Mixture of Inputs: Text Generation Beyond Discrete Token Sampling

Yufan Zhuang[1], **Liyuan Liu**[2], **Chandan Singh**[2], **Jingbo Shang**[1], and **Jianfeng Gao**[2]

[1]UC San Diego
[2]Microsoft Research

Abstract

In standard autoregressive generation, an LLM predicts the next-token distribution, samples a discrete token, and then discards the distribution, passing only the sampled token as new input. To preserve this distribution's rich information, we propose Mixture of Inputs (MoI), a training-free method for autoregressive generation. After generating a token following the standard paradigm, we construct a new input that blends the generated discrete token with the previously discarded token distribution. Specifically, we employ a Bayesian estimation method that treats the token distribution as the prior, the sampled token as the observation, and replaces the conventional one-hot vector with the continuous posterior expectation as the new model input. MoI allows the model to maintain a richer internal representation throughout the generation process, resulting in improved text quality and reasoning capabilities. On mathematical reasoning, code generation, and PhD-level QA tasks, MoI consistently improves performance across multiple models including QwQ-32B, Nemotron-Super-49B, Gemma-3-27B, and DAPO-Qwen-32B, with no additional training and negligible computational overhead.

1 Introduction

Large language models (LLMs) are trained to predict the full distribution of the next token given an input context. To generate desirable sequences of text, various methods have been proposed to sample discrete tokens from these iterative next-token distributions [1, 2]. After the sampling process, only the discrete token is passed as the new input, and the rich predicted distribution is discarded. This process forces the model to commit to a single path in its reasoning, potentially abandoning valuable alternatives that could lead to better solutions.

On the other hand, human thinking first occurs in a high-dimensional and fluid manner before being articulated as natural language. Inspired by this cognitive process, we explore methods to enable LLMs to utilize not only articulated natural language but also partially-formed ideas, competing possibilities, and conceptual associations that exist in a probabilistic space before crystallizing into words.

Specifically, we propose Mixture of Inputs (MoI), a novel approach that takes as input not only a discrete, sampled token but also the sampled token's distribution. This preserves the model's uncertainty and allows it to conduct inner speech in a high-dimensional space. We employ a Bayesian estimation method, treating the token distribution as the prior and the sampled token as the observation, then replacing the conventional one-hot vector with the continuous posterior expectation. With this posterior expectation, a weighted average embedding is passed as the new input to subsequent prediction steps.

Code is available at: `https://github.com/EvanZhuang/mixinputs`.

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

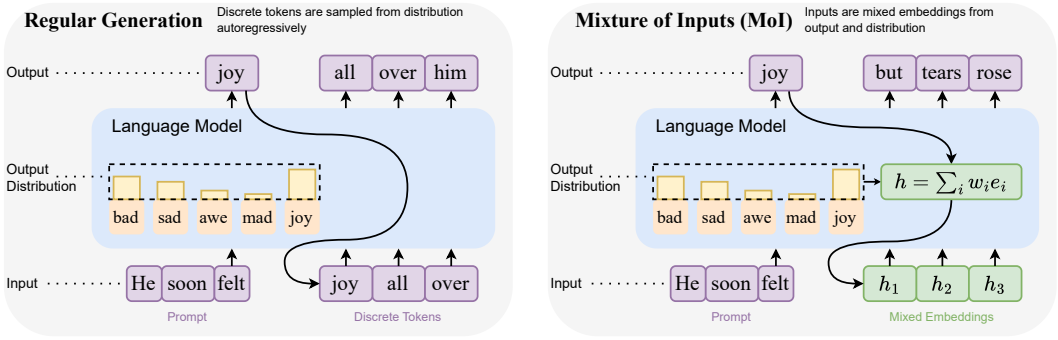


Figure 1: Comparison of the regular autoregressive generation pipeline (left) and our proposed Mixture of Inputs (MoI) strategy (right). In regular generation, only the discrete sampled token is fed back at each step, whereas MoI preserves the full sampling distribution by computing a blended embedding $h = \sum_i w_i e_i$, with weights w_i interpolating embeddings $\{e_i\}_{i=1}^{V}$, letting the model consider several plausible tokens simultaneously within a single forward pass.

MoI is conceptually intuitive and requires no additional training or architectural changes, making it immediately applicable to existing models. We implemented our method in modern LLM serving frameworks and found it to have negligible computational overhead and minimal deployment effort.

We evaluate MoI across a range of tasks—including mathematical reasoning, code generation, and graduate-level question answering—where maintaining uncertainty can play a crucial role in step-by-step inference. Across these domains, MoI brings consistent performance improvements for multiple models including QwQ-32B, Nemotron-Super-49B, Gemma-3-27B, and DAPO-Qwen-32B.

2 Related Work

Linearity of Embedding Representations The foundation of our work builds upon emerging research on the continuous nature of language model embedding spaces. Semantic linearity has been observed in embedding spaces dating back to word embedding models [3] and has been shown in various ways in modern LLMs [4–7]. A more recent work demonstrates that transformer language models naturally learn to process inputs and outputs as smooth, continuous functions rather than merely as discrete tokens [8]. This finding suggests that models inherently operate in a continuous latent space, even when traditionally constrained to discrete token processing. Similarly, Vector-ICL [9] shows that LLMs can effectively decode and process projected text embeddings via linear projection layers when provided with in-context demonstrations. While Vector-ICL projects external continuous data into the embedding space, our MoI directly leverages the linearity of the existing embedding space, demonstrating that meaningful representations can be created through linear combinations of token embeddings. Our work extends these insights by applying them specifically to preserve distributional information during the generation process, showing that this approach can enhance reasoning capabilities without model modifications.

Continuous Chain of Thought Chain-of-thought (CoT) prompting and related works improve language model performance by encouraging step-by-step reasoning through natural language [10–12]. However, these approaches rely on discrete text tokens, which can become inefficient and lengthy. More recently, COCONUT (Chain of Continuous Thought) [13] addresses this limitation by operating directly in the model's hidden state space rather than generating explicit text. By feeding the model's hidden state back as input, COCONUT enables more efficient reasoning that condenses lengthy thoughts into single tokens without the overhead of explicit thought generation. While COCONUT manipulates hidden states during multi-step reasoning processes, our MoI similarly leverages continuous representations but focuses specifically on the input embedding space during token generation. This key difference allows our approach to achieve improved reasoning without requiring architectural changes or model retraining, making it a more lightweight and accessible intervention.

Prompt and Weight Merging Linearity of LLM representations has been explored in a few related applications. Motivated by the success of methods that improve performance by ensembling multiple LLM calls [14–16], learning an ensemble of soft prompts or compressing a large prompt have been studied to enable strong performance without increasing computational cost [17–19]. Similarly, mechanistic methods for steering have proposed adding different latent vectors to elicit desired LLM behaviors [20–22]. The concept of linearity in neural networks extends beyond input representations to model parameters themselves. Recent work demonstrates that when two language models with shared initialization are combined through linear interpolation of their weights, their internal representations blend to produce a stronger model [23]. This discovery has enabled various model-merging techniques, from basic weight averaging to more sophisticated approaches [24, 25]. MoI applies similar linearity principles but at the level of individual tokens rather than full prompts or model weights.

