INFERENCE-AWARE FINE-TUNING FOR BEST-OF-N SAMPLING IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent studies indicate that effectively utilizing inference-time compute is crucial for attaining good performance from large language models (LLMs). Specifically, the Best-of-N (BoN) inference strategy, where an LLM generates multiple responses and a verifier selects the best, has shown strong empirical performance. Motivated by this, we develop a novel *inference-aware fine-tuning* paradigm, which encompasses the *BoN-aware inference* framework as a special case. We devise the first imitation learning and reinforcement learning (RL) methods for fine-tuning LLMs using BoN, overcoming the challenging, non-differentiable argmax operator within BoN. We empirically demonstrate that our BoN-aware models *implicitly* learn a meta-strategy, which interleaves best responses with more diverse responses that might be better suited to a test-time input—a process reminiscent of the exploration-exploitation trade-off in RL. Our experiments demonstrate the effectiveness of BoN-aware fine-tuning in terms of improved performance and inference-time compute. In particular, we show that our methods improve the BoN performance of Gemma 2B on Hendrycks MATH from 26.8% to 30.8%, and Pass@N from 60% to 67%.

1 INTRODUCTION

Recent advances in enhancing reasoning capabilities of large language models (LLMs) highlight the potential for improvements using inference-time computation: several independent threads (Lightman et al., 2023; Wu et al., 2024; Kumar et al., 2024; Hosseini et al., 2024) show that by using search, re-ranking, multi-turn revision, and more generally, any approach that makes use of more tokens and computation at inference time, the performance of LLMs on various tasks can be significantly improved—so much that investing in improving inference-time computation might prove more beneficial than increasing model pre-training compute (Snell et al., 2024).

Despite this promise, existing work largely considers using inference-time computation as an optional post hoc design choice, after conventional pre-training and fine-tuning. However, decoupling training and inference-time compute is *not* optimal; for example, if we knew that an LLM can produce more than one candidate solution to a math problem, then it may be better to explore diverse problem-solving strategies, rather than produce all candidates that represent the model's best strategy at solving the problem. Within the context of reasoning problems, these performance gains may be significant, as LLMs often fail in reasoning problems due to their inability to draw complex inferences about the input and their internal knowledge.

In this work, we introduce a new paradigm named inference-aware fine-tuning, which explicitly considers the inference procedure used at inference time, during training. We particularly focus on the Best-of-N (BoN) inference strategy, where the LLM generates multiple candidate responses, and a verifier selects the best one according to some scoring function (Cobbe et al., 2021). Our inference-aware (or BoN-aware) methodology contrasts traditional fine-tuning methods, which overlooks the specific inference strategy. Particularly, BoN-aware fine-tuning incorporates the BoN inference mechanism into the training process itself.

Our contributions are as follows. (1) We formulate the inference-aware and BoN-aware problem, which accounts for an inference strategy during training; (2) We develop a BoN-aware supervised fine-tuning algorithm which aligns a target distribution with the BoN policy distribution; (3) We extend our method to the general BoN-aware reinforcement learning (RL) setup, allowing the policy to learn to solve a downstream task under the BoN inference strategy. We devise specialized

algorithms for scenarios where the environment reward function can be used as the verifier. These algorithms, inspired by methods that optimize Pass@N accuracy, promote implicit exploration and connect with established self-supervised learning algorithms, further enhancing the effectiveness of BoN-aware fine-tuning; (4) We establish a co-scaling behavior for BoN, which quantifies the exploitation-exploration trade-off of BoN with respect to temperature T and number of samples N.

Empirically, we show that BoN-aware fine-tuning significantly improves the performance of LLMs on Hendrycks MATH (Hendrycks et al., 2021) compared to standard fine-tuning (26.8% to 30.8% increase in BoN accuracy, and 60% to 67% increase for Pass@N). We first illustrate how the number of samples N and temperature T in BoN impact performance and provide insights for their optimization. Then, we show that our BoN-aware methods lead to substantial gains across a range of N and T values, highlighting the robustness and generalizability of the learned "meta-strategy" for diversification. Finally, our results suggest that BoN-aware fine-tuning enables the model to solve problems that are beyond the capabilities of standard fine-tuned models, further demonstrating the benefits of aligning training with the inference-time compute strategy.

2 Inference-Aware Fine-Tuning: A Case Study with Best-of-N

Standard fine-tuning methods typically train LLMs to produce the best response for a given prompt. In LLM fine-tuning, a model (or *policy*) is trained via *supervised fine-tuning* (SFT), by maximizing the likelihood w.r.t. ground-truth data. Formally, we search for a policy $\pi : \mathcal{X} \mapsto \Delta_{\mathcal{Y}}$ that maximizes the likelihood $\mathbb{E}_{x\sim P, y\sim\pi^*(y|x)}[\log \pi\,(y|x)]$, where here, $\mathcal{X}$ and $\mathcal{Y}$ are the space of prompts and outputs of an LLM, P is the prompt distribution, and π^* is a distribution of expert responses. Alternatively, the policy can be fine-tuned via *reinforcement learning* (RL) (Schulman et al., 2017): $\max_{\pi\in\Pi} \mathbb{E}_{x\sim P, y\sim\pi(x)}[R(x, y)]$, to align the LLM's behaviors with the reward function $R(x, y)$. While popular, these methods have not taken the LLM's inference-time strategies into the account.

Inference-Aware Fine-Tuning. To address the gap between how LLMs are trained and how they are used at inference time, we develop inference-aware fine-tuning. During inference, the learned policy π is often not directly used; rather some *inference strategy* $I : \Pi \times \mathcal{X} \mapsto \Delta_{\mathcal{Y}}$ is applied to it. For example, I can be the BoN strategy, which samples multiple candidate responses, and selects the best using the score function of some verifier; or I might be a search mechanism (Lightman et al., 2023) or self-correction (Kumar et al., 2024). To account for this inference strategy I, we alter the objective SFT and RL objectives to be "*aware*" of the inference strategy:

$$\max_{\pi\in\Pi} \mathbb{E}_{x\sim P, y\sim\pi^*(y|x)}[\log I(\pi, x)\,(y)], \text{ and} \qquad \text{(Inference-Aware SFT)}$$

$$\max_{\pi\in\Pi} J(\pi) := \mathbb{E}_{x\sim P, y\sim I(\pi,x)}[R(x, y)], \qquad \text{(Inference-Aware RL)}$$

Indeed, Inference-Aware SFT and Inference-Aware RL are aware of the strategy I. In what follows, we focus on the case where the inference strategy is BoN (i.e., $I \equiv$ BoN), in both the SFT and RL setups. As we will later see, this brings about new algorithms for training the policy.

BoN-Aware Problem Formulation. We begin by formulating the BoN strategy. This inference strategy samples N resposnes from a model with some temperature T, and then selects the best one, based on some verifier score. Formally, the BoN inference policy can be written as:

$$I(\pi, x)(y) = \pi_{\text{bon}}(y|x; \pi, r, N, T) := \arg \max_{y'\in\{y_1,\ldots,y_N\}} r(x, y'),\ \text{s.t. } y_i \overset{T}{\sim} \pi(\cdot|x), x \in \mathcal{X}, \qquad (1)$$

where $\overset{T}{\sim}$ is a sample with temperature T, and $r : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$ is a verifier score[1]. In what follows, when r, N, T are clear from context, we write $\pi_{\text{bon}}(y|x; \pi)$. We see that the above strategy defines a class of BoN policies that is different from the learned policy π, demonstrating the gap between training and inference. We inject this class of BoN polices into the Inference-Aware SFT and Inference-Aware RL frameworks to derive the instantiation of inference-aware fine-tuning.

Besides closing the gap between training and inference and mitigating potential discrepancies between the verifier score r and the true reward R, BoN policies provide further benefits. The BoN mechanism introduces *implicit exploration* during training, bypassing the computational burden of

[1]The verifier score r and the true reward R can be related, or even equal, yet we do not make that assumption here. Usually, r is a model trained to predict R, and therefore serves as a proxy of the true reward.

explicit exploration (Cen et al., 2024). Selecting the best of N samples allows the base policy to explore output variations, inducing a controlled exploration that can lead to more robust and generalizable policies, particularly in scaling behavior w.r.t. temperature T and number of samples N.

Optimizing the BoN policy class is notoriously difficult due to the non-differentiability of the $\arg\max$ operator. Although several differentiable top-k operators (Cuturi et al., 2019; Xie et al., 2020) might be exploited in π_{bon}, they induce approximation error, and more importantly, increase the computational cost dramatically. In our work, we derive a variational formulation of the learning problem w.r.t. π_{bon} without top-k operators, allowing us to construct novel algorithms for inference-aware BoN, using both standard supervised imitation learning (Section 3) and RL (Section 4).

Exploration-Exploitation with BoN. Before constructing our BoN-aware methods, we empirically verify the implicit exploration and exploitation properties of BoN. We do this by revealing optimal co-scaling w.r.t. temperature T and number of samples N. Specifically, for a fixed base policy π, at any prompt $x \in \mathcal{X}$ there is an optimal temperature $T^*(x)$ and optimal number of samples $N^*(x)$ which maximize performance of BoN:

$$N^*(x;\pi), T^*(x;\pi) \in \arg\max_{N,T} \mathbb{E}_{y\sim\pi_{\text{bon}}(y|x;\pi,r,N,T)}[R(x,y]].$$

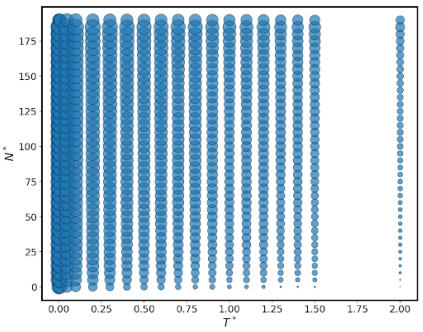


Figure 1: The relationship between the optimal number of samples (N^*) and optimal temperature (T^*) in BoN. The size of each marker at a given (T, N) coordinate indicates the empirical frequency of problems for which that (T, N) pair resulted in the best BoN performance. The plot reveals a trade-off: "easier" problems have small T^* and N^*, while "harder" problems require a larger T^* for exploration and consequently often a larger N^*.

To understand the connection between $N^*(x)$ and $T^*(x)$, we assess the performance of Gemma 2B (Team et al., 2024) on the MATH benchmark (Hendrycks et al., 2021), when applying the BoN inference strategy. Figure 1 shows empirical frequencies of problems, when varying $T^*(x)$ and $N^*(x)$ (larger marker size signifies higher frequency). The figure depicts a tradeoff between T and N, reminiscent of the exploration-expoitation trade-off. When $T^*(x)$ is small, any N is optimal (and particularly also a small N). These "easier" problems do not require heavy exploration (small T^*) and can therefore be more exploitative (small N^*). On the other hand, as T^* increases, the base policy π becomes more stochastic, resulting in more diversity and exploration. These more "difficult" problems, require more exploration (larger T^*), hence less exploitation (larger N^*). Indeed, in such cases, the distribution of N^* shifts to high values. Our results suggest a tradeoff between exploration and exploitation, and further motivates the BoN-aware setup, which can account for this tradeoff uniformly across all samples.

Figure 1 also uncovers a cost-effective recipe for adjusting T and N for optimal BoN performance: we can learn how to fine-tune the model for better inference by simply adjusting these accessible parameters. However, it is important to note that relying solely on model selection has limitations. While this approach offers a computationally inexpensive way to improve BoN's inference-time performance, it may not fully capture the nuances of the LLM's behavior. With sufficient computational resources, general BoN-aware fine-tuning can further unlock performance gains by directly training the LLM to optimize for the exploration-exploitation trade-off of the BoN inference process.

