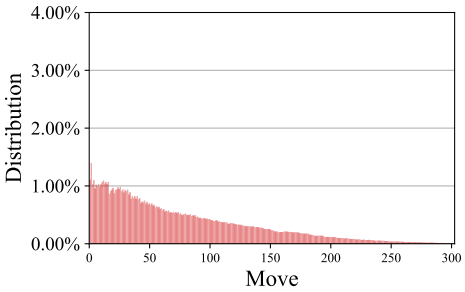


Figure 11: Distribution over move numbers for positions where the LCR exceeds the threshold (0.48). The red bars represent the data distribution (original distribution over all positions), while the blue bars show the sampling distribution, obtained by drawing from the data distribution with weights proportional to the move number.

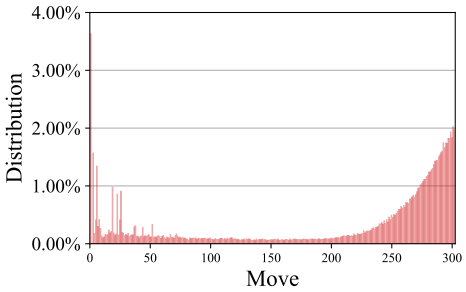
To gain additional insights into our proposed branching strategy, Figure 11 shows the distribution of high-uncertainty positions across different move numbers. The red bars represent the data distribution, corresponding to positions exceeding the LCR threshold, while the blue bars show the sampling distribution obtained by weighting the data distribution according to move number, as defined in Algorithm 1. We observe that the data distribution decreases steadily as the game progresses, reflecting the sparser orange scatter points above the LCR threshold in Figure 5 during later stages. In contrast, the sampling distribution is concave, starting low in the early moves, rising to a peak around move 110, and then gradually declining.

Based on these observations, without branching, exploration of high-uncertainty moves during self-play would implicitly follow the red distribution. With our LCR-guided branching, we explicitly identify high-uncertainty positions and increase the probability of exploring them, especially in the mid-to-late game, as evidenced by the region enclosed by the data and sample distribution. In general, this analysis can be applied to various uncertainty metrics.

A.1.4 Alternative Uncertainty Metrics

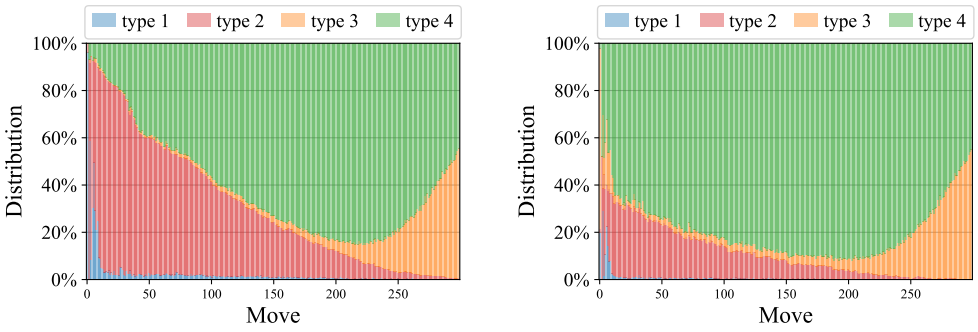


(a) `MinV`: positions in the lowest 10% of $v(s) - 0.5$, where $v(s)$ is the MCTS value.



