NAVIGATING SOLUTION SPACES IN LARGE LANGUAGE MODELS THROUGH CONTROLLED EMBEDDING EXPLORATION

Qinglin Zhu[1*] **Runcong Zhao**[1*] **Hanqi Yan**[1] **Yulan He**[1,2] **Yudong Chen**[3†] **Lin Gui**[1†]

[1]King's College London [2]The Alan Turing Institute [3]University of Warwick
{qinglin.1.zhu, runcong.zhao, hanqi.yan, yulan.he}@kcl.ac.uk
yudong.chen@warwick.ac.uk, lin.1.gui@kcl.ac.uk

ABSTRACT

Large Language Models (LLMs) struggle with reasoning due to limited diversity and inefficient search. We propose an embedding-based search framework that optimises the embedding of the first token to guide generation. It combines (1) Embedding perturbation for controlled exploration and (2) Bayesian optimisation to refine embeddings via a verifier-guided objective, balancing exploration and exploitation. This approach improves reasoning accuracy and coherence while avoiding reliance on heuristic search. Experiments demonstrate superior correctness with minimal computation, making it a scalable, model-agnostic solution.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable potential in various reasoning tasks. Despite these advancements, they still face significant limitations in complex reasoning tasks Lightman et al. (2024); Wang et al. (2023a). Many existing approaches typically increase generation diversity through multiple sampling (Lightman et al., 2024), often controlled by temperature scaling, which adjusts the randomness of token selection Brown et al. (2024). Planning-based methods, such as chain-of-thought reasoning Wei et al. (2024); Wang et al. (2023a) or tree-structured search Yao et al. (2023), attempt to locate the correct answer by following language-based instructions.

Despite these efforts, two key challenges remain: (1) Enhancing generation diversity typically relies on increasing the temperature parameter, which flattens the token distribution. This, however, does not necessarily result in better coverage of the correct answer, as increasing low-probability token likelihood indiscriminately may introduce noise rather than meaningful exploration Holtzman et al. (2020). (2) Existing planning and search methods such as sampling multiple reasoning paths rely heavily on heuristic strategies, guided by prompts (Hao et al., 2023; Qi et al., 2024). However, these approaches do not directly adjust for the model's internal representations, thereby making the search process inefficient and highly dependent on surface-level prompt variations. This often leads to a "wild-goose chase", where search remains constrained by randomness and indirect heuristics rather than systematic optimisation.

To address these challenges, we propose a novel approach using controlled embedding exploration: (1) By injecting a Gaussian embedding into the decoding of the first answer token, we can adjust the distribution of low-probability tokens in a more controlled manner than uniform temperature tuning, leading to more flexible generation. (2) Treating the LLM as a black box verifier, we apply Bayesian optimisation (Frazier, 2018) on the injected embedding to maximise a verification-based reward. This allows us to use observerd rewards to directly guide the exploration in the embedding space. As a result, our method improves performance without requiring a strong verifier—even when both generation and verification originate from the same model. As illustrated in Figure 1, the proposed method offers several advantages: (1) By employing a more flexible strategy for answer generation, the model enables broader answer coverage within the candidate set. (2) In the search phase, instead of relying on language-instruction-based searching or planning, our method directly

*Equal Contribution, †Corresponding author.

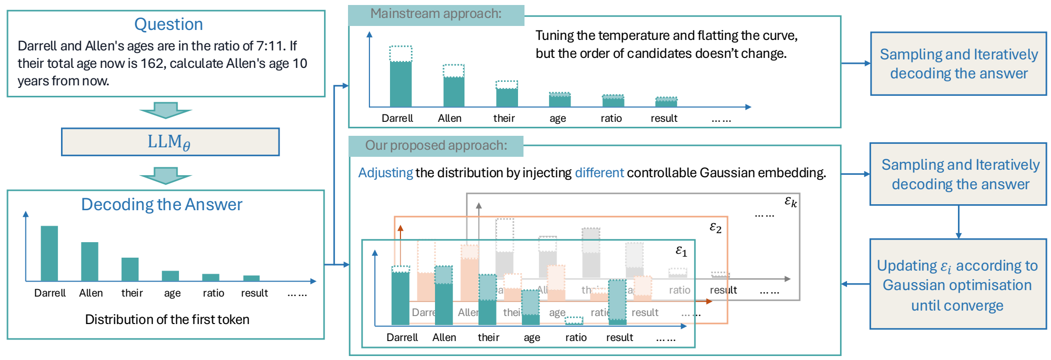


Figure 1: Comparison of Mainstream and Proposed Approaches.

optimises the embedding of the first token using Bayesian optimisation. This optimisation operates in the embedding space rather than the discrete token space and refines the model's reasoning trajectory while modifying only a single additional token in the decoding process. Moreover, since it does not require access to the model's parameters, it serves as an **off-the-shelf solution** that can be easily applied to most mainstream LLMs for reasoning tasks.

Our contributions can be summarised as follows:

- By injecting a Gaussian random vector into the decoding of the first answer token, the method provides a more controlled way to adjust the distribution of low-probability tokens, leading to more flexible and diverse answer generation than standard temperature tuning.
- Instead of language-based instruction or heuristic search, our method directly optimises the embedding of the first token, effectively reducing search complexity to the modification of a single additional token in the decoding process, improving efficiency and accuracy.
- Our framework treats the LLM as a black box, enhancing logical consistency and answer quality without accessing model parameters, allowing seamless integration into mainstream LLMs. Experiments show that it outperforms traditional decoding in correctness while being computationally efficient.

2 Related Work

2.1 Decoding Strategies and Diversity

Recent advances in LLM decoding aim to enhance diversity for tasks requiring creativity and exploration. Traditional methods such as greedy and beam search often produce repetitive outputs Holtzman et al. (2020); Welleck et al. (2019), while sampling-based approaches (top-k, nucleus) introduce randomness but struggle to balance quality and diversity Fan et al. (2018); Holtzman et al. (2020). High-temperature settings can lead to incoherent outputs Nguyen et al. (2024), and adaptive methods like min-p sampling Nguyen et al. (2024) require careful tuning. Debiasing-Diversifying Decoding (D3) mitigates amplification bias but increases computational cost Bao et al. (2024). Crucially, most methods overlook the impact of initial token selection, which significantly influences reasoning outcomes Wang & Zhou (2024). Our approach addresses this by perturbing initial token embeddings with Gaussian noise, reshaping the probability distribution to improve exploration while maintaining quality and efficiency.