3 Supervised BoN-Aware Fine-Tuning

We begin by developing the BoN-aware SFT framework. Under this setting we assume we do not have access to the true reward, and only wish to maximize the likelihood of a dataset of expert examples. Recall the definition of the BoN policy π_{bon} in Equation (1). The Inference-Aware SFT version of BoN becomes:

$$\max_{\pi\in\Pi} \mathbb{E}_{(x,y)\sim\mathcal{D}}[\log \pi_{\text{bon}}(y \mid x;\pi)], \tag{2}$$

A major difficulty in solving Equation (2) is the non-differentiability of the $\arg\max$ operator in the BoN procedure. To address this, we can use the variational approximation of π_{bon} (see Section A.1)

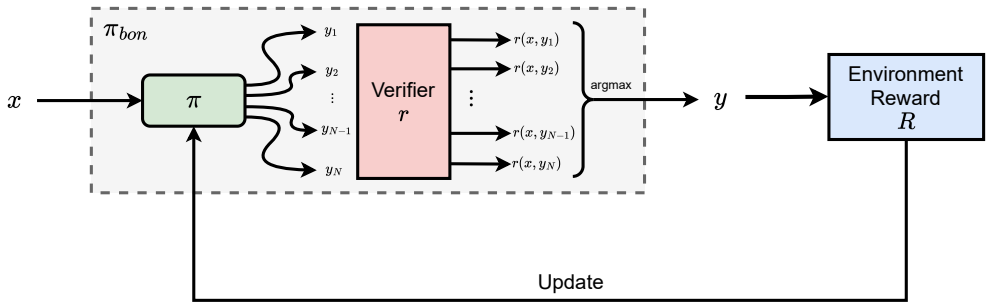


Figure 2: A schematic of BoN-Aware RL fine-tuning. A BoN strategy samples N independent samples from π. A verifier scores each sample, and the best sample is selected. We train the BoN policy using the environment reward, according to Lemma 2

$$\pi_{\text{bon}}(y|x) \propto [\pi(y|x) \cdot \exp(\lambda_N Q_\pi(x, y))], \tag{3}$$

where $Q_\pi(x, y) = \mathbb{E}_{y' \sim \pi(\cdot|x)} \left[\mathbf{1}_{r(x,y) \geqslant r(x,y')}\right]$ is the expected *win-rate* over base policy π, characterizing the probability for which a response y outperforms the responses generated by the base over the verifier score r. The constant $\lambda_N > 0$ is a solution of a 1D-search problem (Gui et al., 2024) (see details in Appendix A.1). It can be shown that λ_N is monotonically increasing in N, and $\lambda_N \propto N$ approximately for large N. Plugging the variational form of Equation (3) into Equation (2) yields:

$$\max_{\pi \in \Pi} \mathbb{E}_{(x,y)\sim\mathcal{D}} [\log \pi_{\text{bon}}(y|x)] := \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\underbrace{\log \pi(y|x)}_{\text{Likelihood}} + \underbrace{\lambda_N \cdot Q_\pi(x, y) - \log Z_\pi(x)}_{\text{Inference-Awareness}} \right], \tag{4}$$

where $Z_\pi(x) = \mathbb{E}_{\pi(y|x)} [\exp(\lambda_N \cdot Q_\pi(x, y))]$ is the partition function.

The above optimization problem reveals two term. While the first term tries to push the base policy π into maximizing the likelihood of the data, the second term regularizes the policy to be more exploratory by increasing the data win rate over the policy. This in turn accounts for the sampling in BoN. For data efficiency when estimating the win rate $Q_\pi(x, y)$ we leverage a common practice in modeling pairwise preferences (Rafailov et al., 2023) to approximate the win rate with its "softened" counterpart: $Q_\pi(x, y) \approx \mathbb{E}_{y' \sim \pi(\cdot|x)} [\sigma(r(x, y) - r(x, y'))]$, where σ is the sigmoid function. Next, we exploit properties of policy gradient (Sutton et al., 1999) and the gradient of energy-based policies (Rafailov et al., 2024) to derive the gradient for Equation (4) (see Appendix A.2 for proof):

Lemma 1 (BoN-SFT). *The gradient of Equation (4) w.r.t. LLM parameters $\theta \in \Theta$ of π is given by* $\mathbb{E}_{(x,y)\sim\mathcal{D}} [\nabla_\theta f(x, y; \theta)] - \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_{bon}(\cdot|x)} [\nabla_\theta f(x, y; \theta)]$, *where*

$$\nabla_\theta f(x, y; \theta) := \nabla_\theta \log \pi_\theta(y|x) + \lambda_N \cdot \mathbb{E}_{y' \sim \pi_\theta} [\nabla_\theta \log \pi_\theta(y'|x) \cdot \sigma(r(x, y) - r(x, y'))]. \tag{5}$$

Our formulation circumvents the non-differentiability of the BoN distribution, allowing solution of BoN-SFT via standard gradient-ascent algorithms. The individual terms of the gradient imply the following: **(1)** π clones the expert behavior by maximizing its likelihood over $\mathcal{D}$; **(2)** it aligns with the verifier score ranking, which assigns a high win-rate to the expert over the base; **(3)** it avoids over-fitting by limiting its likelihood over the BoN sample; and **(4)** it maintains overall response quality by reducing the win rate between its best and average samples.

4 BoN-Aware Fine-Tuning Using Reinforcement Learning

Training LLMs that are amenable to BoN sampling can be framed within the RL paradigm, which trains an agent (LLM) that optimizes its actions within an environment. In this context, the LLM generates N responses (candidates actions) for a given prompt (contexts). A separate macro agent (verifier) selects the candidate deemed most suitable according to a predefined criterion (e.g., probability of success). This action is then deployed to the environment, yielding a reward (e.g., task completion). The key challenge in training this agent lies in achieving two objectives simultaneously: (i) Enhancing agent's exploration capabilities to generate diverse candidates that cover the space of potential solutions and align with the verifier's preferences; (ii) Maximizing the environment reward of the final response. Motivated by this observation, we utilize RL for BoN-aware

fine-tuning, enabling the development of more robust and adaptable LLMs. A schematic of the BoN-Aware RL framework is shown in Figure 2.

The BoN-Aware RL problem takes the following form:

$$\max_{\pi \in \Pi} J(\pi) := \mathbb{E}_{x \sim P, y \sim \pi_{\text{bon}}(\cdot|x;\pi,r,N,T)}[R(x,y)]. \tag{6}$$

We train the BoN policy π_{bon} (paramterized by π) to attain a high environment reward. Apart from enabling better exploration, using the environment reward $R(x,y)$ in BoN-RL allows the base policy to tolerate potential errors in the verifier $r(x,y)$. We first develop a general algorithm for solving the BoN-aware RL problem. We then study an important subclass which assumes a binary reward, a common feature of many reasoning problems (e.g., math, code).

We begin with deriving a gradient estimator to the objective in Equation (6). Exploiting the connection between the BoN policy and its energy-based policy counterpart in Equation (10), and using derivations analogous to those in Lemma 1, we compute the gradient of $J(\theta)$, which leads to a REINFORCE-style algorithm (Williams, 1992) (see Appendix A.3 for proof):

Lemma 2 (BoN-RL). *The gradient of Equation (6) w.r.t. LLM parameters $\theta \in \Theta$ of π is given by*

$$\nabla_\theta J(\theta) = \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{bon}(\cdot|x)} \left[\nabla_\theta \log \pi_\theta (x,y) \cdot (R(x,y) - b(x)) \right], \tag{7}$$

where $b(x) = \mathbb{E}_{y \sim \pi_{bon}(\cdot|x)}[R(x,y)]$ is a baseline for variance reduction (Schulman et al., 2015).

This formulation resembles the standard REINFORCE gradient with the main difference of drawing samples from the BoN policy (instead from the base policy π). This allows one to solve BoN-RL via actor-critic methods (Sutton et al., 2009a). In practice, one can replace $b(x)$ with a learned value baseline $b_\psi(x)$ parameterized by ψ, for which ψ is updated by gradient descent w.r.t. the critic value loss. While BoN-RL inherits the benefits of verifier alignment from BoN-SFT, and can be viewed as a reward-weighted variant of the popular STaR method (Zelikman et al., 2022), generally it can be rather sample inefficient (especially when N is large), as estimating both the value function $b(x)$ and the policy gradient in BoN-RL require samples from the BoN distribution. See Appendix C for a discussion on alleviation using BoN distillation (Sessa et al., 2024).

BoN-RL with Binary Reward and Verifier. While Lemma 2 provides a general method for BoN-aware RL, the policy gradient estimator in Equation (7) is sample ineffiecient for a general rewards and verifiers. However, many domains admit binary success/failure metrics (e.g., reasoning tasks, math, coding) which allow an efficient gradient estimator, obviating the need for value estimation. Specifically, with a binary reward known to the verifier, i.e., $R(x,y) = r(x,y) \in \{0,1\}$, Theorem 1 of Sessa et al. (2024) implies the following closed-form solution of the BoN policy π_{bon}:

$$\pi_{\text{bon}}(y|x) = \begin{cases} \pi(y|x) \cdot P_{\text{fail}}(x)^{N-1} & \text{if } R(x,y) = 0 \\ \frac{\pi(y|x)}{1 - P_{\text{fail}}(x)} \cdot \left(1 - P_{\text{fail}}(x)^N\right) & \text{if } R(x,y) = 1 \end{cases}, \tag{8}$$

where $P_{\text{fail}}(x) := \mathbb{E}_{y' \sim \pi(\cdot|x)} \left[\mathbf{1}_{R(x,y')=0} \right]$ is the fraction of problems on which the base policy π is incorrect. Under the binary assumption, π_{bon} is a weighted distribution of the base policy π, whose importance sampling ratio depends on the its failure probability $P_{\text{fail}}(x)$. Introducing this closed form of π_{bon} to Lemma 2, we obtain the following policy gradient (see Appendix A.4 for proof):

Lemma 3 (BoN-RLB). *Assume $R(x,y) \in \{0,1\}$. The gradient of Equation (6) w.r.t. LLM parameters $\theta \in \Theta$ of π is given by*

$$\mathbb{E}_{x \sim \mathcal{D}} \left[\mathbb{E}_{y \sim \pi_{bon}, R=1} \left[\nabla_\theta \log \pi_\theta(y|x) \right] \cdot g_N^+(P_{fail}(x)) - \mathbb{E}_{y \sim \pi_{bon}, R=0} \left[\nabla_\theta \log \pi_\theta(y|x) \right] \cdot g^-(P_{fail}(x)) \right],$$

where the positive and negative sample-dependent weights are given by

$$g_N^+(p) = \frac{N \cdot p^{N-1}}{1 - p^N}, \quad g^-(p) = \frac{N \cdot p}{1 - p}. \tag{9}$$

This result not only reveals an efficient policy gradient estimator for binary reward, but more importantly demonstrates how BoN-RLB balances positive and negative examples in its gradient update, weighted by factors $g^+(P_{\text{fail}}(x), N)$ and $g^-(P_{\text{fail}}(x))$, respectively. When $P_{\text{fail}}(x)$ is closer to 1 ("harder" problems), the weight $g^+(p, N)$ is significantly larger and is amplified by N. By contrast, though $g^-(p)$ also increases as p reaches 1, it is not dependent on N. Conversely, when P_{fail} is small ("easier" problems), $g^+(p, N) \leqslant p \leqslant g^-(p)$, and this inequality becomes more apparent as N increases. Intuitively, for harder problems, more focus is put on (fewer) correct solutions by sampling

from the current BoN policy, whereas for easier problem, more entropy is introduced, encouraging the base policy to make more exploratory attempts to solve the problem. This is in line with our earlier motivation for BoN-aware fine-tuning: as long as the base policy can produce correct solutions given an input, it need not devote most of its sampling budget N to generating correct answers; instead, it can use it generate more exploratory solutions.

Lemma 3 proposes a novel way to re-weigh BoN-RLB's training examples, which prioritizes harder examples by giving their positive samples exponentially more influence and aggressively redistributing log likelihood away from incorrect responses. This allows the model to "exploit" both positive and negative samples to enhance learning of the harder example. The significance of this asymmetric weighting scheme is that it infuses *implicit exploration* capabilities to the base policy. As Tajwar et al. (2024) observed, when the model reduces the likelihood of negative responses, it shifts that probability mass towards a mode of the learned policy – essentially reinforcing existing successful strategies (exploitation). However, if these high-likelihood regions later produce errors, the resulting negative gradient redistributes this mass again, pushing the model to explore other potential solutions. This iterative process of concentrating probability mass and subsequent redistribution through negative gradients drives a form of exploration, encouraging the model to sample from a diverse range of responses yet maintaining high-likelihood of generating correct solutions.

Positive-only Weighting. Although we have illustrated the benefits of an asymmetric weighting scheme in BoN-RLB for exploration, training with both positive and negative examples may be infeasible (e.g., in a data-limited online RL system that only records positive examples). To tackle this, we apply a change of measure to Lemma 3 with the BoN distribution to derive a policy gradient that only involves positive examples (see Appendix A.5 for proof):

Corollary 4 (BoN-RLB(P)). *Assume $R(x, y) \in \{0, 1\}$. The gradient of Equation (6) w.r.t. LLM parameters $\theta \in \Theta$ of π is given by $\mathbb{E}_{x\sim\mathcal{D}}[\mathbb{E}_{y\sim\pi_{bon},R=1}[\nabla_\theta \log \pi_\theta(y|x)] \cdot \overline{g}^+_N(P_{fail}(x))]$, where $\overline{g}^+_N(p) := \frac{N\cdot p^{N-1}\cdot(1-p)}{(1-p^N)}$.*

Notice that the weighting $\overline{g}^+(p, N)$ is monotonically increasing in $p \in [0, 1]$ and lies within $[0, 1]$ for any N. Using this gradient update, BoN-RLB(P) resembles a weighted version of BoN-STaR (see Remark C.3 in the appendix), where it clones positive examples generated by the current BoN policy and up-weights the more difficult ones, where $P_{\text{fail}}(x)$ is close to 1.