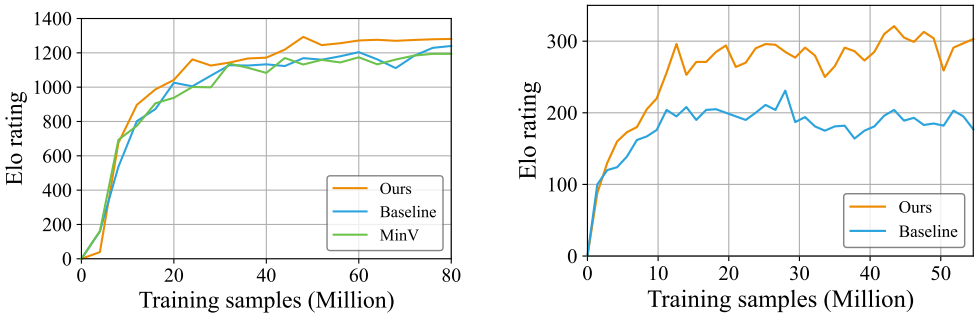
(b) `MinMaxP`: positions in the lowest 10% of $\max(\pi(\cdot \mid s))$, where $\pi(\cdot \mid s)$ is the MCTS policy.

Figure 12: Distribution over move numbers for positions identified as high-uncertainty by the alternative uncertainty metrics, `MinV` and `MinMaxP`.



(a) (LCR, MinMaxP) (b) (MinV, MinMaxP)

Figure 13: Distribution of positions across four uncertainty categories, defined by combinations of value and policy uncertainty shown in each subfigure. The four types are defined as follows: type 1 has high value uncertainty and high policy uncertainty; type 2 has high value uncertainty and low policy uncertainty; type 3 has low value uncertainty and high policy uncertainty; and type 4 has low value uncertainty and low policy uncertainty.



(a) 9x9 Go with MinV. (b) 19x19 Go with additional training samples.

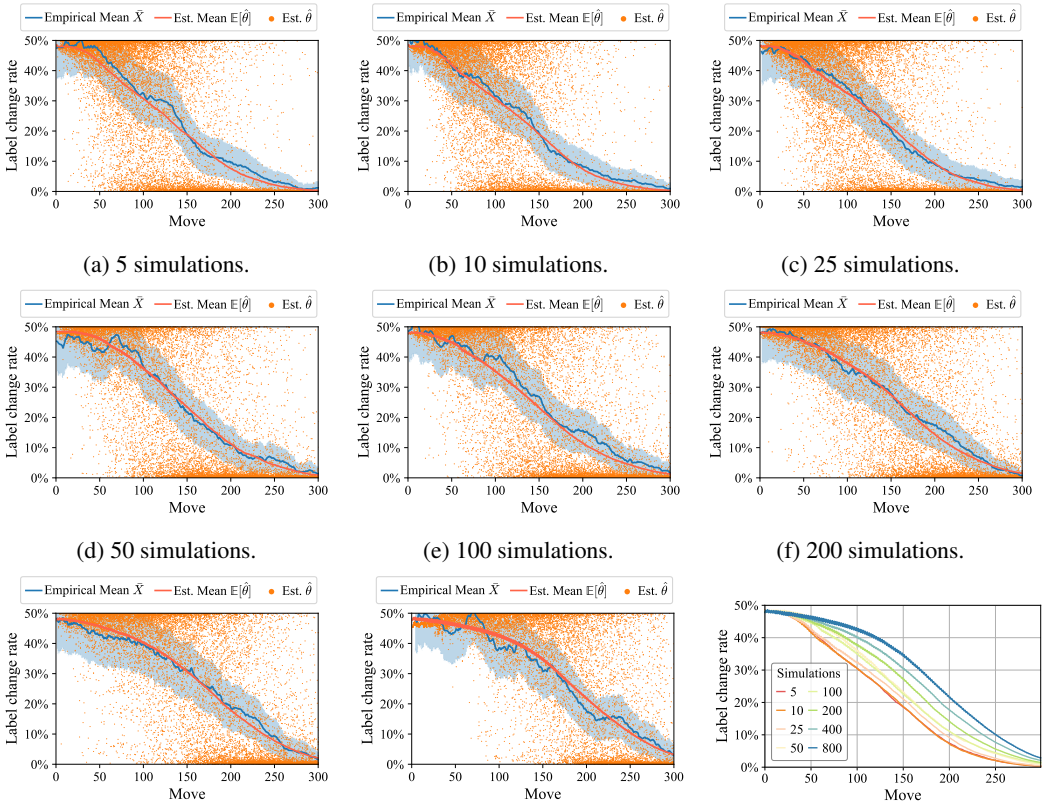
Figure 14: End-to-end evaluation of sample efficiency.