2.2 Efficient Exploration of Solution Spaces

Efficient solution space exploration is crucial for enhancing LLM reasoning while maintaining practical computational costs. Increasing generated samples improves coverage Brown et al. (2024) but is computationally prohibitive. Optimising test-time compute allocation is more effective than scaling model size Snell et al. (2024), though it requires task-specific strategies. Mutual reasoning frameworks leveraging self-play and MCTS Qi et al. (2024); Yan et al. (2024), as well as Tree of Thoughts (ToT) Yao et al. (2023), explore multiple reasoning paths but incur high computational overhead. Thought Space Explorer (TSE) Zhang et al. (2024a) enhances reasoning breadth but at

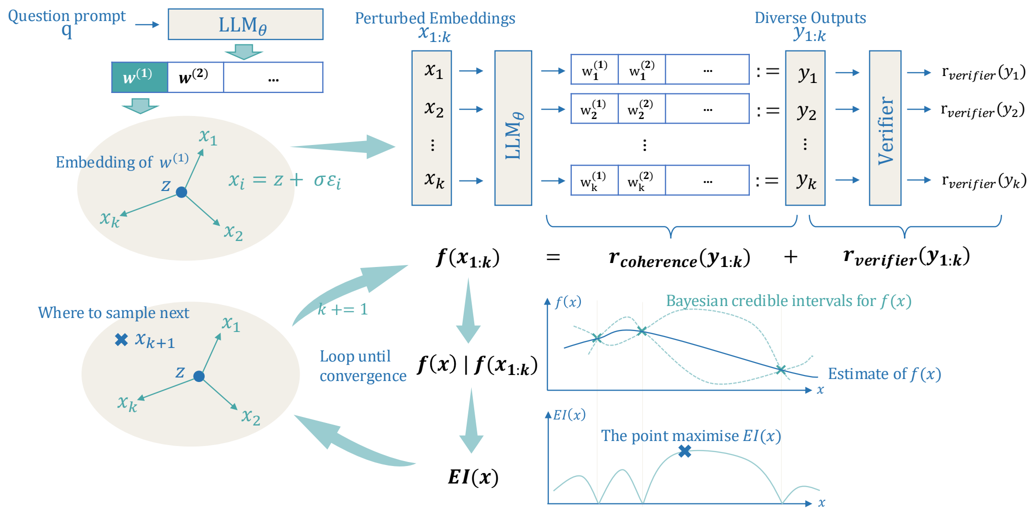


Figure 2: **Overview of Our Approach**. Starting with a natural language question prompt, the model generates initial token embeddings $w^{(1)}$, which, due to greedy decoding, determine the entire output. These embeddings are perturbed to create candidate embeddings $x_{1:k}$, leading to outputs $y_{1:k}$ through greedy search, which are then evaluated for coherence and verifier feedback. A Bayesian optimisation framework updates its estimation of the space based on this feedback and selects the next sampling point that maximises the expected improvement, balancing exploration and exploitation to refine the search for high-quality outputs.

additional cost. Our method refines these approaches by integrating controlled initial-token embedding perturbations with a strategic search algorithm inspired by MCTS and mutual reasoning. By introducing exploration early through embedding perturbation and guiding search via a verifier, we improve efficiency without excessive computational overhead, striking a balance between exploration and exploitation to optimise reasoning performance.

3 PRELIMINARY: TEMPERATURE SCALING

A common approach for generating diverse outputs is temperature scaling, which controls the randomness in the token generation process by modifying the softmax distribution over the model's output logits. For a given temperature $\tau > 0$, the probability of selecting token $w^{(t)}$ at time step t is given by:

$$P(w^{(t)} \mid w^{(1:t-1)}; \theta, \tau) = \frac{\exp(\ell_{t,w^{(t)}}/\tau)}{\sum_w \exp(\ell_{t,w}/\tau)},$$

where $w^{(1:t-1)}$ represents the sequence of tokens $\{w^{(1)}, \ldots, w^{(t-1)}\}$ generated from the first token up to the $(t-1)^{th}$ token, $\ell_{t,w}$ denotes the logit at time t corresponding to token w, and τ controls the sharpness of the distribution. This scaling flattens the distribution but preserves the relative ranking of token probabilities. When τ is low, the results concentrate on a few high-probability tokens, leading to overly deterministic generations with limited diversity. When τ is high, the model may sample low-probability tokens, leading to incoherent outputs.

While this approach increases diversity, it lacks control, blindly flattening token probabilities; adaptability, as it ignores verifier feedback; and efficiency, often requiring multiple samples or retraining (Joy et al., 2023; Xie et al., 2024). These limitations make it ineffective for structured reasoning tasks that demand precise and efficient exploration.

4 METHODOLOGY

Generating accurate answers in complex tasks requires both exploring reasoning paths and verifying for their correctness. To achieve this, we propose a two-step framework as shown in Figure 2: (1) Embedding perturbation applies a Gaussian adjustment to the first-token embedding for controlled modifications beyond uniform tuning. (2) Bayesian optimisation refines the perturbed embedding to maximise a verifier-guided reward, improving reasoning path selection.

4.1 Embedding Perturbation

Given a generative model g_θ and a natural language question prompt q, the first token $w^{(1)}$ is generated using greedy decoding, which selects the token with the highest probability from the model's predicted distribution:

$$w^{(1)} = \arg\max_{w} P(w \mid q; \theta),$$

where $P(\cdot)$ represents the probability distribution over the possible tokens predicted by the model g_θ.

Let $z \in \mathbb{R}^D$ represent the embedding of the token $w^{(1)}$. This embedding serves as a prior, representing a "correct starting point" in the latent space. To explore the neighbourhood of this embedding, we define a set of perturbed embeddings x_i for $i = 1, \dots, k$ as follows:

$$x_i = z + \sigma\varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, I),$$

where ε_i represents random perturbations drawn from a standard normal distribution, and σ is a scaling factor controlling the magnitude of the perturbation. This formulation allows us to sample from the local vicinity of the original embedding z, exploring variations around the initial token representation.

For each perturbed embedding x_i, we treat it as a new special token and add it to the current vocabulary. This special token is then used as the first token for generating an answer. Since we use **greedy decoding**, the choice of x_i **fully determines** the entire output sequence. The remaining tokens $w^{(2)}, w^{(3)}, \dots, w^{(L)}$ are deterministically generated based on x_i:

$$w^{(t)} = \arg\max_{w} P(w \mid x_i, w^{(2:t-1)}, q; \theta).$$

We repeat this process k times, generating k different answers, where $y = w^{(1:L)}$ represents the complete output. Because the output is fully determined by x_i, embedding perturbation effectively serves as a sampling mechanism over the entire answer space.