5 Experiments

In this section we address the following questions: (1) Can we quantify the relationship between the BoN number of samples N and temperature T, enabling joint optimization of these parameters? (2) Do inference-aware fine-tuning methods (SFT and RL) enhance the effectiveness of BoN sampling? (3) Do these improvements generalize across values of N and T?

5.1 Co-scaling Analysis of Sample Size N and Temperature T in BoN

In Figure 3, we outline the BoN and Pass@N[2] performance of a pre-trained Gemma 2B model on MATH over varying N and T. Pass@N consistently increases with higher K, as commonly observed (Brown et al., 2024). As illustrated in Figure 3, our analysis suggests that this relationship can be captured by a function of the following form: $Pass@N(T) \approx \exp(a(T)K^{-b(T)})$, where the parameters $a(T)$ and $b(T)$ are temperature-dependent and derived by fitting the model to data at a specific temperature T. Further analysis of this scaling behavior (detailed in Appendix D.1) indicates that there is a strong positive correlation between the optimal temperature and K, which aligns with the intuition that larger N benefits from broader exploration (higher T), while smaller N favors focused exploitation (constant T). This relationship is straightforward, as there are no verifier errors confounding the selection of best responses.

Our experiments demonstrate an intriguing relationship between BoN accuracy, T, and N. We find that lower temperatures generally yield better BoN accuracy. Furthermore, BoN accuracy generally decreases as N increases, but degrades more rapidly with higher temperatures.With larger N and T, the increased randomness in the base policy inherently generates more "bad" samples (with poor accuracy). This phenomenon suggests that the verifier is sensitive to noise and may mistakenly select random outputs generated at higher temperatures as the best responses due to misalignment

[2]Pass@N is standard terminology; here K and N both denote the number of samples generated.

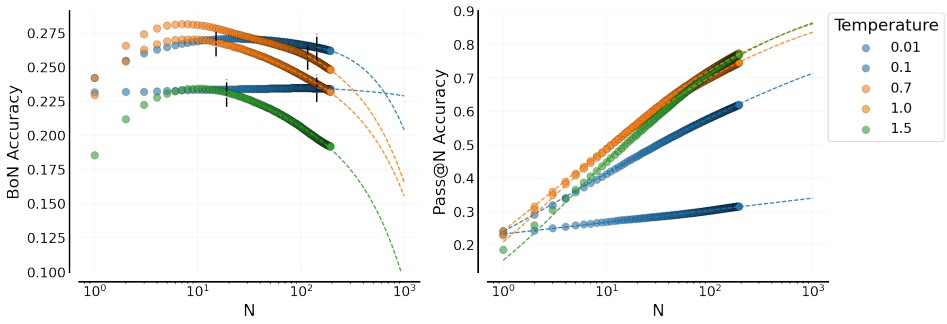


Figure 3: BoN and Pass@N performance of Gemma 2B policy and reward models w.r.t. varying N and T. Pass@N monotonically improves with N; BoN shows inflection points as N increases. Colored dashed lines denote predictions of scaling functions; black dashed lines in BoN plot denote the last inflection points.

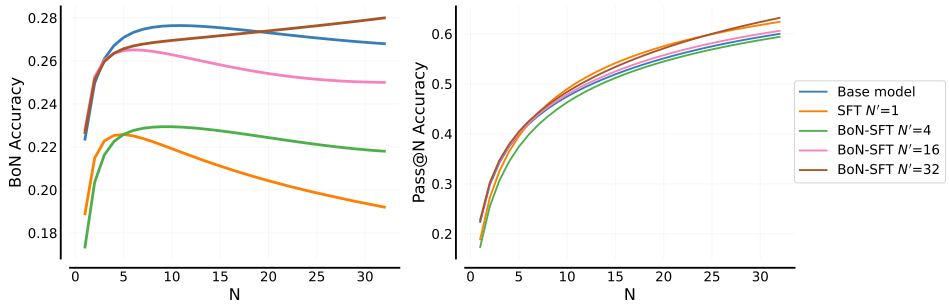


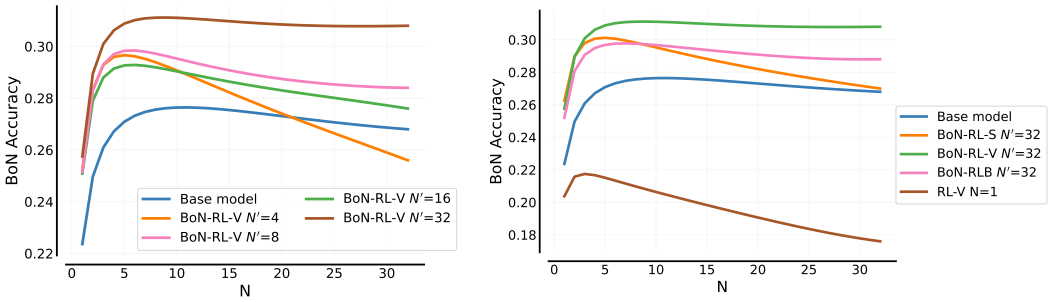
Figure 4: BoN Accuracy and Pass@N Accuracy for BoN-SFT Models.

with the true reward (Type II error). Conversely, at very low temperatures, BoN accuracy improves with N, indicating the algorithm remains in an exploitation phase. Optimal performance is observed at moderate N values, striking a balance between exploration and exploitation.

5.2 Inference-aware Fine-tuning with BoN

Experimental Setup We fine-tune Gemma 2B and 9B (Team et al., 2024) models and evaluate on the Hendrycks MATH benchmark (Hendrycks et al., 2021), (2) two held-out math benchmarks (Functional MATH (Srivastava et al., 2024) and MathOdyssey (Fang et al., 2024)), and (3) the HumanEval coding (Chen et al., 2021) benchmark. Our main evaluation metrics are the accuracy achieved by BoN sampling under a learned verifier (only for the MATH benchmarks in (1) and (2), for which the 2B and 9B models are trained to do point-wise correctness prediction of responses, and they have 69% and 76% accuracy on the MATH500 test set, respectively), and the Pass@N accuracy (i.e. BoN which directly uses the environment reward). To analyze the behavior of BoN sampling, we examine the relationship between N and T by running BoN inference on the evaluation sets with different (N, T) pairs, establishing scaling relationships that correlate these parameters. For BoN-aware fine-tuning, we test both SFT and RL.

Our experiments on BoN-aware LLM fine-tuning test a variety of methods including: (1) **BoN-SFT** from Lemma 1, (2) BoN-RL from Lemma 2 with a verifier, **BoN-RL-V**, (3) BoN-RL with environment reward as verifier, **BoN-RL-S**, (4) **BoN-RLB** from Lemma 3, and (5) **BoN-RLB(P)** from Corollary 4. These methods are designed to leverage the BoN selection strategy during model fine-tuning, under different settings mentioned in Sections 3 and 4. We compare these methods to several baselines: (1) STaR from Remark C.3, which uses self-training over correctly generated responses; (2) RL (Lee et al., 2023) with verifier feedback (RL-V); (3) RL with environment feedback (RL-S); (4) standard supervised finetuning of the base policy (SFT, $N' = 1$); and (5) BoN with the pre-trained model (Base model). This evaluation allows us to assess the effectiveness of our proposed methods relative to existing techniques. We denote by (N', T') and (N, T) the number of samples and temperature used in training and evaluation, respectively. In the following experiments, we set T' to 1 in training, and all evaluations (except where specified) are run with $T = 1$.



(a) BoN-RL-V with varying N' values.

(b) $N' = 32$ Results.

Figure 5: BoN accuracy on Various Training Runs of BoN-RL-V and Comparisons with Baselines

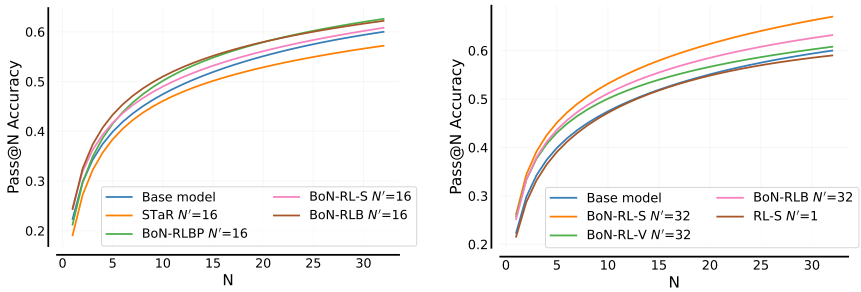
BoN-aware supervised fine-tuning. We first evaluate the BoN and Pass@N performance of offline SFT methods, including BoN-SFT with various N', and base SFT ($N' = 1$), with results shown in Figure 4. We find that base SFT significantly degrades upon the base model, indicating that it causes overfitting or lack of generalization. BoN-SFT is able to improve the BoN accuracy significantly, especially with increasing N'. We find that BoN-SFT with $N' = 32$ achieves the best performance for both BoN accuracy and Pass@N, suggesting that it is able to produce both high-quality and exploratory responses. To improve substantively over the base model, we next turn to RL, which should be more effective by virtue of being on-policy.

BoN-RL-V improves BoN Accuracy. We plot BoN accuracy evaluations in Figure 5 over different training runs of BoN-RL-V and other baselines. Our RL-BoN-V method significantly enhances BoN accuracy by effectively exploring a larger sample space during training (see Figure 5a). We observed peak performance when training with 32 samples ($N' = 32$), also leading to consistent improvements across all evaluation scenarios ($N = 1$ to $N = 32$). Notably, the gains were most pronounced at larger N values, indicating that RL-BoN-V not only excels in the specific BoN scenario it was trained on but also generalizes to other BoN configurations and even greatly improves the base policy's performance ($N = 1$) from 22% to 26%. This impressive performance can be attributed to the enhanced exploration capabilities of RL-BoN-V. Training with a large N' allows the base policy to explore a wider range of responses to generate higher-quality responses, similar to how effective exploration in RL leads to better overall performance, and better generalization across different (BoN inference) scenarios.

Our BoN RL algorithm, BoN-RL-V, with $N' = 32$, significantly outperforms several baselines (Figure 5b). Specifically, it boosts the Bo-32 accuracy of our base model from 26.8% to 30.8%. As expected, the inference-unaware RL-V method performs poorly, likely due to common reward hacking issues (Jinnai et al., 2024). While our other proposed methods, BoN-RL-S and BoN-RLB, show improvement over the base model, they still lag behind BoN-RL-V, indicating that training with a different verifier can boost performance, but not as much as training with the actual verifier used at test time. BoN-RL-V effectively learns to generate high-quality outputs that are easily recognized by the verifier, leading to superior performance. Interestingly, BoN-RLB demonstrates better generalization (w.r.t. BoN accuracy) to smaller N values despite being trained with $N' = 32$. This contrasts with the trend observed in Figure 6b, where BoN-RLB's Pass@N accuracy is worse than that of BoN-RL-S. This could be because BoN-RL-S generates more diverse and higher-quality samples, improving Pass@N, but may also be more susceptible to verifier errors.

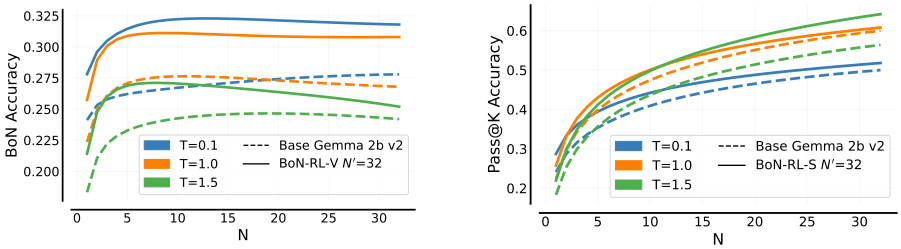
BoN-RL-S, BoN-RLB, and BoN-RLB(P) improve Pass@N. Our BoN-RL-S, BoN-RLB, and BoN-RLB(P) models demonstrate superior performance over the baseline STaR and RL-BoN-V methods in Pass@N evaluations, as shown in Figure 6 with $N' = 16$ (Figure 6a) and $N' = 32$ (Figure 6b). The methods are explicitly trained to maximize BoN with binary environment reward (i.e. Pass@N), leading to better Pass@N scaling at test time. Notably BoN-RL-S improves Pass@32 from 60% (base Gemma 2B model) to 67%.

For $N' = 32$, RL with ground truth feedback performs the worst (59% Pass@32), which is unsurprising because it is trained to maximize Pass@1. Similarly, BoN-RL-V is trained to optimize BoN with a noisy verifier selection, and as a result does not generalize to Pass@N well. By contrast,



(a) $N' = 16$ results. (b) $N' = 32$ results.

Figure 6: Pass@N results comparing the accuracy of various RL-BoN variants that are trained with binary environment reward as verifier with baselines and the general RL-BoN method.