In addition to our proposed LCR metric, we define two alternative uncertainty metrics, MinV and MinMaxP, based on MCTS value and policy, respectively. Figure 12 presents the distribution of high-uncertainty positions under each metric, similar to Figure 11. The MinV distribution shows a similar distribution to LCR, as both are value-based metrics. In contrast, MinMaxP exhibits an opposite trend: its mass concentrates in the endgame, where game results are nearly determined. This is because multiple moves in the endgame often lead to the same game result, corresponding to high policy uncertainty. As expected, MinMaxP also identifies some early-game positions with high policy uncertainty, but these are much fewer compared to the endgame.

Moreover, Figure 13 shows the distribution over each move of uncertainty categorized into four types. Similar to our earlier observations, value-based metrics identify high-uncertainty positions primarily in the early game, whereas policy-based metrics detect them mainly in the endgame. As a result, when only using the MinMaxP metric, the end-to-end training fails. Figure 14a shows the end-to-end training sample efficiency with MinV metric. We observe that the MinV shows a similar efficiency to ours during the early stage of training, but later slows down and approaches the baseline. This is because simply using a fixed percentage of the lowest value alone can only capture a certain amount of high-uncertainty positions, which lacks the flexibility to identify uncertainty per position, as in our proposed LCR metric. On the other hand, we provide more training samples to Figure 8b in Figure 14b. Our method still outperforms the baseline at a later phase of training.

A.1.5 Scaling of the Label Change Rate

In addition to Figure 5, Figure 15 shows the evaluations of empirical and analytical LCR estimation for simulations ranging from 5 to 800. Among all simulations, all curves show a similar trend, and the



(a) 5 simulations. (b) 10 simulations. (c) 25 simulations.

(d) 50 simulations. (e) 100 simulations. (f) 200 simulations.

(g) 400 simulations. (h) 800 simulations. (i) Comparison of $\mathbb{E}[\hat{\theta}]$ curves from panels (a-h).

Figure 15: Label change rate (LCR) with different simulation counts. The empirical mean $\bar{X}$ (blue line) is the average of observed label changes across positions, with a 95% confidence interval shown as the shaded blue region. Analytical estimates $\mathbb{E}[\hat{\theta}]$ (orange line) are averaged from individual LCR estimators $\hat{\theta}$, shown as orange scatter points.

empirical and estimated means are closely aligned. On the other hand, in Section 5.2, we computed the RMSE by comparing the empirical mean (blue line) with the estimated mean LCR (orange line) in Figure 5. Another way to compute the RMSE is based on per-position comparisons. Specifically, we simulate 10 game results from the same position to obtain an empirical change rate, and then compare this to the estimated LCR. The RMSE computed by comparing the mean for 5, 10, 25, 50, 100, 200, 400, 800 simulations is 0.07, 0.08, 0.05, 0.04, 0.03, 0.02, 0.02, 0.02, respectively; the RMSE computed by per-position comparison is 0.14, 0.13, 0.12, 0.11, 0.12, 0.11, 0.18, 0.14, respectively. The RMSE for the per-position comparison is higher, as each position requires more simulations for accurate estimation.

Furthermore, Figure 15i shows the comparison of the estimated mean among different simulations. When the simulation count is smaller, the LCR is lower, meaning the game is likely to lose earlier and faster. In contrast, when doubling the simulations, the LCR at the same move is increased almost linearly, and thus the game result is less determined compared to a smaller simulation. This evaluation not only demonstrates that our proposed LCR scales well across simulations but also suggests that it is a potential metric for detecting the strength of a position.

A.2 Proofs

In this section, we provide the proofs and derivations for Section 4.1, corresponding to appendices A.2.1, A.2.2 and A.2.3. In addition to the Bayesian inference presented earlier, we also present an alternative formulation of Bayesian inference in appendix A.2.4.