3 Methods: Mixture of Inputs

When humans think, they often use natural language as an internal dialogue, but thinking is more fluid and multidimensional than just discrete words and sentences. Our cognition includes partially-formed ideas, competing possibilities, and conceptual associations that exist in a probabilistic space before crystallizing into specific language.

Our proposed method mirrors this cognitive reality by enabling LLMs to take as inputs both discrete tokens (representing specific linguistic choices) and token distributions (capturing the uncertainty, nuance, and competing possibilities that exist in human thought). By combining both as the model input, we obtain a richer representation that better reflects how human thinking operates — balancing the concrete and the probabilistic aspects of cognition.

Specifically, we introduce Mixture of Inputs (MoI). The core idea is to reinterpret token mixing as probabilistic inference under a Bayesian model. This formulation enables a principled mechanism to reconcile the model's prior belief (the output distribution) with its observed evidence (the sampled tokens), resulting in a more robust and statistically grounded method for input blending.

3.1 Token Generation and Embedding Aggregation

A key strength of MoI lies in its simplicity and modularity: it enhances the input representation without altering the model architecture or the underlying sampling algorithm. MoI operates after the language model produces its output distribution and before the next token is fed back into the model for the subsequent generation.

Token Generation Let $\{\boldsymbol{e}_i\}_{i=1}^{V} \in \mathbb{R}^d$ be embedding weights, with hidden dimension d and vocabulary size V. At each decoding timestep t, the language model outputs a probability distribution $\boldsymbol{p}_t = \{p_{t,i}\}_{i=1}^{V}$ over the vocabulary. This is typically followed by a sampling step that selects a token $\boldsymbol{y}_t$ (e.g., via top-k, nucleus sampling, or temperature scaling). In conventional approaches, the model would retrieve the embedding $\boldsymbol{e}_{y_t}$ corresponding to the sampled token and feed it into the next layer as the sole input.

MoI does not modify the sampling process itself: the sampled token y_t is still used as the output token. This design makes MoI fully compatible with any decoding strategy and seamlessly integrable into existing autoregressive generation pipelines.

Embedding Aggregation MoI first uses both the sampled token $\boldsymbol{y}_t$ and the distribution $\{p_{t,i}\}$ to compute $\{w_{t,i}\}$ as in Section 3.2, then uses $\{w_{t,i}\}$ to construct a *mixed embedding* vector $\boldsymbol{h}_t$.

$$\boldsymbol{h}_t = \sum_{i=1}^{V} w_{t,i}\boldsymbol{e}_i, \quad \text{where} \quad w_{t,i} \geq 0,\ \sum_i w_{t,i} = 1. \tag{1}$$

This representation allows the model to reason over a distribution of plausible next tokens rather than committing to a single discrete choice, effectively enabling a form of "inner speech" with richer representational capacity.

3.2 Bayesian Input Construction with MoI

To capture the distribution information, a naive idea might simply be to directly mix the inputs according to the output distribution, setting $w_{t,i} = p_{t,i}$. However, this approach only treats the token distribution as the input and neglects the sampled next token. In Section 6.1, we experiment with this approach (referred to as *Direct Mixture*) and find that it leads to performance degradation in most cases.

Instead, MoI combines two sources of information: (1) the output distribution $\boldsymbol{p}_t$, representing the model's prior belief over possible next tokens, and (2) the sampled token $\boldsymbol{y}_t$, representing a concrete observation drawn from this belief.

To reconcile these two sources, MoI treats the sampling process as probabilistic evidence and formulates the blending of representations as a Bayesian inference problem. Specifically, it constructs a posterior-weighted mixture over token embeddings by computing a new weight vector $\boldsymbol{w}_t = \{w_{t,i}\}_{i=1}^{V}$ that incorporates both the uncertainty in $\boldsymbol{p}_t$ and the evidence from $\boldsymbol{y}_t$.

The resulting mixed embedding $\boldsymbol{h}_t$ is given by Equation 1, and it replaces the embedding for the discrete token as the input to the next decoding step (i.e., replaces $\boldsymbol{e}_{y_t}$ with $\boldsymbol{h}_t$). What changes is the internal representation passed into the model, allowing the decoder to reason over both the chosen token and the context of plausible alternatives.

4 Mixing Weight Estimation

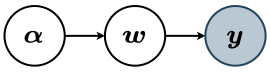
Here, we elaborate our proposed Bayesian estimation method for $\boldsymbol{w}_t = \{w_{t,i}\}_{i=1}^{V}$.

4.1 Dirichlet Mixture Model

In probabilistic modeling, a prior encodes belief before observing new data. Accordingly, we begin by constructing a prior distribution over token choices based on the model's output logits. Specifically, we assume the prior distribution to be Dirichlet, with concentration parameter $\boldsymbol{\alpha}$.

We view $\boldsymbol{y}$ as the output of the sampling process and assume the sampled token comes from a multinomial distribution parametrized by $\boldsymbol{w}$. Then, we estimate the mixing weight $\boldsymbol{w}$ by conducting the posterior estimation.

$$\begin{aligned} &\boldsymbol{w} \sim \mathrm{Dir}(\boldsymbol{\alpha}), && \text{where } \boldsymbol{\alpha} = \mathrm{H}(\boldsymbol{p}) \cdot \boldsymbol{p}, \\ &\boldsymbol{y} \sim \mathrm{Multinomial}(\boldsymbol{w}) && \mathrm{H}(\boldsymbol{p}) \text{ is the normalized entropy of } \boldsymbol{p} \end{aligned}$$



This formulation ensures that tokens with higher model confidence (i.e., lower entropy) exert stronger influence on the posterior, while still respecting the sampled outcome. We will go over each part of the Bayesian model in the following sections.

4.2 Estimating Mixing Weight

Let $\boldsymbol{p}_t \in \Delta^{V-1}$ be the next-token distribution at step t and let $y_{t,i} \in \{0, 1\}$ indicate the sampled token ($y_{t,i} = 1$ iff token i is chosen). We estimate the mixing weights $\boldsymbol{w}_t$ by Bayesian posterior inference in a Dirichlet–Multinomial model.

Entropy-scaled prior. Define the *normalized entropy* H as the following

$$H \;\coloneqq\; \mathrm{H}(\boldsymbol{p}_t) = -\frac{1}{\log V} \sum_{i=1}^{V} p_{t,i} \log p_{t,i} \qquad H \in [0, 1]. \tag{2}$$

We place a Dirichlet prior

$$\boldsymbol{w}_t \sim \mathrm{Dir}(\boldsymbol{\alpha}), \qquad \boldsymbol{\alpha} = \mathrm{H}(\boldsymbol{p}_t)\, \boldsymbol{p}_t, \tag{3}$$

so that the total concentration $\sum_i \alpha_i = \mathrm{H}(\boldsymbol{p}_t)$ grows with uncertainty and vanishes when the model is confident. So when uncertainty is high, the prior distribution will be more widespread over $\boldsymbol{p}_t$, and vice versa.