(a) BoN Accuracy (b) Pass@N

Figure 7: BoN and Pass@N over non-training temperatures. BoN-RL with verifier and exact reward (solid lines) are trained with fixed temperature 1.0. Dashed lines show the base model (Gemma 2B), and solid lines show the finetuned model.

BoN-RL-S (67% Pass@32), BoN-RLB (63% Pass@32) methods focus directly on the Pass@N metric, thus leading to more efficient optimization and improved performance.

For $N' = 16$, STaR performs the worst (55% in Pass@32) as it fails to (i) utilize negative samples in training for implicit exploration (unlike BoN-RLB, 60% Pass@32), (ii) re-weight samples based on difficulty, prioritizing learning from challenging problems and avoiding overfitting to simpler ones (unlike BoN-RLB(P), 60% Pass@32), and (iii) account for the importance sampling factor between the base policy and the BoN policy (unlike BoN-RL-S, 58% Pass@32).

BoN-RL-S vs. BoN-RLB(P). Among the three proposed BoN-RL methods that optimize binary environment reward, we observe that their performances differ depending on N'. BoN-RLB and BoN-RLB(P) are superior to BoN-RL-S on $N' = 16$ (Figure 6a), but worse on $N' = 32$ (Figure 6b), suggesting that they suffer from instability with increasing N'. This is potentially due to the following observations: (i) The asymmetry between the positive (g^+) and negative (g^-) weights in BoN-RLB increases with N', destabilizing its learning at larger N' values; (ii) RL-BoN-S utilizes the variational approximation of π_{bon} in its gradient update, introducing approximation errors that may cause its sub-optimal performance (relative to a stable instance of BoN-RLB trained at $N' = 16$); BoN-RLB(P) only uses positive samples, which inherits the shortcomings of STaR (lack of implicit exploration), yet it re-balances the examples with the difficulty of the problems. Overall, it leads to consistent yet mild performance degradation over BoN-RLB.

Does BoN-aware fine-tuning generalize to different temperatures? Figure 7 shows that BoN-RL-V models demonstrate superior performance over the base model across various evaluation temperatures $T \in \{0.1, 1.0, 1.5\}$, despite being trained only with $T' = 1.0$, indicating that our BoN-aware RL method *generalizes* to sampling configurations outside of its training distribution. For Pass@N, BoN-RL-V outperforms the base model at each evaluation temperature, with wider performance gaps as temperature increases. This observation aligns with our co-scaling behavior studies, indicating the continued benefit of broader exploration even after BoN-aware RL training. Conversely, for BoN-accuracy, lower temperatures favor both models, but BoN-RL-V consistently outperforms the base model, with the performance gap widening at lower temperatures. This further

underscores the generalizability of BoN-RL-V across different exploration-exploitation regimes. Notably, BoN-RL-V demonstrates greater resilience to accuracy degradation at higher temperatures, leading to a milder/negligible BoN accuracy degradation as the evaluation sample size N increases. This also suggests BoN-RL-V's enhanced ability to adapt to verifier failure modes and mitigate Type-II errors stemming from misalignment with the environment reward.

6 Related Work

Large language models (LLMs) can leverage inference-time computation to improve the quality of their generated outputs (Welleck et al., 2024), particularly on reasoning tasks. One common approach is to use chain-of-thought (Wei et al., 2022), where the model generates a step-by-step rationale before generating the final output. Another useful approach that can be combined with chain-of-thought is Best-of-N rejection sampling (Charniak & Johnson, 2005; Stiennon et al., 2020), which is our focus in this work. In Best-of-N, we generate multiple candidate outputs from an LLM and select the best output. BoN re-ranking can be done either using oracle information, such as checking final answers for solving math problems, which is also known as Pass@N (Chen et al., 2021), or learned verifiers (Cobbe et al., 2021; Lightman et al., 2023; Hosseini et al., 2024; Zhang et al., 2024). Recent work also empirically analyzes strategies that optimally trade off additional test-time compute for improved performance (Wu et al., 2024; Snell et al., 2024).

Closely related to our approach is prior work that fine-tunes LLMs to improve their self-correction capabilities (Kumar et al., 2024; Snell et al., 2024) or search capabilities on planning tasks (Gandhi et al., 2024; Lehnert et al., 2024), which allows for more efficient scaling with test-time compute. By contrast, our work focuses on inference-aware fine-tuning that directly optimizes for Best-of-N performance, instead of an intermediate capability that be used at test-time.

To make an LLM amenable to test-time scaling, techniques like STaR (Zelikman et al., 2022) or ReST$^{\text{EM}}$ (Singh et al., 2023) have been employed to fine-tune the model using on-policy data. This process leverages BoN sampling to iteratively generate better responses, and fine-tunes on this curated data, for which the LLM learns to improve its proposal distribution, effectively increasing the likelihood of generating high-quality outputs during inference.

Finally, our work is related to recent work on leveraging tree search to enhance decision-making in reinforcement learning (Dalal et al., 2021). A key challenge in both BoN sampling and tree search lies in mitigating the impact of imperfect value estimation. Dalal et al. (2021) address this in tree search by penalizing actions leading to states with high Q-value error, effectively making inference more pessimistic for out-of-distribution samples. In contrast, in this work we tackle verifier error in BoN not by altering inference, but rather by incorporating it directly into training. Our BoN-aware methods learn to generate responses robust to these errors, aligning training with BoN inference. Furthermore, our BoN framework generalizes conceptually to tree search, with the verifier acting as an approximate Q-function, and training optimizing policy robustness to its errors.

7 Conclusion

We introduced inference-aware fine-tuning, a novel paradigm that bridges the gap between training and inference for LLMs. Specifically for the Best-of-N inference strategy, we discovered a co-scaling law for BoN that guides the optimization of temperature and sample size, developed a gamut of fine-tuning algorithms that handle various imitation learning and reinforcement learning settings, training LLMs to generate diverse and high-quality outputs tailored for BoN inference, demonstrated the efficacy of these methods by significantly improving on BoN accuracy and Pass@N on the standard MATH reasoning benchmark over state-of-the-art baselines, highlighting the robustness and generalizability of our approaches across various BoN configurations.

Our work exemplified how BoN-aware fine-tuning learns a meta-strategy, which interleaves best responses with more diverse responses that might be better suited for BoN sampling. These findings underscore the potential of inference-aware fine-tuning to unlock previously undiscovered capabilities in LLMs through aligning training methodologies with inference-time compute strategies. Future work includes extending this framework to incorporate more complex, inference algorithms (e.g., reasoning, critique-and-revise, MCTS), developing contextual BoN-aware algorithms that can generalize to various tasks, investigating the interplay between the co-scaling of temperature, sample size, and BoN-aware fine-tuning, and applying our algorithms to more larger-scale problems.

Reproducibility Statement. We utilize the publicly available Gemma 2B and 9B language models, the Hendrycks MATH benchmark, and the HumanEval coding benchmark – all accessible to the research community. Our experimental setup is described in detail in Section 5. Furthermore, the appendix provides comprehensive pseudo-code (Algorithms 1 to 4) and implementation details for our BoN-aware fine-tuning algorithms (BoN-SFT, BoN-RL, BoN-RLB, and BoN-RLB(P)). We also delve into the theoretical underpinnings of our methods in the main text and the appendix, enabling a thorough understanding of our approach.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pp. 4447–4455. PMLR, 2024.

Ahmad Beirami, Alekh Agarwal, Jonathan Berant, Alexander D'Amour, Jacob Eisenstein, Chirag Nagpal, and Ananda Theertha Suresh. Theoretical guarantees on the best-of-n alignment policy. *arXiv preprint arXiv:2401.01879*, 2024.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024. URL https://arxiv.org/abs/2407.21787.

Miguel A Carreira-Perpinan and Geoffrey Hinton. On contrastive divergence learning. In *International workshop on artificial intelligence and statistics*, pp. 33–40. PMLR, 2005.

Shicong Cen, Jincheng Mei, Katayoon Goshvadi, Hanjun Dai, Tong Yang, Sherry Yang, Dale Schuurmans, Yuejie Chi, and Bo Dai. Value-incentivized preference optimization: A unified approach to online and offline rlhf. *arXiv preprint arXiv:2405.19320*, 2024.

Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 173–180, 2005.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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. *arXiv preprint arXiv:2110.14168*, 2021.

Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Differentiable ranking and sorting using optimal transport. *Advances in neural information processing systems*, 32, 2019.

Bo Dai, Niao He, Hanjun Dai, and Le Song. Provable bayesian inference via particle mirror descent. In *Artificial Intelligence and Statistics*, pp. 985–994. PMLR, 2016.

Gal Dalal, Assaf Hallak, Steven Dalton, Shie Mannor, Gal Chechik, et al. Improve agents without retraining: Parallel tree search with off-policy correction. *Advances in Neural Information Processing Systems*, 34:5518–5530, 2021.

Meng Fang, Xiangpeng Wan, Fei Lu, Fei Xing, and Kai Zou. Mathodyssey: Benchmarking mathematical problem-solving skills in large language models using odyssey math data. *arXiv preprint arXiv:2406.18321*, 2024.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.

Hans U Gerber, Elias SW Shiu, et al. *Option pricing by Esscher transforms*. HEC Ecole des hautes études commerciales, 1993.

Lin Gui, Cristina Gârbacea, and Victor Veitch. Bonbon alignment for large language models and the sweetness of best-of-n sampling. *arXiv preprint arXiv:2406.00832*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*, 2024.

Yuu Jinnai, Tetsuro Morimura, Kaito Ariu, and Kenshi Abe. Regularized best-of-n sampling to mitigate reward hacking for language model alignment. *arXiv preprint arXiv:2404.01054*, 2024.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

Lucas Lehnert, Sainbayar Sukhbaatar, Paul Mcvay, Michael Rabbat, and Yuandong Tian. Beyond a*: Better planning with transformers via search dynamics bootstrapping. *arXiv preprint arXiv:2402.14083*, 2024.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023.

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. *arXiv preprint arXiv:2305.18290*, 2023.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Pier Giuseppe Sessa, Robert Dadashi, Léonard Hussenot, Johan Ferret, Nino Vieillard, Alexandre Ramé, Bobak Shariari, Sarah Perrin, Abe Friesen, Geoffrey Cideron, et al. Bond: Aligning llms with best-of-n distillation. *arXiv preprint arXiv:2407.14622*, 2024.

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, et al. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*, 2023.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Saurabh Srivastava, Anto PV, Shashank Menon, Ajay Sukumar, Alan Philipose, Stevin Prince, Sooraj Thomas, et al. Functional benchmarks for robust evaluation of reasoning performance, and the reasoning gap. *arXiv preprint arXiv:2402.19450*, 2024.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Richard S. Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *International Conference on Machine Learning (ICML)*, 2009a.

Richard S. Sutton, Hamid Reza Maei, and Csaba Szepesvári. A convergent o(n) temporal-difference algorithm for off-policy learning with linear function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2009b.

Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. Preference Fine-Tuning of LLMs Should Leverage Suboptimal, On-Policy Data, 2024.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen

Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models. *arXiv preprint arXiv:2406.16838*, 2024.

Peter M Williams. Bayesian conditionalisation and the principle of minimum information. *The British Journal for the Philosophy of Science*, 31(2):131–144, 1980.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.

Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. Differentiable top-k with optimal transport. *Advances in Neural Information Processing Systems*, 33:20520–20531, 2020.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Arnold Zellner. Optimal information processing and bayes's theorem. *The American Statistician*, 42(4):278–280, 1988.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024.

Jun Zhu, Ning Chen, and Eric P Xing. Bayesian inference with posterior regularization and applications to infinite latent svms. *The Journal of Machine Learning Research*, 15(1):1799–1847, 2014.

A Theoretical Derivations

A.1 Variational Approximation of BoN

We assume that the verifier score $r(x, y)$ is unique for all x, y, and the base model π has a finite set of possible outcomes for each context (Beirami et al., 2024).

Proposition 5 (Theorem 2 in Gui et al. (2024))**.** *With negligible error, one may effectively approximate π_{bon} as the solution to the following optimization problem:*

$$\pi_{bon}(y|x) \in \underset{\mu(\cdot|x)\in\Delta_{\mathcal{Y}}}{\arg\max}\ \mathbb{E}_{y\sim\mu}\left[Q_\pi(x,y)\right] - \frac{1}{\lambda_N} KL\left(\mu||\pi\right)(x), \tag{10}$$

where $Q_\pi(x,y) = \mathbb{E}_{y'\sim\pi(\cdot|x)}\left[\mathbf{1}_{r(x,y)\geqslant r(x,y')}\right]$ is the expected win-rate *over π, and*

$$\frac{(\lambda_N - 1)\exp(\lambda_N + 1)}{\exp(\lambda_N - 1)} - \log\left(\frac{\exp\lambda_N - 1}{\lambda_N}\right) = \log N - \frac{N-1}{N}, \tag{11}$$

through λ_N scaling sub-linearly with BoN number of samples N.