4.2 Exploring the Embedding Space

While randomly sampling points with infinite computational resources could theoretically approximate the optimal solution, this approach is highly inefficient, especially given the computational expense of sampling with an LLM. Instead, we adopt Bayesian optimisation, which consists of two key components: an *objective function* and an *acquisition function* that determine where to sample next. We use Expected Improvement (EI) as our acquisition function, which offers a closed-form solution (Frazier, 2018) with negligible computational cost, making it significantly more efficient by comparison. EI effectively balances exploration (searching uncertain regions) and exploitation (refining promising areas), selecting the point with the highest EI at each iteration to guide the optimisation process toward convergence.

Optimisation Objective. To evaluate the objective function with k sampled perturbed embeddings, we consider the sequence $x_{1:k} = \{x_1, \dots, x_k\}$, where each $x_i \in \mathbb{R}^D$. The corresponding answers are then generated as described above: $y_{1:k} = \{y_1, \dots, y_k\}$. Comparing and refining multiple generated answers has been shown to improve performance (Miao et al., 2024). Additionally, since LLMs are primarily trained for text generation rather than explicit judgment, prompting them to regenerate and compare outputs can yield better results (Zhang et al., 2024b). Building on these insights, we propose a *verifier-guided approach*, where the model evaluates a batch of candidate answers and produces a refined output $y_v = \mathcal{V}(y_{1:k})$. The correctness of an answer y is then assessed as a binary indicator (0 or 1), based on its alignment with the verifier's final output.

The embedding space may not be uniform, implying that perturbations in different directions can lead to *uneven semantic shifts* (Li et al., 2023; Park et al., 2024). In some dimensions, even small perturbations can significantly alter meaning, potentially disrupting grammar or context consistency and leading to incoherent outputs. To address this issue, we introduce the coherence term to prune low-quality generations, ensuring that only outputs with desirable semantic and syntactic properties are retained. To evaluate the quality of a generated output y, we define an objective function $f(x)$ that balances correctness and fluency:

$$f(x) = r_{\text{verifier}}(y) + r_{\text{coherence}}(y), \tag{1}$$

where:

- **Verifier Score (r_{verifier}):** This is a binary indicator provided by the verifier, reflecting the correctness of y:
$$r_{\text{verifier}}(y) = \mathbb{1}_{\{y_v = y\}};$$
- **Coherence ($r_{\text{coherence}}$):** This term evaluates the fluency of the generated sequence based on token probabilities:
$$r_{\text{coherence}}(y) = \sum_{i=1}^{T} \log P(w^{(i)}),$$
where $P(w^{(i)})$ is the probability of generating token $w^{(i)}$ from the LLM's entire vocabulary.

Bayesian Optimisation. Our goal is to maximise $f(\cdot)$, as defined in equation 1, over the embedding space $\mathbb{R}^D$. To optimise this black-box function, Bayesian optimisation uses a prior distribution on the domain to represent our beliefs about the behavior of the function and iteratively updates this prior using newly acquired data. Specifically, we model the prior joint distribution as a multivariate Gaussian distribution:

$$f(x_{1:n}) \sim \mathcal{N}\big(\mu_0(x_{1:n}), \Sigma_0(x_{1:n}, x_{1:n})\big),$$

where $\mu_0(x_{1:n})$ is the prior mean vector, and $\Sigma_0(x_{1:n}, x_{1:n})$ is the prior covariance matrix.

After observing $f(x_{1:k})$, we aim to infer the value of $f(x)$ at a new point x. Using Bayes' rule (Rasmussen & Williams, 2006), we update the posterior distribution of $f(x)$ conditioned on these observed values:

$$f(x) \mid f(x_{1:k}) \sim \mathcal{N}(\mu_k(x), \sigma_k^2(x)). \tag{2}$$

Here, $\mu_k(x)$ and $\sigma_k^2(x)$ represent the posterior mean and variance, respectively. A detailed discussion on the choice of the prior distribution and the computation of the posterior distribution is provided in Appendix B.2.

A naive way to find the maximiser at this stage would be to select among the previously evaluated points $x_1, \ldots, x_k$ the one with the highest observed function value. Let $f_k^* := \max_{m \leq k} f(x_m)$ denote this value. If we were to sample another point $x \in \mathbb{R}^D$ and observe $f(x)$, then the value of the best observed point would either be $f(x)$ (if $f(x) \geq f_k^*$) or f_k^* (if $f(x) < f_k^*$). The improvement in the value of the best observed point could be expressed as $[f(x) - f_k^*]^+ := \max(f(x) - f_k^*, 0)$.

While we would ideally choose x to maximise this improvement, $f(x)$ is unknown until after the evaluation. Instead, we select x that maximises the expected improvement under the posterior distribution, defined as

$$\text{EI}_k(x) := \mathbb{E}_k\big[[f(x) - f_k^*]^+\big], \tag{3}$$

where $\mathbb{E}_k$ denotes the expectation taken with respect to the posterior distribution equation 2. Using integration by parts, we can write EI equation 3 in a closed-form expression:

$$\begin{aligned}\text{EI}_k(x) = \big[\mu_k(x) - f_k^*\big]^+ &+ \sigma_k(x)\phi\left(\frac{\mu_k(x) - f_k^*}{\sigma_k(x)}\right) \\ &- \big|\mu_k(x) - f_k^*\big|\Phi\left(\frac{\mu_k(x) - f_k^*}{\sigma_k(x)}\right),\end{aligned}$$

where ϕ and Φ denote the probability density function and the cumulative distribution function of the standard normal distribution, respectively.

Our next sampling point, $x_{k+1} \in \mathbb{R}^D$, is the maximiser of EI. We then iteratively update the posterior distribution and the EI function. Details on how we select x to maximise $\text{EI}_k(x)$ can be found in Appendix B.3. Convergence is considered achieved when the change in the objective function between consecutive iterations satisfies $|f_k - f_{k-1}| < \epsilon$, where ϵ is a predefined threshold. Additionally, the algorithm terminates after a maximum of K iterations if convergence has not been reached.

In defining $f(x)$, we assume an ideal verifier with perfect accuracy, meaning it provides an error-free assessment of correctness. However, in practice, the verifier's accuracy is less than 1, introducing uncertainty into its evaluations. To address this noise in Bayesian optimisation, we use an adaptive version of the EI acquisition function that explicitly incorporates observation uncertainty. This adaptation dynamically adjusts the exploration rate based on uncertainty, ensuring a higher probability of convergence while balancing exploration and exploitation (Vakili et al., 2021; Tran-The et al., 2022). Theoretical foundations and implementation details are provided in Appendix B.4.