Pseudo-count observation. The sampled token contributes a single pseudo-count whose weight increases as confidence rises:

$$c_i = \big(\beta + 1 - H\big)\, y_{t,i}, \qquad \text{with hyperparameter } \beta. \tag{4}$$

The hyperparameter β controls the concentration over mixing weight. Smaller values emphasize more on output distributions, while larger values highlight more the sampled output token. The effect of β is easier to observe in Eq. (5). We also conduct an analysis of β's empirical effect in Section 7.1.

Posterior mean. Dirichlet conjugacy yields the posterior mean of $\boldsymbol{w}_t$, and we use that estimation as our mixing weights:

$$w_{t,i} = \frac{\boldsymbol{\alpha}_i + c_i}{\sum_i \boldsymbol{\alpha}_i + N} = \frac{H\, p_{t,i} \;+\; \big(\beta + 1 - H\big)\, y_{t,i}}{\beta + 1}, \quad \text{with } \; N = \sum_i c_i \tag{5}$$

Behavior of w. Eq. (5) smoothly interpolates between the distribution ($w_{t,i} \to p_{t,i}$ when $H \to 1$) and the one-hot token ($w_{t,i} \to y_{t,i}$ when $H \to 0$), thereby reconciling distributional and discrete evidence in a single principled estimator.

The complete procedure for computing the mixture of inputs is summarized in Algorithm 1.

Algorithm 1: Mixture of Inputs

Require: Sampling distribution $\boldsymbol{p}_t$, sampled token $\boldsymbol{y}_t$, hyperparameter β, and embeddings $\{e_i\}_{i=1}^{V}$.
1. Compute entropy H with Eq. 2
2. Compute mixing weight $w_{t,i}$ with Eq. 5
return $\boldsymbol{h}_t = \sum_i w_{t,i}\, \boldsymbol{e}_i$

5 Experimental Setup

We evaluate MoI across a diverse suite of benchmarks spanning competition mathematics, combinatorial problem solving, program synthesis, and graduate-level question answering. These tasks vary widely in structure and domain, allowing us to assess MoI's generality and effectiveness across distinct application settings.

5.1 Tasks and Metrics

To ensure a comprehensive evaluation, we select four challenging benchmarks that span distinct reasoning domains and require different cognitive skills, from symbolic manipulation to procedural generation and scientific comprehension:

AIME [26] consists of complex high-school level mathematical problems that often require multiple stages of symbolic reasoning, algebraic manipulation, and geometric insight. We use the official AIME datasets from 2022 to 2024 and evaluate models based on exact match accuracy, reflecting their ability to arrive at precise, correct solutions.

Count Down 4 [27] is a synthetic numerical reasoning task that presents models with arithmetic puzzles. It requires deriving a target number by applying a sequence of operations (addition, subtraction, multiplication, division) on a fixed set of four input numbers. This benchmark emphasizes procedural and combinatorial reasoning. We report the success rate, indicating whether the model arrives at the correct final equation.

LiveCodeBench [28] is a dynamic and realistic code generation benchmark that includes tasks ranging from simple string manipulations to advanced data structures and algorithms. Each problem specifies a goal in natural language, and the model must generate executable code that meets functional correctness criteria. We use pass@1—the proportion of correct solutions on the first attempt—as the primary evaluation metric.

GPQA [29] is a highly challenging multiple-choice question answering benchmark drawn from graduate-level science and engineering exams. Its diamond subset features the most difficult questions that demand domain-specific knowledge, long-range reasoning, and the integration of multiple concepts. We evaluate models based on multiple-choice accuracy.

5.2 Models

We evaluate MoI using 4 state-of-the-art open-source LLMs with advanced reasoning capabilities.

QwQ-32B [30] is optimized for mathematical and logical reasoning through a curriculum of instruction tuning on symbolic tasks, math word problems, and chain-of-thought datasets.
Llama-3.3-Nemotron-49B [31] is derived from Meta's Llama 3.3 70B model [32]. The model underwent neural architecture search to optimize for inference efficiency, followed by supervised fine-tuning and reinforcement learning. These techniques were applied to enhance the model's reasoning abilities, instruction following capabilities, and tool-calling performance.
Gemma-3-27B [33] is part of Google's Gemma 3 family—multimodal (text + image) models with 128 K token context windows and an integrated SigLIP vision encoder. The 27B variant is instruction-tuned for chat and reasoning.
DAPO-Qwen-32B [34] is a customized version of Qwen2.5-32B [35] that incorporates Decoupled Clip and Dynamic Sampling Policy Optimization (DAPO), which stabilizes and scales RL for long chain-of-thought reasoning. This model is designed to encourage faithful and step-consistent reasoning trajectories.

5.3 Baselines

To quantify the benefit of MoI, we compare it with two decoding schemes that keep the underlying model architecture and sampling mechanism fixed. The primary baseline (*Standard*) is the widely used nucleus sampling with temperature scaling [1]. It represents the default inference recipe shipped with each model. Our second baseline (*Direct Mixture*) constructs the input representation as a simple weighted sum of token embeddings using the softmax probabilities as coefficients, i.e., computing the value of $\boldsymbol{h}_t$ as $\sum_{i=1}^{V} p_{t,i}\boldsymbol{e}_i$. Unlike MoI, it performs no Bayesian reconciliation between the distribution and the sampled token, providing a stringent ablation for assessing the value of our posterior estimator. We also tried directly feeding the mixed output hidden states, but we found that the models cannot make sense of the hidden states without retraining.

5.4 Hyperparameter Settings

We perform 5 runs for all experiments and report the average. For AIME and Count Down 4, we perform hyperparameter grid search on baselines, Direct Mixture and MoI with $\beta \in \{\frac{1}{4}, \frac{1}{2}, 1, 2, 4, 8\}$, $T \in \{0.6, 0.8, 1\}$ and top-p $\in \{0.4, 0.6, 0.8, 0.95\}$. We report the mean result of the best configuration for all three methods. We investigate the importance of these hyperparameters in Section 6.2. For GPQA-Diamond and LiveCodeBench, we use the universal hyperparameter for all of them with $T = 0.6$, top-p $= 0.95, \beta = 1$; more details can be found in Appendix F.

6 Main Results

6.1 MoI Boosts Capabilities of LLMs

Table 1 reports accuracy on four reasoning-intensive benchmarks for four open-source LLMs. Across all 16 model–task pairs, our approach MoI either matches or outperforms the *Standard* autoregressive baseline, with an average absolute gain of 1.8%. In contrast, the ablation that removes distribution–smoothing (*Direct Mixture*) degrades performance in most cases, underscoring the importance of our Bayesian smoothing.