We can show the optimal solution to Equation (10) has a closed form $\pi^*_{\text{bon}} \propto [\pi \cdot \exp(\lambda_N Q_\pi)](y|x)$. This can also be revealed by viewing Equation (10) as the variational form of Bayes' rule (Williams, 1980; Zellner, 1988; Zhu et al., 2014; Dai et al., 2016), whose optimal solution is the posterior. This implies π_{bon} can be represented by an exponential-twisting policy (Gerber et al., 1993) over base policy π with energy function $\lambda_N \cdot Q_\pi(y, x)$, partition function $Z_\pi(x) = \mathbb{E}_{\pi(y|x)}\left[\exp\left(\lambda_N \cdot Q_\pi(x,y)\right)\right]$, and an appropriate λ_N from Equation (11).

In this section we will provide proofs for the technical results in this paper.

A.2 Proof of Lemma 1

Therorem 2 of Gui et al. (2024) shows that, with negligible error, one may effectively approximate π_{bon} as the solution to the following optimization problem:

$$\pi_{\text{bon}}(y|x) \in \underset{\mu(\cdot|x)\in\Delta_{\mathcal{Y}}}{\arg\max}\ \mathbb{E}_{y\sim\mu}\left[Q_\pi(x,y)\right] - \frac{1}{\lambda_N} KL\left(\mu||\pi\right)(x), \tag{12}$$

where $Q_\pi(x,y) = \mathbb{E}_{y'\sim\pi(\cdot|x)}\left[\mathbf{1}_{r(x,y)\geqslant r(x,y')}\right]$ is the expected *win-rate* over π, and this yields the variational form $\pi_{\text{bon}} \propto [\pi \cdot \exp(\lambda_N Q_\pi)](y|x)$. Plugging the variational form of π_{bon} into (2) yields the learning problem for π:

$$\max_{\pi\in\Pi} \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\log \pi_{\text{bon}}(y|x)\right] := \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\underbrace{\log \pi(y|x)}_{\text{Likelihood}} + \underbrace{\lambda_N \cdot Q_\pi(x,y) - \log Z_\pi(x)}_{\text{Inference-Awareness}}\right], \tag{13}$$

Taking gradient of this objective function over $\theta \in \Theta$ implies

$$\begin{aligned}
&\nabla_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\log \pi_\theta(y|x) + \lambda_N \cdot Q_{\pi_\theta}(x,y) - \log Z_{\pi_\theta}(x)\right] \\
=&\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\nabla_\theta \log \pi_\theta(y|x)\right] + \lambda_N \cdot \nabla_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[Q_{\pi_\theta}(x,y)\right] - \nabla_\theta \mathbb{E}_{x\sim\mathcal{D}}\left[\log Z_{\pi_\theta}(x)\right] \\
=&\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\nabla_\theta \log \pi_\theta(y|x)\right] + \lambda_N \cdot \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathbb{E}_{y'\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(y'|x) \cdot \sigma\left(r(x,y) - r(x,y')\right)\right]\right] \\
&- \mathbb{E}_{x\sim\mathcal{D}}\left[\nabla_\theta \log \mathbb{E}_{\pi_\theta(y|x)}\left[\exp\left(\lambda_N \cdot Q_{\pi_\theta}(x,y)\right)\right]\right] \\
=&\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\nabla_\theta \log \pi_\theta(y|x)\right] + \lambda_N \cdot \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathbb{E}_{y'\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(y'|x) \cdot \sigma\left(r(x,y) - r(x,y')\right)\right]\right] \\
&- \mathbb{E}_{x\sim\mathcal{D}}\left[\frac{\mathbb{E}_{\pi_\theta(y|x)}\left[\nabla_\theta \log \pi_\theta(y|x) \cdot \exp\left(\lambda_N \cdot Q_{\pi_\theta}(x,y)\right)\right] + \mathbb{E}_{\pi_\theta(y|x)}\left[\nabla_\theta \exp\left(\lambda_N \cdot Q_{\pi_\theta}(x,y)\right)\right]}{\mathbb{E}_{\pi_\theta(y|x)}\left[\exp\left(\lambda_N \cdot Q_{\pi_\theta}(x,y)\right)\right]}\right].
\end{aligned}$$

This further implies that

$$\begin{aligned}
&\nabla_\theta\, \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\log \pi_\theta(y|x) + \lambda_N \cdot Q_{\pi_\theta}(x,y) - \log Z_{\pi_\theta}(x)\right] \\
=&\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\nabla_\theta f(x,y;\theta)\right] - \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{\pi_\theta(y|x)}\left[\frac{\exp\left(\lambda_N \cdot Q_{\pi_\theta}(x,y)\right)}{\mathbb{E}_{\pi_\theta(y|x)}\left[\exp\left(\lambda_N \cdot Q_{\pi_\theta}(x,y)\right)\right]} \cdot \nabla_\theta f(x,y;\theta)\right]\right],
\end{aligned}$$

through collecting terms from the above expression and recalling the definition of $\nabla_\theta f(x,y;\theta)$ as

$$\nabla_\theta f(x,y;\theta) := \nabla_\theta \log \pi_\theta(y|x) + \lambda_N \cdot \mathbb{E}_{y'\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(y'|x) \cdot \sigma\left(r(x,y) - r(x,y')\right)\right].$$

This further implies that

$$\begin{aligned}
&\nabla_\theta\, \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\log \pi_\theta(y|x) + \lambda_N \cdot Q_{\pi_\theta}(x,y) - \log Z_{\pi_\theta}(x)\right] \\
=&\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\nabla_\theta f(x,y;\theta)\right] - \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_{\text{bon}}}\left[\nabla_\theta f(x,y;\theta)\right],
\end{aligned}$$

completing the proof of this lemma.

A.3 Proof of Lemma 2

Recall the RL objective function

$$\max_{\pi \in \Pi} J(\pi) := \mathbb{E}_{x \sim P, y \sim \pi_{\text{bon}}(\cdot|x;\pi,r,N,T)}[R(x,y)]. \tag{14}$$

Applying the REINFORCE trick (Sutton et al., 2009b) to this problem over the BoN policy class and using the analgous argument from the proof of Lemma 1, we have the following expression for the policy gradient:

$$\begin{aligned}
&\mathbb{E}_{y \sim \pi_{\text{bon}}(\cdot|x), x \sim \mathcal{D}} \left[\nabla_\theta \log \pi_{\text{bon}}(y|x) \cdot R(x,y)\right] \\
=&\mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\text{bon}}(\cdot|x)} \left[\nabla_\theta f\left(x, y; \theta\right) \cdot R(x,y)\right] - \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\text{bon}}} \left[\nabla_\theta f\left(x, y; \theta\right)\right] \cdot \mathbb{E}_{y \sim \pi_{\text{bon}}(\cdot|x), x \sim \mathcal{D}} \left[R(x,y)\right] \\
=&\mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\text{bon}}(\cdot|x)} \left[\nabla_\theta f_\theta\left(x, y\right) \cdot \left(R(x,y) - b(x)\right)\right] \\
=&\mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\text{bon}}(\cdot|x)} \left[\nabla_\theta \log \pi_\theta\left(y|x\right) \cdot \left(R(x,y) - b(x)\right)\right],
\end{aligned}$$

the last equality is due to the fact that $y \sim \pi_{\text{bon}}$ will always has a win-rate of 1, i.e., $Q_{\pi_\theta}(x,y) = 1$ almost surely, for $y \sim \pi_{\text{bon}}(\cdot|x)$. This completes the proof of this lemma.

A.4 Proof of Lemma 3

Using the log-likelihood trick, and plugging in the BoN distribution from Equation (8), the gradient of Equation (6) can be computed as

$$\begin{aligned}
&\mathbb{E}_{y \sim \pi_{\text{bon}}(\cdot|x), x \sim \mathcal{D}} \left[\nabla_\theta \log \pi_{\text{bon}}(y|x) \cdot R(x,y)\right] = \mathbb{E}_{y \sim \pi_{\text{bon}}(\cdot|x), R(x,y)=1, x \sim \mathcal{D}} \left[\nabla_\theta \log \pi_{\text{bon}}(y|x)\right] \\
=&\mathbb{E}_{x \sim \mathcal{D}}\Bigg[\mathbb{E}_{y \sim \pi_{\text{bon}}(\cdot|x), R(x,y)=1,} \left[\nabla_\theta \log \pi_\theta(y|x)\right] + (1 - \mathbb{E}_{y' \sim \pi(\cdot|x)} \left[\mathbf{1}_{R(x,y')=0}\right]^N) \nabla_\theta \log \frac{1 - \mathbb{E}_{y' \sim \pi(\cdot|x)} \left[\mathbf{1}_{R(x,y')=0}\right]^N}{1 - \mathbb{E}_{y' \sim \pi(\cdot|x)} \left[\mathbf{1}_{R(x,y')=0}\right]}\Bigg] \\
=&\mathbb{E}_{x \sim \mathcal{D}}\Bigg[\mathbb{E}_{y \sim \pi_{\text{bon}}(\cdot|x), R(x,y)=1} \left[\nabla_\theta \log \pi_\theta(y|x)\right] - \frac{1 - \mathbb{E}_{y' \sim \pi(\cdot|x)} \left[\mathbf{1}_{R(x,y')=0}\right]^N}{1 - \mathbb{E}_{y' \sim \pi(\cdot|x)} \left[\mathbf{1}_{R(x,y')=0}\right]} \mathbb{E}_{y' \sim \pi} \left[\nabla_\theta \log \pi_\theta(y'|x) \cdot \mathbf{1}_{R(x,y')=1}\right] \\
&\quad + \mathbb{E}_{y' \sim \pi} \left[\nabla_\theta \log \pi_\theta(y'|x) \cdot \mathbf{1}_{R(x,y')=1}\right] \cdot N \cdot \mathbb{E}_{y' \sim \pi(\cdot|x)} \left[\mathbf{1}_{R(x,y')=0}\right]^{N-1}\Bigg] \\
=&\mathbb{E}_{x \sim \mathcal{D}}\left[\mathbb{E}_{y \sim \pi_{\text{bon}}(\cdot|x), R(x,y)=1} \left[\nabla_\theta \log \pi_\theta(y|x)\right] \cdot \frac{N I_{\text{ref}}(x)^{N-1}(1 - I_{\text{ref}}(x))}{1 - I_{\text{ref}}(x)^N}\right]. \\
=&\mathbb{E}_{x \sim \mathcal{D}}\Bigg[\mathbb{E}_{y \sim \pi_{\text{bon}}(\cdot|x), R(x,y)=1} \left[\nabla_\theta \log \pi_\theta(y|x)\right] \cdot \frac{N \cdot I_{\text{ref}}(x)^{N-1}}{1 - I_{\text{ref}}(x)^N} \\
&\qquad - \mathbb{E}_{y \sim \pi_{\text{bon}}(\cdot|x), R(x,y)=0} \left[\nabla_\theta \log \pi_\theta(y|x)\right] \cdot \frac{N \cdot I_{\text{ref}}(x)}{1 - I_{\text{ref}}(x)}\Bigg]
\end{aligned}$$

A.5 Proof of Corollary 4

Using the log-likelihood trick, and plugging in the BoN distribution from Equation (8), the gradient of problem Equation (6) can be computed as

$$\begin{aligned}
&\mathbb{E}_{x \sim \mathcal{D}}\Bigg[\mathbb{E}_{y \sim \pi_{\text{bon}}(\cdot|x), R(x,y)=1} \left[\nabla_\theta \log \pi_\theta(y|x)\right] \cdot \frac{N \cdot I_{\text{ref}}(x)^{N-1}}{1 - I_{\text{ref}}(x)^N} \\
&\qquad - \mathbb{E}_{y \sim \pi_{\text{bon}}(\cdot|x), R(x,y)=0} \left[\nabla_\theta \log \pi_\theta(y|x)\right] \cdot \frac{N \cdot I_{\text{ref}}(x)}{1 - I_{\text{ref}}(x)}\Bigg] \\
=&\mathbb{E}_{x \sim \mathcal{D}}\left[\mathbb{E}_{y \sim \pi_{\text{bon}}(\cdot|x), R(x,y)=1} \left[\nabla_\theta \log \pi_\theta(y|x)\right] \cdot \frac{N I_{\text{ref}}(x)^{N-1}(1 - I_{\text{ref}}(x))}{1 - I_{\text{ref}}(x)^N}\right].
\end{aligned}$$

Algorithm 1 BoN-SFT

Input: Verifier score r, environment reward R, expert dataset $\mathcal{D}$
for $t = 1, 2, \ldots$ **do**
  Sample a batch of prompts and solutions $\{x_i, y_i\}_{i=1}^{B}$ from the expert data $\mathcal{D}$.
  for $i = 1, \ldots, B$ **do**
    Sample N responses $\{y_{i,j}\}_{j=1}^{N}$ from $\pi_\theta(\cdot|x_i)$.
    Select the BoN response $y_i^* = \arg\max_j r(x_i, y_{i,j})$.
    Compute the gradient $\nabla_\theta f_\theta(x_i, y_i)$ using Equation (5).
  end for
  Update θ by following the gradient in Theorem 1 at learning rate $\alpha > 0$, i.e.,

$$\theta \leftarrow \theta + \alpha \left(\frac{1}{N} \sum_{i=1}^{N} \left[\nabla_\theta f\left(x_i, y_i; \theta\right)\right] - \left[\nabla_\theta f\left(x_i, y_i^*; \theta\right)\right] \right)$$

end for

B Pseudo-code and Implementation Details

Pseudo-code for all our SFT and RL methods is presented in Algorithms 1 to 4. Our implementation follows the standard use of an anchor policy, updated using exponential moving average. The policy is trained via BoN-aware losses, with additional KL divergence loss to the anchor policy. Table 1 shows the hyper-parameters used for all of our experiments.