A.2.1 Variance Reduction

Suppose $Z \sim \text{Bernoulli}(w)$, meaning that Z is a random variable that takes the value 1 with probability w and 0 with probability $1 - w$. Let $Z_1, \ldots, Z_n \overset{\text{i.i.d.}}{\sim} Z$, and define the sample mean as $\bar{Z} = \sum_{i=1}^{n} Z_i/n$. Then,

$$\text{Var}[\bar{Z}] = \text{Var}\left[\frac{1}{n}\sum_{i=1}^{n} Z_i\right] = \frac{1}{n^2}\sum_{i=1}^{n}\text{Var}[Z_i] = \frac{1}{n^2}\sum_{i=1}^{n} w(1-w) = \frac{w(1-w)}{n}. \tag{6}$$

This result follows from the fact that the variance of a $\text{Bernoulli}(w)$ random variable is $w(1-w)$, and that Z_i are independent and identically distributed.

A.2.2 Uncertainty Estimator

According to the definition of the label change rate (LCR) in Section 4.1, we compute the LCR for two independently-sampled variables Z_i and Z_j, where $i \neq j$, as follows:

$$\begin{aligned}
\Pr(Z_i \neq Z_j) &= \Pr(Z_i = 0, Z_j = 1) + \Pr(Z_i = 1, Z_j = 0) && (7)\\
&= (1-w)w + w(1-w) && (8)\\
&= 2w(1-w). && (9)
\end{aligned}$$

Similarly, we can compute label *unchanged* rate as:

$$\begin{aligned}
\Pr(Z_i = Z_j) &= \Pr(Z_i = 0, Z_j = 0) + \Pr(Z_i = 1, Z_j = 1) && (10)\\
&= (1-w)^2 + w^2. && (11)
\end{aligned}$$

Thus, we can define a random variable $X = \mathbf{1}_{Z_i \neq Z_j}$ for some $i \neq j$. This is an indicator variable that equals 1 if $Z_i \neq Z_j$, and 0 otherwise:

$$X = \begin{cases} 1 & \text{if } Z_i \neq Z_j, \\ 0 & \text{if } Z_i = Z_j. \end{cases} \tag{12}$$

Moreover, we can write the probability mass function of X as:

$$p(X = x \mid w) = \begin{cases} 2w(1-w), & \text{if } x = 1 \\ w^2 + (1-w)^2, & \text{if } x = 0 \end{cases} \tag{13}$$

and the expectation of X is

$$\mathbb{E}[X] = \Pr(Z_i \neq Z_j) = 2w(1-w). \tag{14}$$

It is important to note that these results hold for any $i \neq j$. In the following, we explore how different choices of i and j lead to different Bayesian formulations.

A.2.3 Bayesian Inference

Suppose that we observe a sequence of label change rates, denoted as a dataset $D = \{X_1, X_2, \ldots, X_m\}$, where $X_i \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(\theta)$ and θ represents the random variable for a label change. In particular, we choose $X_i = \mathbf{1}_{Z_{2i} \neq Z_{2i-1}}$ using paired game outcomes for $i \in [1, m]$,

where $m = \lfloor n/2 \rfloor$. Since the X_i are i.i.d., the joint likelihood of the data is:

$$
\begin{aligned}
p(D \mid \theta) &= p(X_1, X_2, \ldots, X_m \mid \theta) && (15)\\
&= \prod_{i=1}^{m} p(X_i \mid \theta) && (16)\\
&= \prod_{i=1}^{m} \theta^{X_i}(1-\theta)^{(1-X_i)} && (17)\\
&= \theta^{\sum_{i=1}^{m} X_i}(1-\theta)^{\sum_{i=1}^{m}(1-X_i)} && (18)\\
&= \theta^{s}(1-\theta)^{m-s}, && (19)\\
& && (20)
\end{aligned}
$$

where $s = \sum_{i=1}^{m} X_i$ is the number of observed label changes.