Consistency across model scales. MoI achieves gains for both medium-sized (Gemma-3-27B) and larger (32 to 49 B-parameter) models. The largest improvement appears on Nemotron-Super-49B, where MoI adds up to +4.1% on GPQA-Diamond and +2.6% on Count Down 4, lifting the overall average to 55.45% (+2.36%). These results indicate that mixture-of-inputs remains beneficial even when the underlying model already possesses strong zero-shot reasoning abilities.

Task-specific trends. Improvements are most pronounced on benchmarks requiring extended symbolic manipulation. Count Down 4 benefits the most (+3.7% mean gain), suggesting that explicitly representing uncertainty over arithmetic operations mitigates the compounding error typical in multi-step numerical reasoning. Gains on AIME and GPQA-Diamond further show that MoI generalizes from high-school mathematics to graduate-level science QA, while LiveCodeBench sees more modest but still positive changes.

Table 1: Main results on four benchmarks with four large language models. The "Input Info." column indicates the source of input passed into the model: *Output Token* uses only the sampled discrete token, *Output Dist.* uses the full output probability distribution, and *Token + Dist.* combines both. Accuracy (%) is reported on AIME, Count Down 4, GPQA-Diamond, and pass@1 is used on LiveCodeBench. *Standard* uses conventional sampling, that is temperature–scaled nucleus sampling, *Direct Mixture* removes the posterior estimation, and MoI is our full approach. Shaded cells highlight MoI and its performance gain (absolute difference over the conventional generation).

Model	Method	Input Info.	AIME	CountDown4	GPQA-D	LiveCodeBench	Avg
QwQ-32B	Standard	Output Token	77.78	79.25	58.08	76.32	72.86
	Direct Mixture	Output Dist.	72.00	66.88	51.52	53.42	60.96
	MoI	Token + Dist.	80.00	80.01	60.10	76.51	74.15
	Gain vs. Standard		**+2.22**	**+0.76**	**+2.02**	**+0.19**	**+1.29**
Nemotron-Super-49B	Standard	Output Token	54.89	56.93	60.60	39.92	53.09
	Direct Mixture	Output Dist.	60.00	51.72	60.10	16.04	46.97
	MoI	Token + Dist.	57.11	59.53	64.65	40.50	55.45
	Gain vs. Standard		**+2.22**	**+2.60**	**+4.05**	**+0.58**	**+2.36**
Gemma-3-27B	Standard	Output Token	25.56	56.51	46.97	31.31	40.09
	Direct Mixture	Output Dist.	26.44	55.47	51.52	31.99	41.36
	MoI	Token + Dist.	26.89	59.38	47.47	32.87	41.65
	Gain vs. Standard		**+1.33**	**+2.87**	**+0.50**	**+1.56**	**+1.56**
DAPO-Qwen-32B	Standard	Output Token	64.67	72.03	42.42	54.01	58.28
	Direct Mixture	Output Dist.	62.67	67.19	37.88	23.87	47.90
	MoI	Token + Dist.	64.44	78.75	42.93	55.18	60.33
	Gain vs. Standard		**-0.23**	**+6.72**	**+0.51**	**+1.17**	**+2.05**

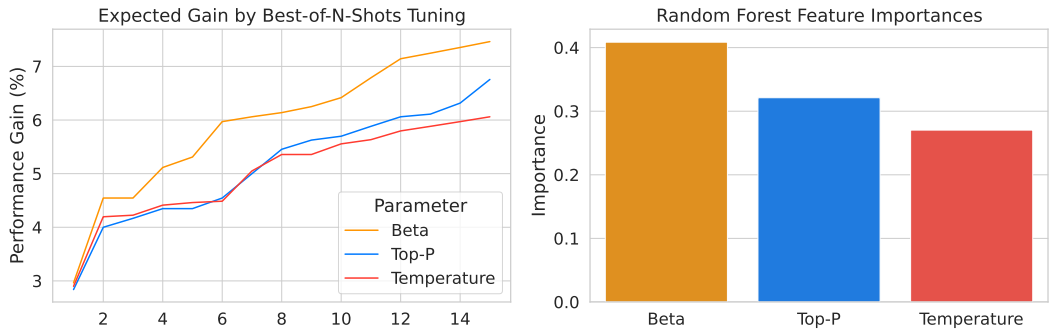


Figure 2: **Hyperparameter Importance Analysis.** Comparison of three key hyperparameters (β in MoI, top-p, and temperature) across four LLMs on two mathematical reasoning tasks. **Left:** Expected performance gain (%) when optimizing each hyperparameter individually through best-of-N-shots tuning. The graph shows β consistently outperforms other parameters as N increases. **Right:** Relative feature importance derived from random forest regression analysis, confirming β's strong influence (0.41) on model performance compared to top-p (0.32) and temperature (0.27). These results demonstrate that β is highly influential for effectively controlling input mixing during chain-of-thought reasoning.

Role of autoregressive inputs. Feeding back the *full* output distribution alone is insufficient: *Direct Mixture* often harms accuracy (e.g., -22.9% on LiveCodeBench for Nemotron-Super-49B). The combination of the sampled token *and* its distributional context lets the model retain a discrete anchor while preserving alternative hypotheses, yielding the best of both worlds.

Together, these findings demonstrate that MoI offers a principled and consistently effective way to enhance multi-step reasoning. By marrying discrete choices with probabilistic context, it improves accuracy without sacrificing decoding efficiency or requiring model-specific fine-tuning.

6.2 Hyperparameter Importance Analysis

To understand which factors most strongly influence reasoning performance, we analyze three key hyperparameters: β, top-p, and temperature. This analysis spans four LLMs and two mathematical

reasoning tasks, with multiple runs and grid search over the hyperparameter space, as described in Section 5.4.

Fig. 2 provides two complementary perspectives on hyperparameter importance. The left plot tracks expected performance gain when optimizing each parameter individually through best-of-N-shots tuning, with experiment setup explained in Appendix E.1. As N increases from 1 to 15, β consistently yields the highest gains, reaching nearly 7.5% improvement at N=15, while top-p and temperature plateau at approximately 6.0-6.5%. This separation becomes particularly pronounced after N=10, suggesting that β's impact grows with more extensive search.

The right panel quantifies each parameter's importance through random forest regression analysis, with experiment setup explained in Appendix E.2. With inputs as hyperparameters and accuracy as the target, this reveals β as the dominant factor (importance score of 0.41), followed by top-p (0.32) and temperature (0.27).

7 Analysis

7.1 Task-Dependent Optimal Mixing Strategies

Different reasoning tasks may benefit from varied degrees of distribution mixing. To investigate this phenomenon, we analyze the parameter sensitivity of two distinct benchmark types: **AIME** (requiring advanced mathematical reasoning) and **Count Down 4** (demanding extensive combinatorial enumeration). Fig. 3 visualizes how performance varies with the mixing parameter β across four LLMs, showing the deviation from each task's global mean accuracy.