Dimension Reduction. One shortcoming of using traditional Bayesian optimisation methods for identifying the point with maximum EI (Mockus, 1975; Hvarfner et al., 2024) is that they perform poorly when the search space exceeds 20–30 dimensions due to the curse of dimensionality (Kandasamy et al., 2015; Letham et al., 2020; Wang et al., 2023b). In high-dimensional spaces, surrogate models require an exponentially larger number of points to accurately estimate the maximum of the EI function, making optimisation highly inefficient. With the dimension of embedding vectors for LLMs typically ranging from 768 to 8192 or more, traditional methods are impractical in our setting.

To address this, we leverage a dimension reduction approach based on random embeddings (Wang et al., 2016; Nayebi et al., 2019). Specifically, if a function $f : \mathbb{R}^D \to \mathbb{R}$ has an effective dimension $d_e \leq D$, then with high probability, there exists a lower-dimensional representation $g(u) := f(Au)$, where A is a random projection matrix. This allows optimisation to be performed in a lower-dimensional space $\mathbb{R}^d$ instead of the original $\mathbb{R}^D$. Using this approach, we iteratively optimise the function in the reduced space and map solutions back to the original space. Theoretical foundations and implementation details are provided in Appendix B.5.

5 EXPERIMENTS

We benchmark our method against strong baselines and conduct ablation studies.

5.1 EXPERIMENTAL SETUP

Datasets and Models. We conduct experiments using three LLMs: Llama-3.1-8B-Instruct (Meta, 2024), Qwen2-7B-Instruct (Yang et al., 2024), and Mistral-8B-Instruct (Jiang et al., 2023). The models are evaluated on four benchmark datasets, including three complex mathematical tasks (GSM8K (Cobbe et al., 2021), GSM-Hard (Gao et al., 2022), SVAMP (Patel et al., 2021)), and one commonsense reasoning task StrategyQA (Geva et al., 2021).

Baselines. Our baselines include: (1) CoT Prompting, which includes zero-shot CoT (Kojima et al., 2022) and Few-Shot CoT (Wei et al., 2022); (2) Self-Consistency Decoding (Wang et al., 2023c), which involves sampling answers at various temperatures $\tau \in \{0.4, 0.6, 0.8\}$ and selecting the final answer through majority voting; (3) FIRE (Chen et al., 2024), which adjusts the decoding process by setting the temperature of the first token to 30 to enhance diversity, while subsequent tokens are generated using the standard temperature setting; (4) CoT-Decoding (Wang & Zhou, 2024), which generates k answers by sampling the top-k tokens from the probability distribution of the first token. Each of these top-k tokens is used as the starting point for decoding the remainder of the answer; and (5) RAP (Hao et al., 2023), which uses Monte Carlo Tree Search to explore reasoning paths strategically, balancing exploration and exploitation to find solutions efficiently. Note that RAP requires problem decomposition via examples; hence we only report its performance in the few-shot setting.

Setup & Hyperparameters. Experiments are conducted in zero-shot and few-shot settings, with prompts including 1, 2, 4, and 8 exemplars for few-shot settings. To reduce variance, each configuration is repeated five times with different random seeds. We report the mean and standard deviation of accuracy across all runs. The convergence threshold is set to 0.01.

Table 1 presents the accuracy of our method compared to baselines across four benchmarks and three LLMs under zero-shot and few-shot (8-shot) settings. The full table can be found in Table 6 in

Table 1: Performances of different reasoning methods on Accuracy (%) across all benchmarks.