We use linear annealing for the KL-coefficient. For all our RL experiments, we use a value baselines to reduce variance of our reward estimates. We normalize our advantage estimates w.r.t. the batch. For BoN-RLB the value network estimates $P_{\text{fail}}(x)$. We add additional clipping of the coefficients g_N^+, g_N^- by clipping the value estimates for P_{fail}.

Table 1: Hyperparameters used in experiments.

Hyperparameter	**Value**
Base model	Gemma 2b v2
Optimizer	AdamW
Learning rate policy	3e-6
Policy warmup steps	100
Learning rate value	1e-5
Anchor EMA	0.01
Training steps	2500
Batch size	32
Sampling temperature	1.0
KL coefficient anneal steps	2500
KL coefficient anneal range	$1.0 \rightarrow 0.075$
KL coefficient anneal delay	10
Clipping values for P_{fail}	$\{0.01, 0.99\}$

B.1 Analysis of BoN-RLB Weights

Recall the BoN-RLB weights of Lemma 3:

$$g_N^+(p) = \frac{N \cdot p^{N-1} \cdot (1-p)}{(1-p^N)^2}, \quad g^-(p) = \frac{p}{1-p}.$$

and $\overline{g}_N^+(p) := \frac{N \cdot p^{N-1} \cdot (1-p)}{(1-p^N)^2} - \frac{p^N}{1-p^N}$ of Corollary 4.

Figure 8 analyzes these weights, illustrating their behavior concerning the base policy's failure rate (p) and the number of samples (N). The left column shows how the weights evolve with increasing N for various fixed p, representing different difficulty levels. The right column shows how weights change with increasing failure probability p for fixed sample sizes N.

Algorithm 2 BoN-RLB(P)

Input: Environment reward R, dataset $\mathcal{D}$
for $t = 1, 2, \ldots$ **do**
  Sample a batch of prompts $\{x_i\}_{i=1}^{B}$ from $\mathcal{D}$.
  for $i = 1, \ldots, B$ **do**
    Sample N responses $\{y_{i,j}\}_{j=1}^{N}$ from $\pi_\theta(\cdot|x_i)$.
    Sample rewards for all candidate responses $\{R(x_i, y_i)\}_{i=1}^{N}$ from environment.
    Select the BoN response $y_i^* = \arg\max_j R(x_i, y_{i,j})$.
    Empirically estimate the base failure probability for each x_i, $i \in \{1, \ldots, B\}$,

$$\widehat{P}_{\text{fail}}(x_i) := \frac{1}{N} \sum_{j=1}^{N} \mathbf{1}_{R(x_i, y_{i,j})=0}.$$

  end for
  Update θ by following the gradient in Corollary 4 at learning rate $\alpha > 0$, i.e.,

$$\theta \leftarrow \theta + \alpha \left(\frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \log \pi_\theta(y_i^{*,+}|x_i) \cdot \overline{g}^{+}(P_{\text{fail}}(x_i), N) \right)$$

  where $y_i^{*,+}$ represents the BoN sample that achieves a reward of 1.
end for

Algorithm 3 BoN-RLB

Input: Environment reward R, dataset $\mathcal{D}$
for $t = 1, 2, \ldots$ **do**
  Sample a batch of prompts $\{x_i\}_{i=1}^{B}$ from $\mathcal{D}$.
  for $i = 1, \ldots, B$ **do**
    Sample N responses $\{y_{i,j}\}_{j=1}^{N}$ from $\pi_\theta(\cdot|x_i)$.
    Sample rewards for all candidate responses $\{R(x_i, y_i)\}_{i=1}^{N}$ from environment.
    Select the BoN response $y_i^* = \arg\max_j R(x_i, y_{i,j})$.
    Empirically estimate the base failure probability for each x_i, $i \in \{1, \ldots, B\}$,

$$\widehat{P}_{\text{fail}}(x_i) := \frac{1}{N} \sum_{j=1}^{N} \mathbf{1}_{R(x_i, y_{i,j})=0}.$$

  end for
  Update θ by following the gradient in Lemma 3 at learning rate $\alpha > 0$, i.e.,

$$\theta \leftarrow \theta + \alpha \left(\frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \log \pi_\theta(y_i^{*,+}|x_i) \cdot g^{+}(P_{\text{fail}}(x_i), N) - \nabla_\theta \log \pi_\theta(y^{*,-}|x_i) \cdot g^{-}(P_{\text{fail}}(x_i)) \right)$$

  where $y_i^{*,+}$, $y_i^{*,-}$ represent the BoN samples that achieve rewards of 1 and 0 respectively.
end for

A key observation from the left column is the intersection point of $g_N^+(p)$ and $g^-(p)$. The shifting balance between $g_N^+(p)$ and $g^-(p)$ with varying p directly reflects the exploration-exploitation trade-off. On the right column we see how the weights evolve as the failure probability (p) increases for fixed sample sizes. As p approaches 1, signifying very difficult problems, both $g_N^+(p)$ and $g^-(p)$ increase, but $g_N^+(p)$ rises more dramatically, especially for larger values of N. This sharp increase in $g_N^+(p)$ highlights the algorithm's increasing emphasis on learning from the few correct responses that are available in challenging scenarios. The effect is amplified by larger sample sizes: the more attempts are made, the more valuable the scarce successes become. The $\overline{g}_N^+(p)$ weight, used when only positive feedback is available, exhibits a similar upward trend with p but with a less pronounced increase. This more moderate behavior can be attributed to the subtraction term in its formula, which tempers the influence of the positive samples and promotes a more balanced learning approach.

Finally, the potentially very large values of $g_N^+(p)$ for hard problems and larger N introduce challenges for estimation. These high weights amplify the impact of individual positive samples, making the training more vulnerable to noise and potentially hindering convergence to a stable optimal policy. This underscores the need for techniques like gradient clipping or regularization to mitigate the

Algorithm 4 BoN-RL

Input: Verifier score r, environment reward R, dataset $\mathcal{D}$
for $t = 1, 2, \ldots$ **do**
  Sample a batch of prompts $\{x_i\}_{i=1}^{B}$ from $\mathcal{D}$.
  for $i = 1, \ldots, B$ **do**
    Sample N responses $\{y_{i,j}\}_{j=1}^{N}$ from $\pi_\theta(\cdot|x_i)$.
    Select the BoN response $y_i^* = \arg\max_j r(x_i, y_{i,j})$.
    (If environment reward R is available to the BoN algorithm, we replace verifier r with that.)
    Sample the reward $R(x_i, y_i^*)$ from environment.
    Compute the gradient $\nabla_\theta f_\theta(x_i, y_i)$ using Equation (5).
  end for
  Update θ by following the gradient in Lemma 2 at learning rate $\alpha > 0$, i.e.,

$$\theta \leftarrow \theta + \alpha \left(\frac{1}{B} \sum_{i=1}^{B} \nabla_\theta f_\theta \left(x_i, y_i^*\right) \cdot \left(R(x_i, y_i^*) - b(x_i)\right) \right)$$

  where $b_\psi(x_i), i = 1, \ldots, B$ is a learned baseline value function of π_{bon}, i.e.,

$$\psi^* \in \arg\min_{\psi} \frac{1}{B} \sum_{i=1}^{B} [R(x_i, y_i^*) - b_\psi(x_i)]^2$$

  Update value estimate ψ using the current environment reward target and BoN policy trajectories.
end for

destabilizing effects of high weight values and ensure robust learning, or alternatively, using $\overline{g}_N^+(p)$ with Corollary 4 to ensure boundness of the gradient weights.

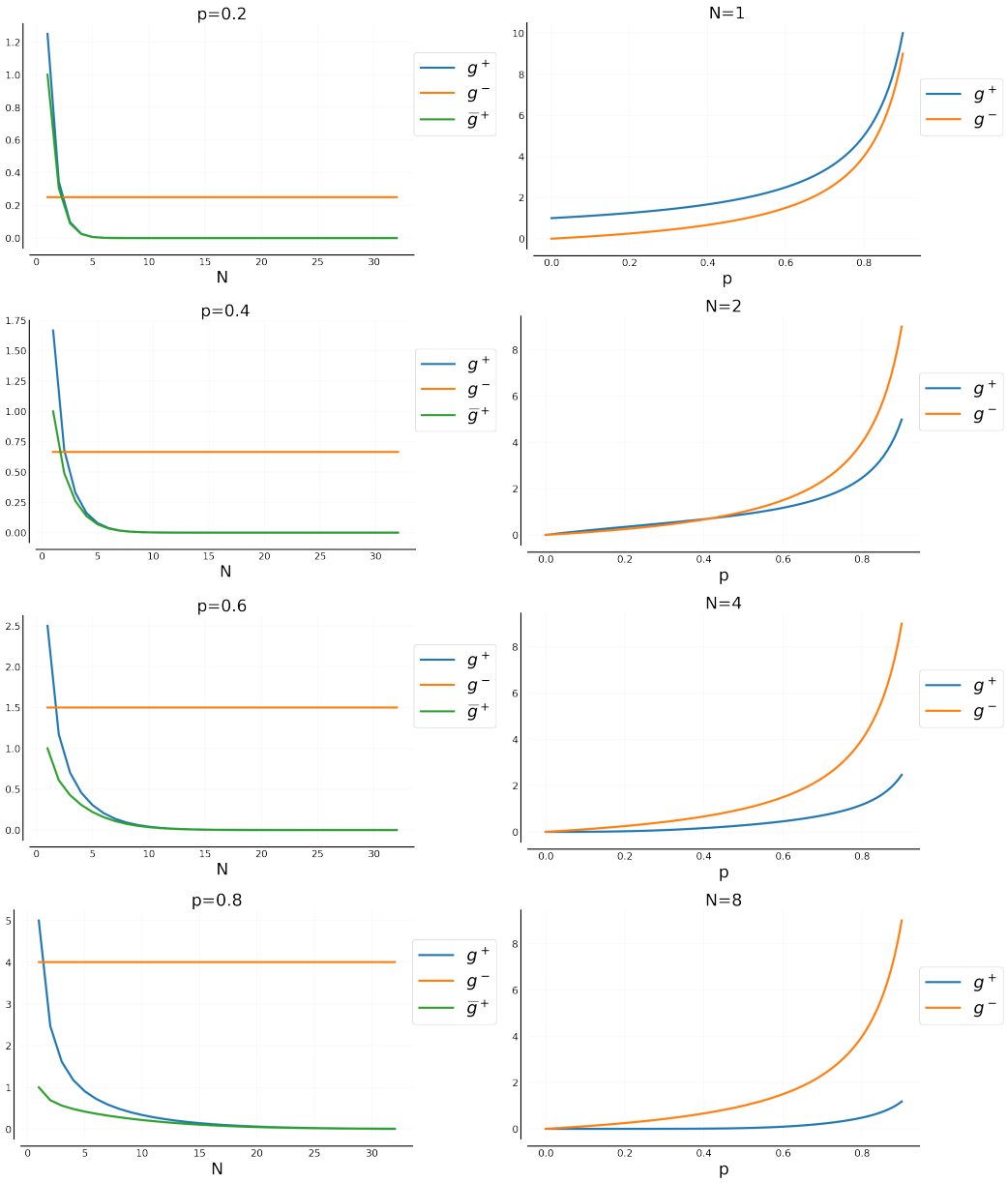


Figure 8: BoN-RLB weights $g_N^+(p)$, $g^-(p)$, and $\overline{g}_N^+(p)$ as functions of p (failure probability) and N (number of samples). The left column shows the behavior of the weights with respect to N for fixed values of p. The right column shows the behavior of the weights with respect to p for fixed values of N.