The results reveal an interesting inverse relationship between task type and optimal β values. AIME performance peaks at low β values ($\beta \leq 1$), with accuracy dropping sharply when $\beta > 1$. In contrast, Count Down 4 shows the opposite pattern, performing substantially below average at low β values but excelling when $\beta > 1$. This divergence suggests fundamental differences in how distribution mixing affects distinct reasoning processes.

For reasoning-intensive AIME problems, low β values promote greater consideration of alternative solution paths while maintaining focus on the most promising directions. Conversely, for enumeration-intensive Count Down 4 problems, higher β values increase concentration on the most probable combinations, effectively pruning the vast search space.

These findings highlight the importance of task-appropriate β calibration when deploying MoI. Lower values suit open-ended reasoning, while higher values suit systematic enumeration—an adaptability that fixed decoding strategies lack.

7.2 Case Study: Linear Prompt Blending with Various Lengths

Although our main experiments focus on token-by-token blending at generation time, we also investigate whether a similar blending strategy applied to instruction prompts of varying lengths can boost performance. To this end, we perform 10-shot in-context learning on five sentiment analysis benchmarks using three medium-sized LLMs, building on prior work showing that prompt wording and structure have a major impact on classification accuracy [15].

We assembled three prompt pools: (1) binary sentiment analysis on Rotten Tomatoes [36], SST2 [37], and IMDB [38], consisting of 96 prompts of length 3–16 words (mean 7.57); (2) 6-class emotion classification on the Emotion dataset [39], with 32 prompts of length 3–15 words (mean 7.27); and (3) 3-class financial sentiment on the Financial Phrasebank [40], comprising 40 prompts of length 2–14 words (mean 6.51).

Our blending procedure first linearly extrapolates each prompt's embedding to the maximum length in its pool and then averages these fixed-length embeddings to form a single "blended" prompt representation. This approach integrates semantic nuances from all constituent prompts while preserving their instructional intent.

Table 2 reports 10-shot accuracy under the expectation over randomly drawn single-prompt, the blended-prompt, and the absolute gain over baseline. Across most benchmarks and models, linear prompt blending consistently outperforms random single-prompt selection, further demonstrating that embedding-space mixing can be highly effective for boosting LLMs' capacity.

Table 2: 10-shot in-context learning accuracy (%) for three LLMs (Llama3 8B [32], Mistral 7B [41], Gemma 7B [42]) on five sentiment analysis benchmarks. We compare the expectation of a single-prompt baseline against embedding-space prompt blending via linear interpolation of prompts.

Model	Method	Rotten Tomatoes	SST2	IMDB	Emotion	Financial Phrasebank	Average
Llama3 8B	Single Prompt	91.58	94.12	87.26	53.82	68.76	79.11
	Linear Interpolation	92.68	94.49	95.40	51.75	72.34	81.33
	Gain	**+1.10**	**+0.37**	**+8.14**	**-2.07**	**+3.58**	**+2.22**
Mistral 7B	Single Prompt	89.32	91.11	85.06	54.87	70.75	78.22
	Linear Interpolation	92.21	94.03	92.82	51.60	73.42	80.82
	Gain	**+2.89**	**+2.92**	**+7.76**	**-3.27**	**+2.67**	**+2.59**
Gemma 7B	Single Prompt	86.66	87.18	87.31	50.77	72.63	76.91
	Linear Interpolation	92.30	93.34	93.88	50.30	74.39	80.84
	Gain	**+5.64**	**+6.16**	**+6.57**	**-0.47**	**+1.76**	**+3.93**

7.3 Throughput Analysis

The mixing weight calculation is lightweight and efficient. We perform a throughput analysis, shown in Table 3, to examine the runtime overhead added by MoI. We measure generation statistics for solving the Count Down 4 task, and we record the average input and output throughput over 5 runs. To better compare the throughput, we picked the benchmarks where the generation length is about the same. The median difference between MoI-generated text and baseline-generated text is 1.7%.

Table 3: Throughput analysis (tokens/s) for QwQ-32B with and without MoI in vLLM.

Method	Input Speed	Output Speed
Standard	62.87	1,143.31
MoI	61.36	1,101.44
Overhead	2.40%	3.66%

8 Discussion

Limitations and Future Work While MoI demonstrates consistent gains on a wide range of benchmarks, its current scope is intentionally focused on tasks that can be objectively evaluated. As a result, applications such as open-ended generation or creative writing, where objectives are less formally defined, remain outside the current scope and present promising directions for further study. Additionally, we observe that the hyperparameter β exhibits task-dependent behavior. This suggests that different task types benefit from varying degrees of distributional mixing, a phenomenon worthy of deeper theoretical exploration. Future work could investigate adaptive or test-time β tuning strategies.

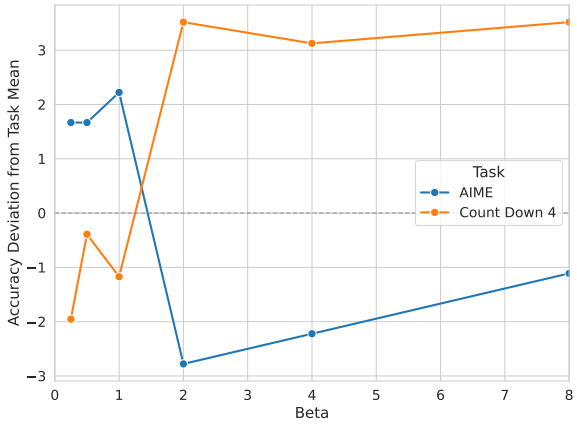


Figure 3: **Task-dependent Optimal Mixing Strategies.** The plot shows accuracy deviation from task mean across different β values for AIME (reasoning-heavy) and Count Down 4 (enumeration-heavy), averaged across four LLMs. Lower β values ($\beta \leq 1$) significantly benefit AIME's performance while higher β values ($\beta > 1$) improve Count Down 4. This divergence demonstrates how MoI's impact varies based on task characteristics: reasoning-intensive tasks perform better with stronger distribution mixing (low β) to be more creative, while enumeration-intensive tasks benefit from higher distribution mixing (high β) that helps explore the combinatorial search space with more focus.

Conclusion We presented Mixture of Inputs (MoI), a training-free enhancement to autoregressive generation that preserves distributional information. By treating input mixing as a Bayesian inference problem, MoI maintains a richer internal representation throughout the generation process, allowing models to conduct a form of inner speech beyond discrete tokens while requiring no architectural changes or additional training.

Our evaluation across LLMs and benchmarks demonstrates consistent performance improvements. MoI's conceptual simplicity, negligible computational overhead, and immediate applicability to existing models make it a practical enhancement that bridges the gap between the high-dimensional nature of thought and the discrete nature of language.