Model	Method	GSM8K		GSM-Hard		SVAMP		StrategyQA	
		Zero Shot	Few Shot	Zero Shot	Few Shot	Zero Shot	Few Shot	Zero Shot	Few Shot
LLaMA3-8B-Ins	COT	53.0$_{\pm 0.0}$	77.4$_{\pm 0.0}$	14.0$_{\pm 0.0}$	28.0$_{\pm 0.0}$	61.0$_{\pm 0.0}$	83.0$_{\pm 0.0}$	58.5$_{\pm 0.0}$	68.5$_{\pm 0.0}$
	SC($\tau = 0.4$)	73.0$_{\pm 1.6}$	80.4$_{\pm 1.4}$	25.7$_{\pm 0.4}$	31.8$_{\pm 1.8}$	79.1$_{\pm 1.2}$	87.1$_{\pm 1.0}$	64.7$_{\pm 0.7}$	71.6$_{\pm 0.8}$
	SC($\tau = 0.6$)	73.6$_{\pm 2.5}$	80.6$_{\pm 1.5}$	24.5$_{\pm 1.1}$	31.2$_{\pm 1.3}$	76.1$_{\pm 3.9}$	87.7$_{\pm 1.2}$	59.9$_{\pm 2.0}$	71.3$_{\pm 1.5}$
	SC($\tau = 0.8$)	65.0$_{\pm 2.0}$	81.1$_{\pm 1.1}$	21.8$_{\pm 1.3}$	30.8$_{\pm 0.9}$	69.6$_{\pm 2.0}$	87.4$_{\pm 1.2}$	54.4$_{\pm 2.6}$	72.7$_{\pm 1.2}$
	FIRE	73.8$_{\pm 2.3}$	79.6$_{\pm 2.9}$	25.2$_{\pm 3.0}$	25.7$_{\pm 2.1}$	81.5$_{\pm 0.8}$	87.6$_{\pm 2.0}$	63.0$_{\pm 3.7}$	72.8$_{\pm 1.5}$
	CoT-Decoding	73.9$_{\pm 1.9}$	80.3$_{\pm 1.7}$	24.8$_{\pm 1.3}$	30.3$_{\pm 1.3}$	83.2$_{\pm 1.2}$	88.2$_{\pm 1.0}$	64.6$_{\pm 1.6}$	73.3$_{\pm 1.8}$
	RAP	-	80.7$_{\pm 1.4}$	-	32.7$_{\pm 1.2}$	-	87.9$_{\pm 1.1}$	-	73.4$_{\pm 1.1}$
	Ours	**79.4**$_{\pm 1.2}$	**84.3**$_{\pm 1.4}$	**28.2**$_{\pm 1.8}$	**35.7**$_{\pm 1.0}$	**88.2**$_{\pm 1.3}$	**90.2**$_{\pm 0.6}$	**67.2**$_{\pm 0.7}$	**75.6**$_{\pm 0.8}$
	w/o r_{verifier}	76.8$_{\pm 1.0}$	82.0$_{\pm 0.5}$	26.3$_{\pm 1.3}$	34.8$_{\pm 0.3}$	86.7$_{\pm 1.2}$	89.5$_{\pm 0.5}$	66.2$_{\pm 2.8}$	74.3$_{\pm 1.6}$
	w/o $r_{\text{coherence}}$	77.4$_{\pm 2.1}$	83.4$_{\pm 0.7}$	27.9$_{\pm 1.5}$	35.3$_{\pm 1.3}$	84.6$_{\pm 2.4}$	90.1$_{\pm 0.9}$	66.0$_{\pm 1.3}$	75.0$_{\pm 1.5}$
Qwen2-7B-Ins	COT	64.5$_{\pm 0.0}$	82.5$_{\pm 0.0}$	40.0$_{\pm 0.0}$	55.5$_{\pm 0.0}$	43.5$_{\pm 0.0}$	86.0$_{\pm 0.0}$	63.0$_{\pm 0.0}$	70.0$_{\pm 0.0}$
	SC($\tau = 0.4$)	81.2$_{\pm 0.6}$	85.7$_{\pm 1.5}$	47.5$_{\pm 1.4}$	55.4$_{\pm 0.7}$	72.3$_{\pm 2.0}$	90.3$_{\pm 1.2}$	67.1$_{\pm 1.5}$	71.1$_{\pm 1.6}$
	SC($\tau = 0.6$)	80.2$_{\pm 1.9}$	85.4$_{\pm 0.9}$	46.2$_{\pm 1.9}$	53.4$_{\pm 0.6}$	77.3$_{\pm 1.2}$	90.4$_{\pm 0.6}$	67.5$_{\pm 0.7}$	69.1$_{\pm 1.2}$
	SC($\tau = 0.8$)	80.0$_{\pm 0.9}$	85.1$_{\pm 1.6}$	47.3$_{\pm 1.3}$	55.4$_{\pm 0.9}$	78.6$_{\pm 2.1}$	90.6$_{\pm 1.2}$	67.0$_{\pm 1.0}$	70.1$_{\pm 0.8}$
	FIRE	81.0$_{\pm 1.8}$	83.0$_{\pm 1.3}$	45.1$_{\pm 2.0}$	51.0$_{\pm 1.8}$	76.3$_{\pm 2.2}$	90.6$_{\pm 0.2}$	67.6$_{\pm 0.8}$	68.1$_{\pm 0.8}$
	CoT-Decoding	82.0$_{\pm 2.8}$	84.5$_{\pm 2.1}$	46.7$_{\pm 2.3}$	52.1$_{\pm 1.0}$	78.6$_{\pm 1.6}$	89.7$_{\pm 0.5}$	65.9$_{\pm 1.5}$	69.5$_{\pm 2.1}$
	RAP	-	86.2$_{\pm 1.2}$	-	56.2$_{\pm 0.8}$	-	90.8$_{\pm 1.1}$	-	**71.3**$_{\pm 1.3}$
	Ours	**88.6**$_{\pm 1.2}$	**90.0**$_{\pm 1.4}$	**53.7**$_{\pm 1.6}$	**58.7**$_{\pm 0.5}$	**83.4**$_{\pm 2.4}$	**92.2**$_{\pm 0.8}$	**68.1**$_{\pm 1.5}$	70.3$_{\pm 1.3}$
	w/o r_{verifier}	87.0$_{\pm 1.0}$	89.7$_{\pm 1.8}$	51.2$_{\pm 2.1}$	58.3$_{\pm 1.0}$	73.5$_{\pm 4.0}$	90.5$_{\pm 1.5}$	66.0$_{\pm 0.9}$	68.3$_{\pm 1.6}$
	w/o $r_{\text{coherence}}$	87.3$_{\pm 2.0}$	89.2$_{\pm 1.3}$	52.0$_{\pm 1.5}$	60.0$_{\pm 1.0}$	76.7$_{\pm 1.4}$	90.7$_{\pm 0.6}$	66.5$_{\pm 0.5}$	69.5$_{\pm 1.5}$
Mistral-7B-Ins	COT	42.0$_{\pm 0.0}$	54.0$_{\pm 0.0}$	14.5$_{\pm 0.0}$	24.0$_{\pm 0.0}$	52.0$_{\pm 0.0}$	72.0$_{\pm 0.0}$	62.0$_{\pm 0.0}$	69.0$_{\pm 0.0}$
	SC($\tau = 0.4$)	52.9$_{\pm 0.5}$	58.3$_{\pm 1.5}$	19.5$_{\pm 1.0}$	26.1$_{\pm 1.5}$	67.4$_{\pm 2.5}$	77.8$_{\pm 1.0}$	63.9$_{\pm 1.5}$	72.6$_{\pm 1.2}$
	SC($\tau = 0.6$)	55.1$_{\pm 3.6}$	57.4$_{\pm 1.0}$	20.7$_{\pm 1.5}$	25.3$_{\pm 1.6}$	69.7$_{\pm 1.6}$	78.4$_{\pm 2.0}$	64.2$_{\pm 1.0}$	71.7$_{\pm 0.8}$
	SC($\tau = 0.8$)	50.2$_{\pm 2.6}$	57.7$_{\pm 2.6}$	19.1$_{\pm 2.0}$	26.6$_{\pm 1.1}$	68.3$_{\pm 0.9}$	77.6$_{\pm 1.1}$	64.9$_{\pm 1.0}$	72.1$_{\pm 1.5}$
	FIRE	47.2$_{\pm 2.9}$	56.1$_{\pm 3.2}$	18.1$_{\pm 1.9}$	26.3$_{\pm 1.4}$	67.1$_{\pm 1.9}$	78.4$_{\pm 1.2}$	64.2$_{\pm 1.0}$	71.0$_{\pm 2.2}$
	CoT-Decoding	47.3$_{\pm 3.0}$	58.2$_{\pm 2.3}$	16.6$_{\pm 0.7}$	27.4$_{\pm 1.6}$	69.4$_{\pm 2.5}$	78.6$_{\pm 1.4}$	63.5$_{\pm 1.5}$	72.7$_{\pm 2.1}$
	RAP	-	58.6$_{\pm 1.8}$	-	27.6$_{\pm 1.2}$	-	79.4$_{\pm 1.1}$	-	72.4$_{\pm 1.3}$
	Ours	**61.4**$_{\pm 2.5}$	**62.7**$_{\pm 1.0}$	**25.8**$_{\pm 1.8}$	**32.5**$_{\pm 1.5}$	**72.2**$_{\pm 2.2}$	**82.1**$_{\pm 1.2}$	**66.1**$_{\pm 1.9}$	**72.8**$_{\pm 1.5}$
	w/o r_{verifier}	59.5$_{\pm 1.3}$	59.7$_{\pm 2.8}$	24.8$_{\pm 0.8}$	30.8$_{\pm 2.1}$	69.5$_{\pm 2.3}$	79.8$_{\pm 0.8}$	64.8$_{\pm 0.6}$	71.8$_{\pm 1.4}$
	w/o $r_{\text{coherence}}$	61.2$_{\pm 2.3}$	60.3$_{\pm 2.5}$	25.5$_{\pm 3.3}$	29.5$_{\pm 2.3}$	70.2$_{\pm 1.6}$	80.0$_{\pm 1.0}$	65.5$_{\pm 1.7}$	72.0$_{\pm 1.3}$