On the other hand, according to Bayes' theorem, the posterior distribution is given by:

$$
p(\theta \mid D) = \frac{p(\theta)p(D \mid \theta)}{\displaystyle\int p(\theta)p(D \mid \theta)\, d\theta}, \tag{21}
$$

where $p(\theta)$ is the conjugate prior for the Bernoulli likelihood, which is the Beta distribution. Thus, we express the prior probability distribution as $\theta \sim \mathrm{Beta}(\alpha, \beta)$. Namely,

$$
p(\theta) = \frac{1}{B(\alpha,\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1} \propto \theta^{\alpha-1}(1-\theta)^{\beta-1} \tag{22}
$$

where $B(\alpha, \beta)$ is the Beta function. We also have

$$
\begin{aligned}
\mathbb{E}[\theta] &= \frac{\alpha}{\alpha+\beta} && (23)\\
&= \Pr(Z_{2i} \neq Z_{2i-1}) = 2w(1-w), && (24)
\end{aligned}
$$

where Equation 23 is the expectation of $Beta(\alpha, \beta)$ and Equation 24 is derived from Equation 14 by definition. By applying Bayes' theorem, the posterior distribution is proportional to the product of the prior and the likelihood:

$$
p(\theta \mid D) \propto p(D \mid \theta) \cdot p(\theta) \propto \theta^{s}(1-\theta)^{m-s} \cdot \theta^{\alpha-1}(1-\theta)^{\beta-1} = \theta^{\alpha+s-1}(1-\theta)^{\beta+m-s-1}. \tag{25}
$$

Thus, the posterior distribution is:

$$
\theta \mid D \sim \mathrm{Beta}(\alpha + s, \beta + m - s) \tag{26}
$$

The posterior mean, which serves as the Bayesian estimate of θ, namely the posterior label change rate (LCR), is given by:

$$
\mathbb{E}[\theta \mid D] = \frac{\alpha + s}{\alpha + \beta + m}. \tag{27}
$$

A.2.4 Alternative Bayesian Inference

Previously, we only used $m = n/2$ of the n observed label changes for the Bayesian update. In this section, we study the option of using $m = n - 1$ observed label changes for Bayesian update, which breaks the independent assumption. In particular, we choose $X_i = \mathbf{1}_{Z_i \neq Z_{i-1}}$ using the previous outcome to calculate the label change rate for $i \in [2, n]$. In this case, X_is are still identically distributed to $\mathrm{Bernoulli}(\theta)$ but not independent. Before we dive into the calculation of $p(D \mid \theta)$, we first define $\oplus$ as the xor operation, namely, the addition under mod 2. Hence, we have $X_i = Z_i \neq Z_{i-1} = Z_i \oplus Z_{i-1}$ and thus,

$$
\begin{aligned}
Z_i &= Z_{i-1} \oplus X_i && (28)\\
&= Z_1 \oplus (X_2 \oplus \cdots \oplus X_i). && (29)
\end{aligned}
$$

Then, we have the following probability:

$$\Pr(Z_1 = z_1) = \theta^{z_1}(1-\theta)^{1-z_1}, \tag{30}$$

$$\Pr(Z_k = z_k \mid X_1 = x_1) = \Pr(Z_k = z_1 \oplus s_k \mid X_1 = x_1) = \theta^{z_1 \oplus s_k}(1-\theta)^{1-z_1 \oplus s_k}, \tag{31}$$

where $s_i = x_2 \oplus \cdots \oplus x_i$, and $z_i = z_1 \oplus s_i$.