Acknowledgement

Our work is sponsored in part by NSF CAREER Award 2239440, NSF Proto-OKN Award 2333790, Sponsored Research Projects from companies like Cisco and eBay, as well as generous gifts from Google, Adobe, and Teradata. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and should not be interpreted as necessarily representing the views, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes not withstanding any copyright annotation hereon.

References

[1] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[2] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[3] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013.

[4] Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

[5] Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. *arXiv preprint arXiv:2309.00941*, 2023.

[6] Yibo Jiang, Goutham Rajendran, Pradeep Ravikumar, Bryon Aragam, and Victor Veitch. On the origins of linear representations in large language models. *arXiv preprint arXiv:2403.03867*, 2024.

[7] Jack Merullo, Noah A Smith, Sarah Wiegreffe, and Yanai Elazar. On linear representations and pretraining data frequency in language models. *arXiv preprint arXiv:2504.12459*, 2025.

[8] Samuele Marro, Davide Evangelista, X Angelo Huang, Emanuele La Malfa, Michele Lombardi, and Michael Wooldridge. Language models are implicitly continuous. *arXiv preprint arXiv:2504.03933*, 2025.

[9] Yufan Zhuang, Chandan Singh, Liyuan Liu, Jingbo Shang, and Jianfeng Gao. Vector-icl: In-context learning with continuous vector representations. *arXiv preprint arXiv:2410.05629*, 2024.

[10] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[11] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

[12] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback, 2023.

[13] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

[14] Amanda Bertsch, Alex Xie, Graham Neubig, and Matthew R Gormley. It's mbr all the way down: Modern generation techniques through the lens of minimum bayes risk. *arXiv preprint arXiv:2310.01387*, 2023.

[15] John X Morris, Chandan Singh, Alexander M Rush, Jianfeng Gao, and Yuntian Deng. Tree prompting: efficient task adaptation without fine-tuning. *arXiv preprint arXiv:2310.14034*, 2023.

[16] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

[17] Guanghui Qin and Jason Eisner. Learning how to ask: Querying lms with mixtures of soft prompts. *arXiv preprint arXiv:2104.06599*, 2021.

[18] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*, 2023.

[19] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9):1–35, 2023.

[20] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

[21] Nishant Subramani, Nivedita Suresh, and Matthew E Peters. Extracting latent steering vectors from pretrained language models. *arXiv preprint arXiv:2205.05124*, 2022.

[22] Liu Yang, Ziqian Lin, Kangwook Lee, Dimitris Papailiopoulos, and Robert Nowak. Task vectors in in-context learning: Emergence, formation, and benefit. *arXiv preprint arXiv:2501.09240*, 2025.

[23] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022.

[24] Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36:7093–7115, 2023.

[25] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[26] Mathematical Association of America. American invitational mathematics examination (aime), 2024. `https://maa.org/maa-invitational-competitions/`.

[27] Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. Tinyzero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

[28] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

[29] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

[30] Qwen. Qwq-32b: Embracing the power of reinforcement learning, March 2025.

[31] NVIDIA. Llama-nemotron: Efficient reasoning models, 2025.

[32] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[33] Gemma, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

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

[35] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[36] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, 2005.

[37] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[38] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[39] Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[40] P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65, 2014.

[41] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[42] Gemma, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[43] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[45] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[46] Anthropic. Claude 3.5. `https://www.anthropic.com`, 2023. Accessed: 2025-05-15.

[47] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[48] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024.

A Comparing over Hyperparameter Grid Search

We perform a head-to-head evaluation between our method (MoI) and the standard text generation with temperature-scaled nucleus sampling (baseline), under two complementary regimes:

Best-case: each method is run with its single best-performing hyperparameter configuration, Fig. A1 summarizes the results.

Grid-average: performance is averaged across all combinations in the hyperparameter grid (see details in Section 5.4). Fig. A2 provides these averages and confirms that the gains are not an artifact of cherry-picking one lucky setting.

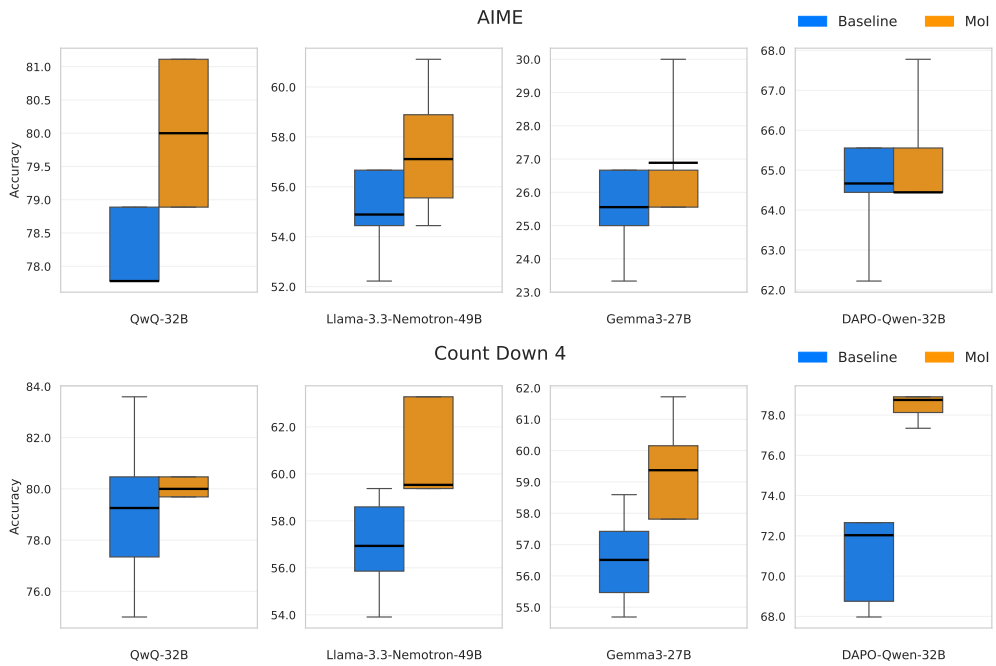


Figure A1: We show a comparison of distributions of evaluation results across the best top-p and temperature hyperparameter for baseline and with MoI. The results indicate strong performance gain brought by incorporating the sampling distribution in the generation process.

B Generalization of a single hyper-parameter setting across tasks

A practical concern when adding new decoding hyperparameters is whether the values tuned on one task will transfer to others. To investigate this, we conduct a grid search over the hyperparameters described in Section 5.4. For each (method, model) pair, we keep the configuration that maximizes the average accuracy on AIME & CountDown 4 and freeze it for the remainder of the study.

Table A1 shows the results on the held-out GPQA-Diamond and LiveCodeBench. The single-tuned setting consistently outperforms Standard nucleus sampling on 15 of the 16 (model, task) pairs.

Overall, these results suggest that MoI requires only modest tuning effort: once the hyperparameters are calibrated on a small proxy set, they generalize robustly to new tasks and domains without much further adjustment.