C ALGORITHMIC EXTENSIONS

C.1 ENTROPY-REGULARIZED RL

We would like to study an entropy-regularized RL problem for the π_{bon} policy. Recall that generally in entropy-regularized RL, we solve

$$\max_{\pi(\cdot|x)\in\Delta} \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{y\sim\pi(\cdot|x)}\left[R(x,y)\right] - \beta \cdot KL(\pi||\pi_\beta)(x)\right], \tag{15}$$

where $R(x,y)$ is the environment reward (that is not necessarily identical to the verifier score model), π_β is a baseline policy, and $\beta > 0$ is the weight for the KL regularization term. Using the consistency condition of KL-regularized MDP, solving for the optimal policy of this problem is equivalent to finding a solution pair of V and $\pi \in \Delta$ of the following equation:

$$V(x) = R(x,y) + \beta \log \pi_\beta(y|x) - \beta \log \pi(y|x),\ \forall x \in \mathcal{D},\ \forall y \tag{16}$$

Now, we further parameterize the policy variable π with the BoN policy π_{bon}, then with the sufficiency part of the consistency condition one can show that π_{bon} is an optimal RL policy of Equation (15) if there exists a pair of V and π that satisfies the following equation

$$V(x) = R(x,y) + \beta \log \pi_\beta(y|x) - \beta\left(\log \pi(y|x) + \lambda_N \cdot Q_\pi\left(y,x\right) - \log Z_\pi(x)\right),\ \forall x \in \mathcal{D},\ \forall y \tag{17}$$

There are two ways to approximately find the solution in Equation (17). The first way is to reformulate the above equation with a condition that equates the values between any pairwise states and outputs (x,y,y'):

$$R(x,y') + \beta \log \frac{\pi_\beta(y'|x)}{\pi(y'|x)} + \beta\lambda_N \cdot Q_\pi\left(y',x\right) = R(x,y) + \beta \log \frac{\pi_\beta(y|x)}{\pi(y|x)} + \beta\lambda_N \cdot Q_\pi\left(y,x\right),\ \forall x \in \mathcal{D},\ \forall y, y'. \tag{18}$$

Suppose one have access to pairwise labels in the data-set, then this formulation eliminates any terms that are independent to y and circumvents the need of solving for the value function V. One may approximately solve Equation (18) by minimizing the following ℓ_2 loss:

$$\min_{\pi\in\Delta} \mathbb{E}_{(x,y,y')\in\mathcal{D}}\left[\left(g(x,y;\pi) - \left(g(x,y';\pi)\right)^2\right]\right],$$

$$g(x,y;\pi) := R(x,y) + \beta \log \frac{\pi_\beta(y|x)}{\pi(y|x)} + \beta\lambda_N \cdot Q_\pi\left(y,x\right).$$

This formulation is similar to that in IPO (Azar et al., 2024). However, unlike IPO, where the term $g(x,y;\pi)$ is linear in the logits of π and therefore one can show that its ℓ_2 minimization problem has a unique solution, in this case $g(x,y;\pi)$ also depends on Q_π, which is a function of π (and thus a nonlinear function of its logits), preventing us from drawing similar conclusions that the ℓ_2 minimization problem has a unique solution. Therefore, even if one can exactly solve this ℓ_2 minimization problem (and make the loss zero), there is no guarantee that the solution policy π^* corresponds to the base policy of an optimal π_{bon} policy to the KL-regularized RL problem.

For the second approach, consider the following linear programming reformulation of Equation (17):

$$\min_{V,\pi\in\Delta} \mathbb{E}_{x\in\mathcal{D}}[V(x)]$$
$$\text{s.t. } V(x) \geqslant R(x,y) + \beta \log \pi_\beta(y|x) - \beta\left(\log \pi(y|x) + \lambda_N \cdot Q_\pi\left(y,x\right) - \log Z_\pi(x)\right),\ \forall x \in \mathcal{D},\ \forall y \tag{19}$$

Since the inequality constraint is a convex function in π and an affine function in V, by strong duality it has the following equivalent Lagrangian-dual formulation:

$$\begin{aligned}
&\max_{\kappa(\cdot,\cdot)\geqslant 0} \min_{V,\pi\in\Delta} \mathbb{E}_{(x,y)\in\mathcal{D}}\left[V(x) + \kappa(x,y)\cdot\left(R(x,y) + \beta\frac{\pi_\beta(y|x)}{\pi(y|x)} - \beta\left(\lambda_N \cdot Q_\pi\left(y,x\right) - \log Z_\pi(x) - V(x)\right)\right)\right] \\
&= \max_{\kappa(\cdot,\cdot)\geqslant 0} \min_{V} \mathbb{E}_{\mathcal{D}}\left[\left(1-\kappa(x,y)\right)\cdot V(x) + \kappa(x,y)\cdot\left(R(x,y) + \beta\cdot\pi_\beta(y|x)\right)\right] - \max_{\pi\in\Delta} \mathbb{E}_{\mathcal{D}}\left[\kappa(x,y)\cdot\log\pi_{\text{bon}}(y|x;\pi)\right]
\end{aligned} \tag{20}$$

This formulation can be viewed as an weighted-SFT approach that iteratively updates (i) the base policy π that maximizes the likelihood of π_{bon} over data $\mathcal{D}$, weighted with importance weights $\kappa(x,y)$, and (ii) the importance weight function κ itself. Here, the value function $V(x)$ is simply an auxiliary variable.

	Pass@N	BoN	MajorityVoting
Gemma-9B	0.986	0.989	0.89
Gemma-2B	0.998	0.998	0.784

Table 2: R-squared values for different language models and inference algorithms.

C.2 Improved Efficiency with BoN Distillation

While Lemma 1 provides a recipe for training a base policy to adapt to the BoN inference strategy, a key challenge lies in the computational cost and data inefficiency associated with BoN sampling, especially when N is large. Particularly, each gradient update requires generating N samples from the current base policy, which can be prohibitively expensive. Furthermore, using these samples solely for a single gradient update may deem wasteful.

To alleviate this issue, leveraging the recent advances in BoN Distillation (BoND) (Sessa et al., 2024), an RLHF algorithm that distills BoN behaviors into a standard LLM, we approximate the BoN distribution of the current π. This results in an iterative, two-step procedure. First, we estimate a BoND policy π_{BoND} (parameterized by weights ϕ) of π by solving the distribution-matching problem: $\min_\phi \mathbb{E}_{x\sim\mathcal{D}}[\text{KL}(\pi_\phi||\pi_{\text{bon}})(x)]$, where the backward-KL metric induces quantile-based advantage and mode-seeking behaviors to π_{BoND}. Utilizing the variational form $\pi_{\text{bon}}(y|x) \propto \pi \cdot \exp\left(\lambda_N Q_\pi\right)(y|x)$, this problem can be further reformulated as

$$\pi_{\text{BoND}}(y|x) \in \arg\max_\phi \ \mathbb{E}_{x\sim\mathcal{D}}[\mathbb{E}_{y\sim\pi_\phi(\cdot|x)}[Q_\pi(y,x)] - \frac{1}{\lambda_N}\text{KL}(\pi_\phi||\pi)(x)]. \tag{21}$$

Second, equipped with the BoND policy, we change the gradient of Lemma 1 with the approximate gradient $\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\nabla_\theta f\left(x,y;\theta\right)\right] - \mathbb{E}_{x\sim\mathcal{D},y\sim\pi_{\text{BoND}}(\cdot|x)}\left[\nabla_\theta f\left(x,y;\theta\right)\right]$. In general, this approach is also well-connected with Contrastive Divergence (Carreira-Perpinan & Hinton, 2005) in energy-based learning, which promotes the idea of approximately sample from the current target distribution (π_{bon} in our case). It shows that the learning algorithm can still converge to an optimum w.r.t. the original objective function as long as the gradient estimated by the approximate samples still points at an ascending direction.

C.3 Connection to STaR

Consider the popular STaR method (Zelikman et al., 2022) applied for training π_{bon}, which updates θ by following the reward-weighted gradient:

$$\mathbb{E}_{x\sim\mathcal{D},y\sim\pi_{\text{bon}}(\cdot|x)}\left[\nabla_\theta \log \pi_\theta(y|x)\cdot R(x,y)\right]. \tag{22}$$

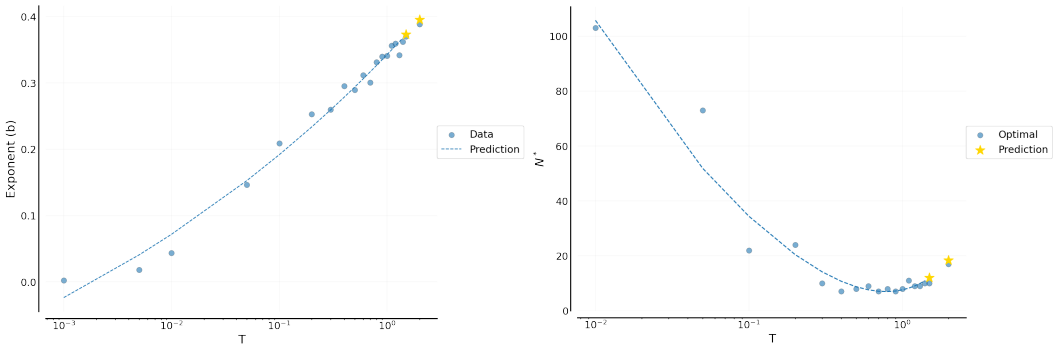
Notice that the policy gradient of BoN-RL is a sum of two terms: $\nabla_\theta J(\theta) = g_1(\theta) + g_2(\theta)$, where $g_1(\theta) = \mathbb{E}_{x\sim\mathcal{D},y\sim\pi_{\text{bon}}(\cdot|x)}\left[\nabla_\theta \log \pi_\theta(y|x)\cdot R(x,y)\right]$ is equivalent to that of BoN-STaR, updating π via weighted supervised fine-tuning over the responses and the rewards obtained by the current BoN policy, and $g_2(\theta) = \mathbb{E}_{x\sim\mathcal{D},y\sim\pi_{\text{bon}}(\cdot|x)}\left[\nabla_\theta(\lambda_N Q_\pi - \log\mathbb{E}_\pi\exp(\lambda_N Q_\pi))(x,y)\cdot R(x,y)\right]$ accounts for the gradient effect of the importance sampling term $(\exp\lambda_N Q_\pi/Z_\pi)(x,y)$ between π and π_{bon}, emphasizing on how much it can improve the reward. The additional $g_2(\theta)$ component makes BoN-RL amenable to the distributional shifts introduced by the BoN procedure, enabling the base policy to be adept at utilizing the BoN exploration mechanism to optimize the reward.

D Experimental Details

D.1 Additional Scaling Results with Gemma 9B verifier and policy models

Similar to Gemma2B co-scaling experiments, for Gemma 9B co-scaling, we present additional results in Figure 9. We analyze the optimal exponent $b^*(T)$ w.r.t different temperatures (see co-scaling in

Generalization of scaling predictions. In Table 2, we compare various inference algorithms and LLMs of different sizes. For MajorityVoting algorithm, we use MC estimation to simulate different sample sizes. We use the same functional form used for co-scaling experiments in **????**



(a) Pass@N (b) BoN

Figure 9: Scaling of exponent w.r.t temperature in Pass@N and optimal N w.r.t. temperature in BoN. Dashed curves denote in-training predictions, stars denote extrapolation values for the corresponding temperatures.

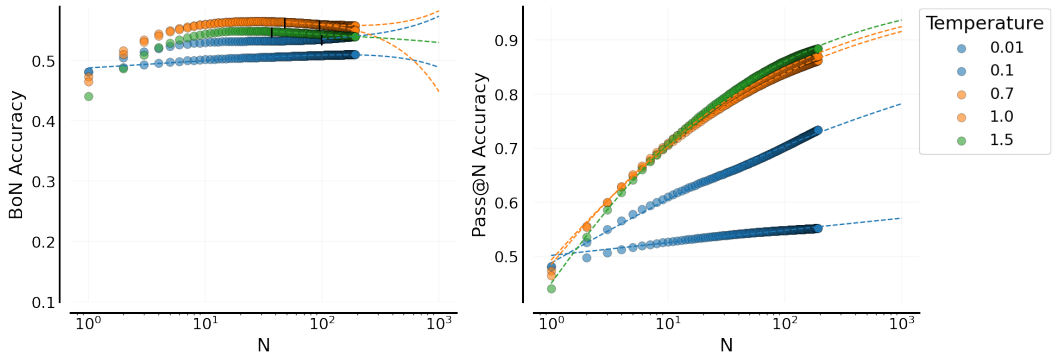


Figure 10: Pass@N (left) and BoN (right) performance for Gemma-9B. While curves show similar shape as Gemma-2B models, overall performance is globally improved and overoptimization is reduced.

Gemma-9B Results. In Figure 10, we present results for Gemma-9B policy and reward models. Using Gemma-9B improves both Pass@N and BoN significantly compared to Gemma-2B. We observe that the gap between using large temperatures (0.7 or 1.0) and very small temperatures (0.1) also increased. While Gemma-2B showed very strong reward model overoptimization for larger N and temperatures, we see a lesser overoptimization for Gemma-9B models.