Then, the joint probability of the dataset is

$$\Pr(X_2 = x_2, \ldots, X_n = x_n) = \sum_{z_1=0}^{1} \Pr(Z_1 = z_1) \Pr(X_2 = x_2, \ldots, X_n = x_n \mid Z_1 = z_1) \tag{32}$$

$$= \sum_{z_1=0}^{1} \Pr(Z_1 = z_1) \Pr(X_2 = x_1 \oplus s_2, \ldots, X_n = x_1 \oplus s_n \mid Z_1 = z_1) \tag{33}$$

$$= \sum_{z_1=0}^{1} \Pr(Z_1 = z_1) \prod_{k=2}^{n} \Pr(X_k = x_1 \oplus s_k \mid Z_1 = z_1) \tag{34}$$

$$= \sum_{z_1=0}^{1} \theta^{z_1}(1-\theta)^{1-z_1} \prod_{k=2}^{n} \theta^{z_1 \oplus s_k}(1-\theta)^{1-z_1 \oplus s_k} \tag{35}$$

$$= (1-\theta) \prod_{k=2}^{n} \theta^{s_k}(1-\theta)^{1-s_k} + \theta \prod_{k=2}^{n} \theta^{1-s_k}(1-\theta)^{s_k}, \tag{36}$$

since we have $z_1 \oplus s_k = s_k$ when $z_1 = 0$, and $z_1 \oplus s_k = 1 - s_k$ when $z_1 = 1$.

Let $m_1 = \sum_{k=2}^{n} s_k$ be the number of ones in $\{s_2, \ldots, s_n\}$, and $m_0 = n - m_1$. Then, we can rewrite the joint probability as:

$$P(X = x) = \theta^{m_1}(1-\theta)^{m_0} + \theta^{m_0}(1-\theta)^{m_1} \tag{37}$$

Suppose that we still assume a Beta prior, that is, $\theta \sim \mathrm{Beta}(\alpha, \beta)$. By applying Bayes' theorem, the posterior distribution can be expressed as:

$$p(\theta \mid D) \propto p(D \mid \theta) \cdot p(\theta) \propto (\theta^{m_1}(1-\theta)^{m_0} + \theta^{m_0}(1-\theta)^{m_1}) \cdot \theta^{\alpha-1}(1-\theta)^{\beta-1} \tag{38}$$

$$\propto \theta^{m_1+\alpha-1}(1-\theta)^{m_0+\beta-1} + \theta^{m_0+\alpha-1}(1-\theta)^{m_1+\beta-1}. \tag{39}$$

Thus, the posterior distribution becomes

$$\theta \mid D \sim \mathrm{Beta}(m_1 + \alpha, m_0 + \beta) + \mathrm{Beta}(m_0 + \alpha, m_1 + \beta). \tag{40}$$

And the corresponding posterior LCR is:

$$\mathbb{E}[\theta \mid D] = \frac{(\alpha + m_1)B(m_1 + \alpha + m_0 + \beta) + (\alpha + m_0)B(m_0 + \alpha, m_1 + \beta)}{(\alpha + \beta + n)(B(m_1 + \alpha, m_0 + \beta) + B(m_0 + \alpha, m_1 + \beta))}. \tag{41}$$

A.3 Hyperparameters

Table 1 shows the hyperparameters for self-play and training. In 9x9 Go, the model consists of 6 ResNet residual blocks [37] with 96 channels each, while in 19x19 Go, the model consists of 5 Nested Bottleneck blocks [38] with 192 channels each, following the model architecture in the latest KataGo models [39]. Our pretrained model follows previous work [7] to train on the first 100M samples from the public repository [39] and set a baseline of an Elo rating of 9958. Moreover, the LCR threshold for Equation 4 is $\mathrm{LCR}_0 = 0.48$ for both games.

Table 1: Hyperparameters for self-play and training.

		9x9 Go Training	19x19 Go Training	19x19 Go Pre-training
Self-play	MCTS simulation	400		
	MCTS c_{puct}	1.25		
	Dirichlet noise ratio ε	0.25		-
	Dirichlet parameter α	0.12	0.03	
	G states per iteration	0.3M	1.2M	
Training	Optimizer	SGD		
	Batch size	512		
	Optimizer: momentum	0.9		
	Optimizer: weight decay	1e-4		
	Optimizer: learning rate	0.01		
	Sample factor	2	1	2