C Results on Instruction-following Tasks

We conducted additional experiments on MT-Bench [43], using the four larger models, and the two smaller models, Llama-3.1-8B [32] and Qwen-2.5-14B [35]. Below, we report the average performance for standard decoding (temperature=0.6, top-p=0.95) and our Mixture of Inputs (MoI)

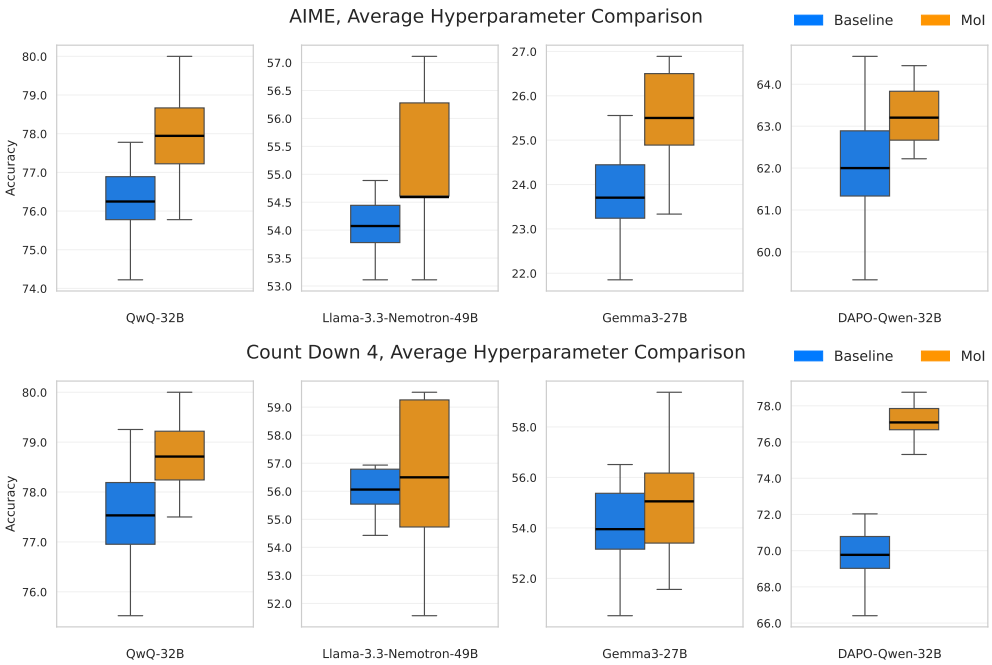


Figure A2: We show a comparison of distributions of evaluation results across all top-p and temperature hyperparameters. The results indicate almost universal performance gain across average hyperparameter settings.

Table A1: Additional results on four benchmarks with four large language models. For every (method, model) we tune the decoding hyperparameters on AIME and CountDown 4, fix the best setting, and reuse it when assessing GPQA-Diamond and LiveCodeBench. Accuracy (%) is reported on AIME, Count Down 4, GPQA-Diamond, and pass@1 is used on LiveCodeBench. *Standard* uses conventional sampling, that is temperature–scaled nucleus sampling and MoI is our full approach.

Model	Method	Input Info.	AIME	CountDown4	GPQA-D	LiveCodeBench	Avg
QwQ-32B	Standard	Output Token	76.89	78.04	58.08	76.48	72.37
	MoI	Token + Dist.	78.44	79.22	61.41	77.85	74.23
	Gain vs. Standard		**+1.55**	**+1.18**	**+3.33**	**+1.37**	**+1.86**
Nemotron-Super-49B	Standard	Output Token	54.67	56.77	63.64	40.35	53.86
	MoI	Token + Dist.	57.11	59.22	64.95	40.90	55.55
	Gain vs. Standard		**+2.44**	**+2.45**	**+1.31**	**+0.55**	**+1.69**
Gemma-3-27B	Standard	Output Token	24.44	55.21	45.45	32.37	39.37
	MoI	Token + Dist.	26.67	57.19	46.26	32.92	40.76
	Gain vs. Standard		**+2.23**	**+1.98**	**+0.81**	**+0.55**	**+1.39**
DAPO-Qwen-32B	Standard	Output Token	64.67	70.31	42.93	56.91	58.70
	MoI	Token + Dist.	61.56	75.00	43.94	57.50	59.50
	Gain vs. Standard		**-3.11**	**+4.69**	**+1.01**	**+0.59**	**+0.80**

decoding (β=1,2,4, same decoding configuration), together with the relative improvements brought by MoI (Δ). As summarized in Table A2, MoI consistently outperforms the baseline for all models.

We also noticed MoI reduced repetition in the response. Particularly, we measured the 3-gram and 4-gram repetition rate in the response for MT-bench and found MoI reduces the repetition on 72.9% of responses, reducing the average 4-grams repetition rate from 4.27% to 3.13%. We also provide a side-by-side comparison on a coding task as below (QwQ-32B).

With standard sampling (repetitive around HCA definition):

To address the problem of finding the highest common ancestor (HCA) of two nodes in a binary tree, we need to clarify the definition. Based on common terminology, the lowest common ancestor (LCA) is the deepest node shared by both paths from

the nodes to the root. However, since the problem specifies "not LCA," we assume the HCA refers to the shallowest common ancestor (closest to the root) [content continues...]

With mixture of inputs sampling (reduced repetitiveness):

To find the highest common ancestor (HCA) of two nodes in a binary tree, we can utilize a recursive approach similar to finding the Lowest Common Ancestor (LCA). The HCA is the deepest node that is an ancestor of both given nodes. Here's how to implement this: [content continues...]

Table A2: Comparison of conventional sampling (Baseline) and MoI scores across six large language models. For each model, we report the baseline score, MoI score, and the relative improvement ($\Delta\%$) of MoI over the baseline.

Model	Baseline Score	MoI Score	$\Delta\%$ (MoI vs Baseline)
QwQ-32B	9.25	9.51	**+2.81%**
Nemotron-Super-49B	9.41	9.48	**+0.74%**
Gemma-3-27B	9.05	9.38	**+3.65%**
DAPO-Qwen-32B	8.96	9.46	**+5.54%**
Llama-3.1-8B	8.24	8.65	**+4.98%**
Qwen-2.5-14B	8.87	9.32	**+5.07%**

D Statistical Robustness of the Results

We conducted formal significance testing. Specifically, we applied McNemar's test to compare MoI against the baseline across all four benchmarks for each model. Table A3 reports the resulting p-values. These results validate that the observed gains are statistically significant in most cases.

To further address concerns about variance, we conducted an extensive 64-run evaluation on the AIME dataset across four models with the same configuration as in experiments of Table 1. The results in Table A4 confirm consistent improvements from MoI.

Table A3: McNemar's test p-values comparing model performance across four benchmarks. Lower values indicate statistically significant differences between baseline and MoI.