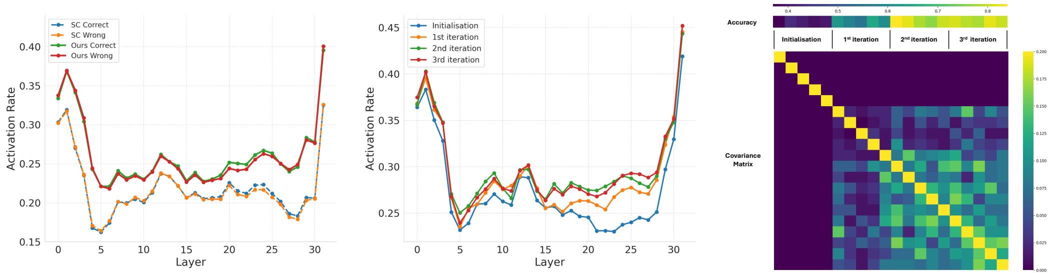
Appendix C.5. Our approach consistently outperforms the best-performing baseline across different models, especially in the zero-shot setting (average improvement of 5% on GSM8K and 3% on GSM-Hard). Similar gains appear in the few-shot setting, where our method achieves the highest accuracy on most tasks and model variants. While effective, SC requires extensive hyperparameter tuning (e.g. varying temperature values) for each individual model and dataset to achieve optimal performance. In contrast, our more systematic search method improves solution quality consistently without the need for separate tuning in each scenario.

Coverage Analysis. For each method, we calculate the probability of covering the correct answer in at least one of the generated answers. Our approach consistently achieves the highest coverage across all models and datasets. For instance, on GSM8K with LLaMA3-8B-Ins in the zero-shot setting, our method attains 91.8% coverage, outperforming FIRE (84.5%) and CoT-Decoding (85.3%). Detailed coverage probabilities for all models and datasets can be found in Table 7 in Appendix C.5. These results demonstrate that our controlled exploration strategy effectively enhances the likelihood of generating correct answers, highlighting its robustness over traditional methods.

Effect of Exploration with Embedding Perturbations and Bayesian Optimisation. A natural question to consider is why adding noise to embeddings leads to more diverse answer generation than temperature tuning. We follow Naik et al. (2024) to investigate this from the perspective of neuron activations in the Transformer's MLP layers. As shown in Figure 3a, applying our method increases the activation rate of neurons by roughly 3–4% in nearly all layers relative to the Self-Consistency (SC) baseline, suggesting that our perturbations stochastically trigger more diverse neural pathways.

To probe whether a specific subset of "critical neurons" may be responsible for correct reasoning, we identify neurons whose activations exhibit the strongest correlation with correctness. In Figure 3b, we track the activation rate of these critical neurons across our Bayesian optimisation iterations. We observe a steady increase, particularly in layers 15–30, suggesting that our iterative sampling and verification procedure increasingly activates these key pathways.

Together, these findings support the hypothesis that our embedding perturbation and controlled exploration approach not only diversifies generation but also systematically uncovers and reinforces



(a) MLP layer activation rates across Transformer layers for the first five tokens of generated answers, sampled 200 times.

(b) Activation rates of critical MLP neurons across Transformer layers during initialization and iterations.

(c) Visualization of the accuracy and covariance matrix evolution during Bayesian optimisation across iterations.

Figure 3: Transformer MLP activation and Bayesian optimisation convergence analysis.

the neuron activations crucial for deriving correct answers. For more detailed experimental procedures, please refer to Appendix C.2.

Convergence of Bayesian Optimisation. Another question regarding our search algorithm is how quickly and reliably it converges. To investigate this, we track two key metrics across our Bayesian optimisation iterations: (1) The evolution of the covariance matrix among the sampled embedding points. As shown in Figure 3c, the covariance matrix becomes more structured over iterations, showing higher correlations among top-performing candidates. (2) The proportion of test examples that terminate after the nth iteration for each dataset in both zero-shot and few-shot settings, as reported in Table 2. With a maximum of 4 iterations, no search exceeds the fourth iteration, and only a small fraction require iteration 4. This rapid termination suggests that the EI-driven sampling strategy quickly identifies promising regions of the embedding space for most queries, minimising the need for further rounds of exploration.

Table 2: Proportion (%) of test examples that terminate at the nth iteration for each dataset and setting using LLaMA.

Shot	Iteration	GSM8K	GSM-Hard	SVAMP	StrategyQA
Zero	1	$65.0_{\pm 2.9}$	$70.3_{\pm 3.0}$	$64.8_{\pm 1.9}$	$58.8_{\pm 2.3}$
	2	$30.1_{\pm 2.9}$	$24.4_{\pm 3.0}$	$28.9_{\pm 1.8}$	$31.1_{\pm 2.3}$
	3	$4.1_{\pm 1.6}$	$4.9_{\pm 1.6}$	$5.4_{\pm 1.3}$	$8.8_{\pm 1.8}$
	4	$0.8_{\pm 0.8}$	$0.4_{\pm 0.3}$	$0.9_{\pm 0.5}$	$1.4_{\pm 0.5}$
Few	1	$76.8_{\pm 2.2}$	$77.4_{\pm 2.2}$	$79.7_{\pm 2.0}$	$66.7_{\pm 3.7}$
	2	$20.7_{\pm 0.8}$	$20.6_{\pm 2.7}$	$19.1_{\pm 2.8}$	$26.6_{\pm 3.7}$
	3	$2.5_{\pm 1.4}$	$1.8_{\pm 0.6}$	$1.2_{\pm 1.0}$	$6.1_{\pm 2.1}$
	4	$0.0_{\pm 0.0}$	$0.2_{\pm 0.3}$	$0.0_{\pm 0.0}$	$0.6_{\pm 0.4}$

Efficiency and Performance Analysis. Table 3 presents a comparison of our approach with RAP in both performance and efficiency. Our method achieves better accuracy on all tasks while drastically reducing computational overhead. Specifically, our input token consumption averages only 6.19% of RAP's, and our output token usage is 63.28% of RAP's. These results highlight that our method not only improves accuracy but also substantially reduces token usage, thereby delivering superior overall efficiency.