D.2 Model Training Details

For the MATH benchmark, we trained the Gemma 2B and 9B models with the Hendrycks MATH dataset. Following Lightman et al. (2023), we augment the original 7500 MATH training problems with 4500 problems from the test set, evaluating performance on the remaining 500 problems. In the supervised setting, we leverage a larger Gemini 1.5 Flash model (Reid et al., 2024) to generate MATH solutions with answers and steps (32 candidates for each of the MATH problems), subsampling only the correct responses and distilling knowledge into the Gemma 2B model. In the RL setting, we use a binary environment reward denoting whether the model's answer matches the ground truth answer. The verifier used in all BoN experiments is a separate pre-trained Gemma 2B model that predicts the probability of a correct response given the prompt. The verifier is trained with the data collected from the Gemini 1.5 Flash model.

Alternatively, to benchmark our models on code generation, we train on MBPP (Austin et al., 2021) and evaluate on the HumanEval benchmark (Chen et al., 2021), following the standard procedures delineated in Kumar et al. (2024).

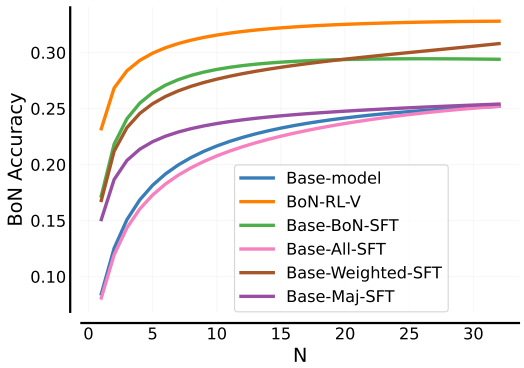


Figure 11: BoN Accuracy on MATH comparing Base Gemma 2B and BoN RL-V with other fine-tuning techniques: (1) BoN-SFT: Distillation of BoN sample for N=16; (2) All-SFT: Distillation of all N=16 samples (i.e., average sample); (3) Weighted-SFT: Distillation of all N=16 samples by average re-sampling w.r.t. verifier scores; and (4) Maj-SFT: Distillation of majority voting strategy.

D.3 Additional BoN-aware Fine-tuning Results

We now present additional results on BoN-aware fine-tuning (both SFT and RL).

D.3.1 Comparing BoN-aware Fine-tuning with Base-BoN Distillation Baselines

We consider various alternative methods to improve Gemma 2B BoN accuracy through various data generation methods. We distill the Gemma 2B model using these datasets and compare to the base Gemma 2B model and our BoN-RL-V method. We consider the following four distillation benchmarks (all run over Hendrycks MATH):

1. Base-BoN-SFT: In this method we generate a dataset of the best of $N = 16$ samples for each example in the dataset. We use the best sample as target to distill Gemma 2B.
2. Base-All-SFT: We use the full range of $N = 16$ samples as targets. This dataset is used to distill Gemma 2B to the average effective sample of the base model.
3. Base-Weighted-SFT: Similar to Base-All-SFT, we sample $N = 16$ samples for each example. We then re-sample $N = 16$ examples (from these samples, with repetition), weighted according to verifier scores. This dataset is used to distill Gemma 2B to the average effective sample, weighted by verifier scores.
4. Base-Maj-SFT: We use majority voting over $N = 16$ samples to select a target. We distill Gemma 2B to predict the majority voted target.

We show the BoN accuracy results of these methods in Figure 10. While the aforementioned baselines do improve BoN performance over the Base Gemma 2B model, they are still out-performed by our BoN-RL-V method, indicating the value of utilizing the inference BoN strategy explicitly during training.

D.3.2 Gemma 9B Hendrycks MATH

We additionally benchmark a larger model, GemmaV2 9B, on Hendrycks MATH, with results shown in Figure 13. We observe that, similar to the trends of the experiments run with the Gemma 2B counterpart, BoN-RLV achieves the best BoN performance, while BoN-RL-S achieves the best Pass@N performance, with both substantially improving over the base model.

D.3.3 Gemma on held-out MATH benchmarks

To evaluate the generalization capabilities of our BoN-aware finetuned models, we additionally evaluate on two completely held-out and challenging benchmarks, Functional Math (Srivastava et al., 2024) and MathOdyssey (Fang et al., 2024). We present the results of these held-out benchmarks with Gemma 2B and 9B models in Figures 14,15,16,17, respectively, and observe that our fine-tuned models improve on both BoN and Pass@N for these held-out benchmarks.

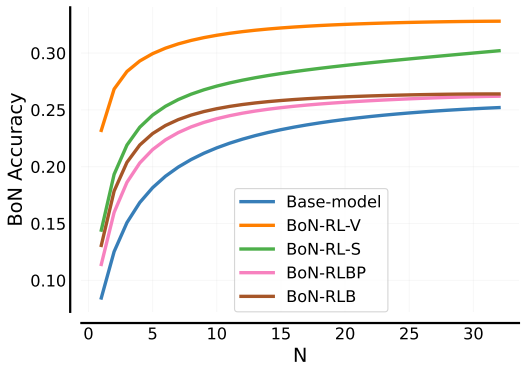
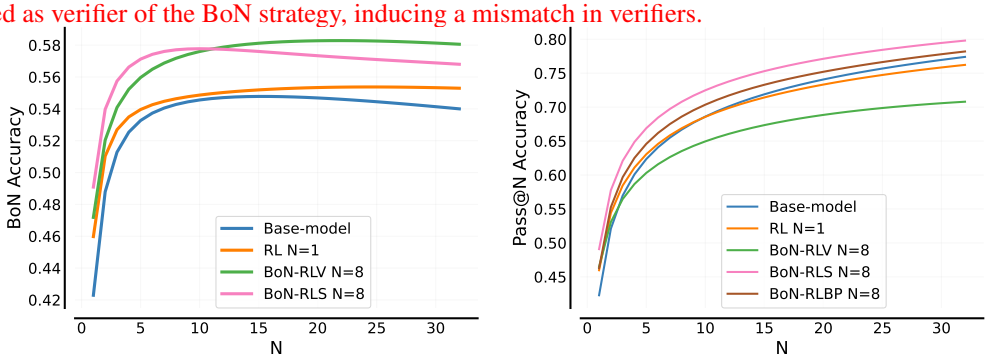


Figure 12: **Verifier Mismatch.** Plots show BoN accuracy under verifier-reward mismatch using Gemma 2B on MATH. During training verifier was used for BoN. On test, environment reward was used as verifier of the BoN strategy, inducing a mismatch in verifiers.

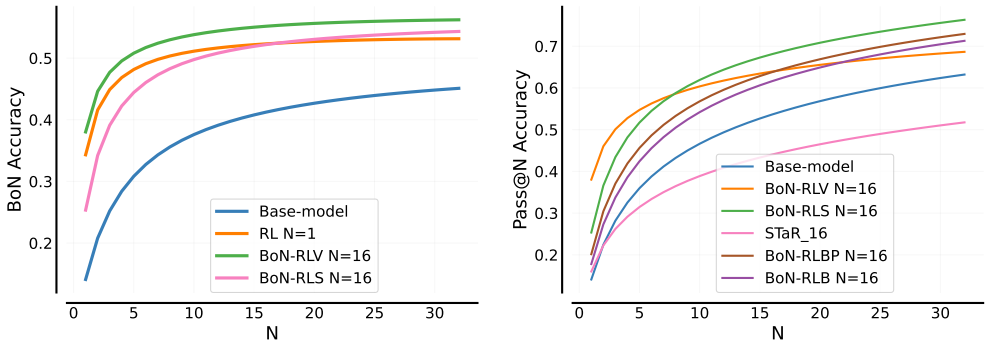


(a) BoN (b) Pass@N

Figure 13: BoN and Pass@N Accuracy Results on Hendrycks MATH with Gemma 9B.

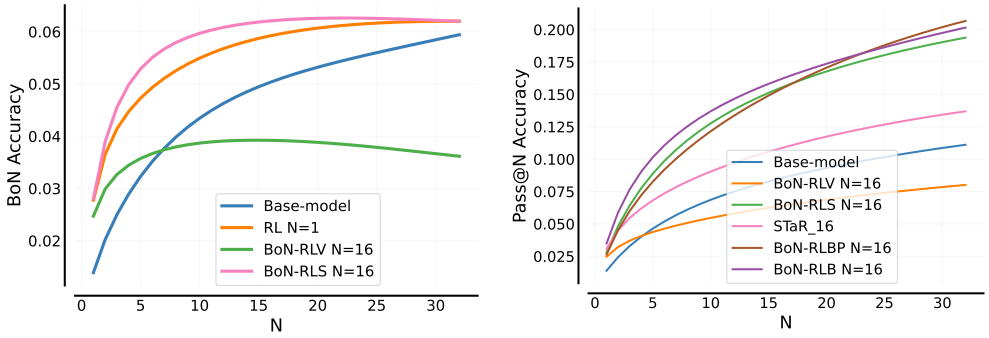
D.3.4 Gemma 2B on Coding Benchmarks

In Figure 18, we illustrate the performance of various Pass@N-aware finetuning methods and baselines that are trained with the MBPP coding dataset on HumanEval. We see that these methods, particularly BoN-RLBP, significantly improve upon the base model, increasing the pass@16 performance of the base Gemma 2B model from 61.6% to 67.1%. By contrast, standard RL fine-tuning (i.e with $N' = 1$) actually decreases the evaluation pass@16 to 59.8%.



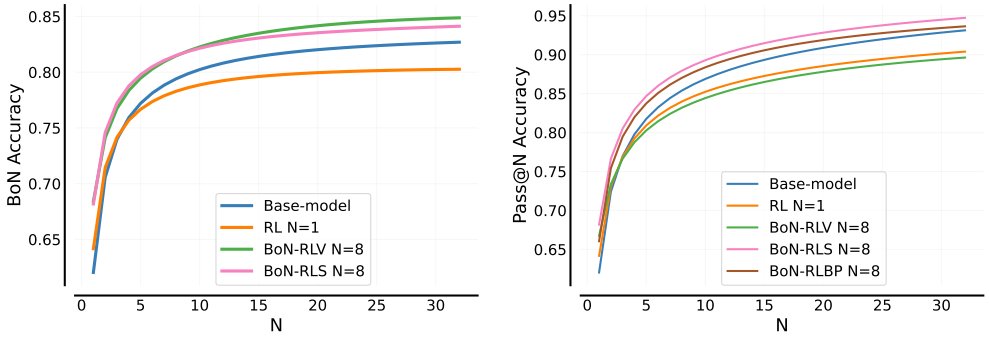
(a) BoN (b) Pass@N

Figure 14: BoN and Pass@N Accuracy Results on Functional Math with Gemma 2B.



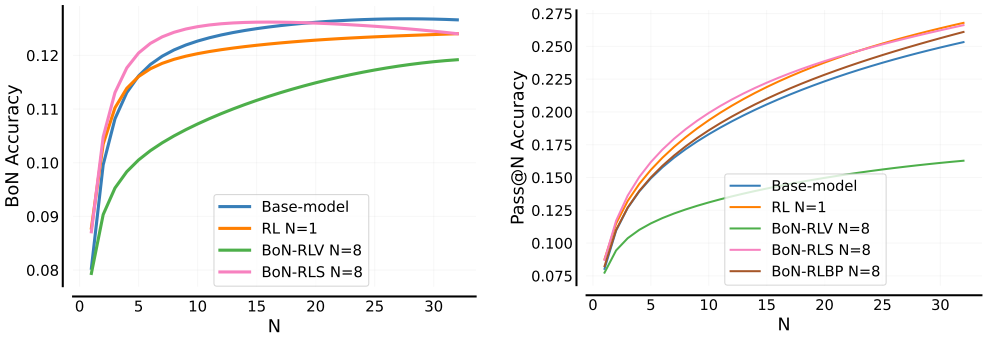
(a) BoN (b) Pass@N

Figure 15: BoN and Pass@N Accuracy Results on MathOdyssey with Gemma 2B.



(a) BoN (b) Pass@N

Figure 16: BoN and Pass@N Accuracy results on Functional Math with Gemma 9B.



(a) BoN (b) Pass@N

Figure 17: BoN and Pass@N Accuracy Results on MathOdyssey with Gemma 9b.

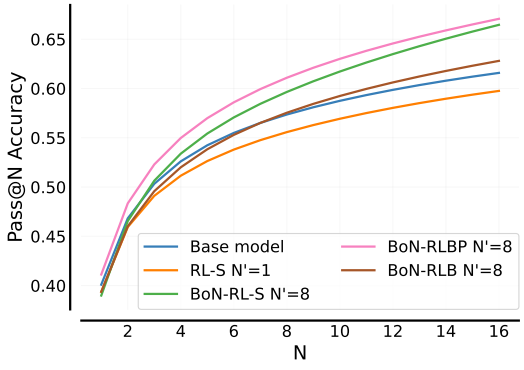


Figure 18: Pass@N Accuracy Results for Gemma 2B on HumanEval Coding Benchmark.