Model	CountDown4	AIME	GPQA-Diamond	LiveCodeBench
QwQ-32B	< 0.001	< 0.001	0.003	0.04
Nemotron-Super-49B	< 0.001	< 0.001	< 0.001	0.02
Gemma-3-27B	< 0.001	< 0.001	0.04	< 0.001
DAPO-Qwen-32B	< 0.001	0.8	0.02	< 0.001

Table A4: Comparison of baseline and MoI performance across four large language models over 64 runs on AIME. Each result reports the mean (μ), standard deviation (σ) and min-max score range. We observe consistent performance gain brought by MoI.

Model	Baseline ($\mu \pm \sigma$)	Baseline Range	MoI ($\mu \pm \sigma$)	MoI Range	Gain
QwQ-32B	76.02 ± 2.51	72.22–82.22	77.66 ± 2.06	73.33–82.22	**+1.64**
Nemotron-Super-49B	54.80 ± 2.93	48.89–61.11	55.03 ± 3.54	44.44–62.22	**+0.23**
Gemma-3-27B	21.80 ± 2.34	16.67–27.78	24.91 ± 2.30	20.00–30.00	**+3.11**
DAPO-Qwen-32B	61.16 ± 2.50	56.67–66.67	62.57 ± 2.49	57.78–68.89	**+1.41**

Table A5: Hyperparameter configuration by task. AIME and Count Down 4 use grid-search ranges; GPQA-Diamond and LiveCodeBench share a single universal setting.

Hyperparameter	AIME	Count Down 4	GPQA-D	LiveCodeBench
β	$\{\frac{1}{4}, \frac{1}{2}, 1, 2, 4, 8\}$	$\{\frac{1}{4}, \frac{1}{2}, 1, 2, 4, 8\}$	1	1
Top-p	$\{0.40, 0.60, 0.80, 0.95\}$	$\{0.40, 0.60, 0.80, 0.95\}$	0.95	0.95
Temperature T	$\{0.6, 0.8, 1.0\}$	$\{0.6, 0.8, 1.0\}$	0.6	0.6
Max generation length	32,768	8,192	16,384	16,384
Chat template	Default Templates	Default Templates	No Chat Templates[†]	Default Templates

[†]Except for Gemma-3-27B, the performance degradation is significant without chat template.

E Additional Setups

E.1 Best-of-N Analysis Setup

To measure how quickly a limited tuning budget yields performance gains, we simulate a *best-of-N random search* for $N = 1, \ldots, 15$. For every model-task pair in AIME and Count Down 4 we start from the complete Cartesian grid of hyperparameters in Table 4. At each Monte-Carlo replicate we uniformly draw N distinct values for a single target hyperparameter (β, top-p, or temperature) while keeping the other two at their default settings, retrieve the corresponding validation accuracies that were pre-computed during the main grid search, and record the improvement of the best sampled configuration over the initial draw. Repeating this procedure 256 times and averaging across the four LLMs produces the curves in Fig. 2.

E.2 Random-Forest Regression Analysis Setup

Every completed grid-search run — defined by a specific model, task, and random seed—serves as one training example for a model-agnostic importance analysis. Each example is encoded by the triple $(\beta, \text{top-}p, T)$ and labeled with its accuracy. We fit a `RandomForestRegressor` from SCIKIT-LEARN [44] with 100 trees that have unrestricted depth. Impurity-based Gini importances rank the hyperparameters as β (0.41), top-p (0.32), and temperature (0.27), as shown in Fig. 2.

E.3 Setup for Case Study

Section 7.2 investigates linear prompt blending on five sentiment benchmarks with three 7B-sized LLMs: Llama3 8B [32], Mistral 7B [41] and Gemma 1.1 7B [42].

For each dataset, we sampled from GPT4o [45] and Claude [46] to curate a diverse pool of task prompts—96 for binary sentiment, 32 for six-class emotion, and 40 for financial sentiment—verifying that each prompt forms a syntactically valid query when concatenated with the input sentence. Letting $L_{\max}$ denote the length of the longest prompt in a pool, every prompt-embedding matrix $\mathbf{E} \in \mathbb{R}^{L \times d}$ is extended to length $L_{\max}$ via linear interpolation, and then combined by simple averaging across prompts to form the blended prompt.

During inference, we prepend this blended vector to each of the 10-shot demonstrations and feed the embeddings directly to the model. For the single-prompt baseline we calculate the average accuracy of the prompts as the expectation of randomly choosing any single prompt, whereas the blended prompt is evaluated once; consequently, Table 2 reflects an identical number of forward passes per model. Because all other factors remain fixed, any performance difference isolates the effect of the prompt representation itself.

F Hyperparameters

Table A5 lists the full search space for AIME and COUNT DOWN 4, along with the universal settings used for GPQA-DIAMOND and LIVECODEBENCH. All searches use five random seeds; reported results are seed-averaged.

G Implementation Details

We implement MoI on top of the vLLM framework [47], which supports efficient tensor parallelism. Mixing weights are computed from both the output token and the associated logits after each generation step. The resulting mixed inputs are cached and used as input for the subsequent decoding step.

For GPQA evaluations, we use the Language Model Evaluation Harness framework [48]. All models are evaluated using the default configuration with "thinking mode" enabled. The only exception is Gemma-3-27B, which requires an additional prompt to elicit multiple-choice outputs in the format of (A, B, C, D).

LiveCodeBench evaluations follow its official implementation [28], using the default setup and test suite corresponding to the period from May 2023 to May 2024. At the time of writing, generation templates were not officially available for Llama-3.3-Nemotron-Super-49B, Gemma-3-27B, and DAPO-Qwen-32B. We manually created a template for Llama-3.3-Nemotron-Super-49B based on its official documentation, including the required system prompt to activate its thinking mode. For Gemma-3-27B and DAPO-Qwen-32B, we adopt the `GenericBase` and `CodeQwenInstruct` templates, respectively.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We propose MoI, our main claim is using both the sampled output and the distribution can boost language models' generation capabilities. It is accurately reflected in the title and introduction.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We have a limitation section in Section 8.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: We provided derivations and model formulations in Section 4.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We include the hyperparameter settings in Appendix F, details on implementations in Appendix G, that are sufficient for reproducing our results.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We provide our implementations along with our submission.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We discuss the hyperparameters in Section 5.4 and provide detailed table in Appendix F.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We provide the error bar analysis in Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We include a runtime analysis in Section 7.3.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We conducted our research conforming the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [NA]

    Justification: We explore a technique that has no direct relationship or impact on society.

    Guidelines:

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

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

11. **Safeguards**

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

    Answer: [NA]

    Justification: We have no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We have provided appropriate citations to code, data and models used in our research.

    Guidelines:

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

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

13. **New assets**

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

    Answer: [Yes]

    Justification: We have provided a readme in our implementation.

    Guidelines:

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

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

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

    Answer: [NA]

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

    Guidelines:

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

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

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

    Answer: [NA]

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

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

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