Table 3: Comparison of performance and token usage between our method and RAP across benchmarks.

Category	Shot	GSM8K	GSM-Hard	SVAMP	StrategyQA
Result (%)	RAP	80.7	32.7	87.9	73.4
	Ours	**84.3**	**35.7**	**90.2**	**75.6**
Input Token Count	RAP	$25710.8k$	$33152.1k$	$15058.5k$	$17426.2k$
	Ours	$1457.1k$	$1847.8k$	$1172.8k$	$1180.6k$
Output Token Count	RAP	$334.1k$	$402.5k$	$241.2k$	$274.1k$
	Ours	$211.9k$	$262.5k$	$162.4k$	$155.4k$

5.2 Ablation Studies

Objective Function. To evaluate the importance of each reward component, we removed either the verifier score (**w/o** r_{verifier}) or the coherence term (**w/o** $r_{\text{coherence}}$) from our method. Tables 1 compare these ablated variants with our full approach. In all tasks and model configurations, omitting either term degrades both accuracy and coverage, indicating that both components are vital. The verifier score clearly helps filter out incorrect or spurious solutions, while the coherence penalty ensures each output remains semantically consistent, particularly for complex or multi-step reasoning. Indeed, both correctness-guided verification and semantic coherence play essential roles in navigating the solution space effectively.

Impact of Lower-Dimensional Space Dimensionality. We evaluate our Bayesian optimisation approach under various reduced dimensions before mapping back to the full embedding space (Figure 5). Across all four tasks and both zero- and few-shot settings, performance tends to improve up to $d = 50$. Although increasing d to 60 sometimes yields a small additional gain, the differences are minor, and $d = 50$ consistently achieves near-best or best results.

Impact of Special Token Placement. We compare three ways of inserting the perturbed special token into the prompt: at the beginning (*First*), somewhere in the middle (*Middle*), or as an appended token (*Last*). Table 4 shows for both zero-shot and few-shot settings, placing the special token at the end of the prompt (*Last*) generally yields higher accuracy and better coverage. One possible explanation is that placing the special token last ensures minimal disruption to the original semantics of the prompt, while still allowing our method to alter the initial token embedding and induce sufficiently diverse generation pathways.

Table 4: Comparison of accuracy and coverage across different special token placements.

Type	Shot	Iteration	GSM8K	GSM-Hard	SVAMP	StrategyQA
Result	Zero	First	$77.7_{\pm 2.5}$	$25.5_{\pm 1.3}$	$85.0_{\pm 0.9}$	$67.0_{\pm 1.3}$
		Middle	$78.5_{\pm 3.1}$	$27.3_{\pm 1.3}$	$84.0_{\pm 0.9}$	$\mathbf{67.8}_{\pm 4.4}$
		Last (ours)	$\mathbf{79.4}_{\pm 1.2}$	$\mathbf{28.2}_{\pm 1.8}$	$\mathbf{88.2}_{\pm 1.3}$	$67.2_{\pm 0.7}$
	Few	First	$82.1_{\pm 0.8}$	$29.0_{\pm 3.5}$	$89.2_{\pm 0.3}$	$74.1_{\pm 0.8}$
		Middle	$82.3_{\pm 0.8}$	$32.3_{\pm 1.8}$	$89.7_{\pm 1.3}$	$74.0_{\pm 3.0}$
		Last (ours)	$\mathbf{84.3}_{\pm 1.4}$	$\mathbf{35.7}_{\pm 1.0}$	$\mathbf{90.2}_{\pm 0.6}$	$\mathbf{75.6}_{\pm 0.8}$
Coverage	Zero	First	$85.8_{\pm 2.3}$	$31.2_{\pm 2.8}$	$93.0_{\pm 1.8}$	$92.7_{\pm 0.0}$
		Middle	$89.5_{\pm 2.6}$	$32.7_{\pm 0.8}$	$93.3_{\pm 1.0}$	$93.1_{\pm 0.3}$
		Last (ours)	$\mathbf{91.8}_{\pm 1.4}$	$\mathbf{37.0}_{\pm 1.5}$	$\mathbf{93.8}_{\pm 0.4}$	$\mathbf{93.7}_{\pm 1.3}$
	Few	First	$92.0_{\pm 0.5}$	$40.8_{\pm 1.2}$	$94.7_{\pm 1.2}$	$93.4_{\pm 0.9}$
		Middle	$92.2_{\pm 1.0}$	$44.7_{\pm 2.5}$	$94.5_{\pm 0.5}$	$93.1_{\pm 0.8}$
		Last (ours)	$\mathbf{92.2}_{\pm 0.8}$	$\mathbf{49.8}_{\pm 1.0}$	$\mathbf{95.8}_{\pm 1.2}$	$\mathbf{93.3}_{\pm 1.8}$

Verifier Comparison: Judgement vs. Generation. Inspired by recent work suggesting that LLMs can be more adept at *generating* correct outputs than critiquing existing ones (Miao et al., 2024; Zhang et al., 2024b), we explore four verifier strategies: **Single-Judge**, which evaluates each candidate independently; **Single-Generate**, which regenerates a purportedly correct answer for each candidate; **Multi-Judge**, which scores multiple candidates collectively; and **Multi-Generate**, which produces a new solution from multiple candidates, labeling any matching candidate as correct. The detailed definitions of these prompt templates are provided in Appendix C.3.

Table 5a reports the binary classification accuracies for each verifier. *Multi-Generate* yields the highest verification accuracy on all datasets. This indicates that leveraging the model's generative capabilities leads to more reliable correctness assessment.

Table 5: Combined results for verifier strategies: (a) Binary Classification Accuracy and (b) Final Accuracy in Overall Framework.

(a) Binary Classification Accuracy

	GSM8K	GSM-Hard	SVAMP	StrategyQA
Single-Judge	75.9	60.9	82.7	63.9
Multi-Judge	80.4	46.8	87.5	67.3
Single-Generate	78.0	40.7	82.7	71.7
Multi-Generate (ours)	**87.6**	**78.2**	**93.4**	**78.9**

(b) Final Accuracy in Overall Framework

Zero-shot

Verifier	GSM8K	GSM-Hard	SVAMP	StrategyQA
Single-Judge	$76.3_{\pm 1.5}$	$28.1_{\pm 1.6}$	$83.0_{\pm 1.4}$	$63.0_{\pm 1.7}$
Multi-Judge	$77.4_{\pm 2.2}$	$26.5_{\pm 2.2}$	$86.5_{\pm 1.3}$	$67.1_{\pm 0.8}$
Single-Generate	$76.5_{\pm 1.1}$	$27.6_{\pm 2.1}$	$84.3_{\pm 0.0}$	$66.4_{\pm 0.0}$
Multi-Generate	$\mathbf{79.4}_{\pm 1.2}$	$\mathbf{28.2}_{\pm 1.8}$	$\mathbf{88.2}_{\pm 1.3}$	$\mathbf{67.2}_{\pm 0.7}$

Few-shot

Verifier	GSM8K	GSM-Hard	SVAMP	StrategyQA
Single-Judge	$82.4_{\pm 1.5}$	$35.0_{\pm 1.4}$	$89.6_{\pm 1.4}$	$72.0_{\pm 1.3}$
Multi-Judge	$82.5_{\pm 1.3}$	$\mathbf{36.1}_{\pm 2.1}$	$90.1_{\pm 0.9}$	$73.2_{\pm 1.2}$
Single-Generate	$82.4_{\pm 1.3}$	$34.5_{\pm 1.4}$	$89.7_{\pm 1.2}$	$74.7_{\pm 1.2}$
Multi-Generate	$\mathbf{84.3}_{\pm 1.4}$	$35.7_{\pm 1.0}$	$\mathbf{90.2}_{\pm 0.6}$	$\mathbf{75.6}_{\pm 0.8}$

We compare final solution accuracy in Table 5b. While single verifiers judge or generate answers individually, multi-candidate generation leads to the highest end-to-end performance. *Multi-Generate* outperforms alternatives in both zero-shot and few-shot settings, harnessing the model's generative capacity more effectively than judgment-based verifiers. Notably, even substituting simpler verifiers keeps our framework competitive with strong baselines, underscoring the robustness and efficacy of generation-based verification. Details on how the number of sampled embeddings k influences performance are provided in Appendix C.4.

6 Conclusions and Future Directions

We introduce an embedding-based optimisation framework that enhances LLM reasoning by refining the first-token embedding. By integrating controlled perturbations with Bayesian optimisation, our method improves accuracy, is model-agnostic, and remains computationally efficient. Our approach relies on a verifier that may provide unreliable feedback, impacting optimisation. Additionally, it operates at the token level without clear interpretability of how perturbations influence reasoning. Future work will focus on improving verifier reliability, extending optimisation beyond the first token, and enhancing interpretability to better understand perturbation effects on reasoning.

ACKNOWLEDGMENTS

This work was supported in part by the UK Engi- neering and Physical Sciences Research Council (EPSRC) through a Turing AI Fellowship (grant no. EP/V020579/1, EP/V020579/2) and a New Horizons grant (grant no. EP/X019063/1), and In- novate UK through the Accelerating Trustworthy AI programme (grant no. 10093055). A PhD studentship from the Chinese Scholarship Council funds Qinglin Zhu. The authors also acknowledge the use of the King's Computational Research, Engineering, and Technology Environment (CREATE) at King's College London.

REFERENCES

Keqin Bao, Jizhi Zhang, Yang Zhang, Xinyue Huo, Chong Chen, and Fuli Feng. Decoding matters: Addressing amplification bias and homogeneity issue for llm-based recommendation, 2024. URL https://arxiv.org/abs/2406.14900.

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.

Weizhe Chen, Zhicheng Zhang, Guanlin Liu, Renjie Zheng, Wenlei Shi, Chen Dun, Zheng Wu, Xing Jin, and Lin Yan. Flaming-hot initiation with regular execution sampling for large language models. *arXiv preprint arXiv:2410.21236*, 2024.

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.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 889–898, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1082. URL https://aclanthology.org/P18-1082.

Peter I. Frazier. A tutorial on bayesian optimization, 2018. URL https://doi.org/10.48550/arXiv.1807.02811. arXiv preprint.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*, 2022.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics (TACL)*, 2021.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.507. URL https://aclanthology.org/2023.emnlp-main.507/.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Carl Hvarfner, Erik O. Hellsten, and Luigi Nardi. Vanilla bayesian optimization performs great in high dimensions. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235, Vienna, Austria, 2024. PMLR.

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.

Tom Joy, Francesco Pinto, Ser-Nam Lim, Philip H. S. Torr, and Puneet K. Dokania. Sample-dependent adaptive temperature scaling for improved calibration. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23)*, 2023.

Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional bayesian optimisation and bandits via additive models. In *Proceedings of the International Conference on Machine Learning*, pp. 295–304. PMLR, 2015.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Benjamin Letham, Roberto Calandra, Akshara Rai, and Eytan Bakshy. Re-examining linear embeddings for high-dimensional bayesian optimization. *Advances in Neural Information Processing Systems*, 33:1546–1558, 2020.

Linyang Li, Ke Ren, Yunfan Shao, Pengyu Wang, and Xipeng Qiu. Perturbscore: Connecting discrete and continuous perturbations in nlp. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 6638–6648, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.442. URL https://aclanthology.org/2023.findings-emnlp.442/.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Meta. Introducing meta llama3: The most capable openly available llm to date, 2024. URL https://ai.meta.com/blog/meta-llama-3/.

Ning Miao, Yee Whye Teh, and Tom Rainforth. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

J. Mockus. On bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pp. 400–404. Springer, 1975.

Ranjita Naik, Varun Chandrasekaran, Mert Yuksekgonul, Hamid Palangi, and Besmira Nushi. Diversity of thought improves reasoning abilities of llms. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Amin Nayebi, Alexander Munteanu, and Matthias Poloczek. A framework for bayesian optimization in embedded subspaces. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pp. 4752–4761. PMLR, 2019.

Minh Nguyen, Andrew Baker, Clement Neo, Allen Roush, Andreas Kirsch, and Ravid Shwartz-Ziv. Turning up the heat: Min-p sampling for creative and coherent llm outputs, 2024. URL https://arxiv.org/abs/2407.01082.

Jae Hyeon Park, Gyoomin Lee, Seunggi Park, and Sung In Cho. Not All Classes Stand on Same Embeddings: Calibrating a Semantic Distance with Metric Tensor. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. URL https://openreview.net/forum?id=zHUznVsBZp. Last modified: 13 Nov 2024.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. Mutual reasoning makes smaller llms stronger problem-solvers, 2024. URL https://arxiv.org/abs/2408.06195.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Hung Tran-The, Sunil Gupta, Santu Rana, and Svetha Venkatesh. Regret bounds for expected improvement algorithms in gaussian process bandit optimization. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 151, Valencia, Spain, 2022. PMLR.

Sattar Vakili, Kia Khezeli, and Victor Picheny. On information gain and regret bounds in gaussian process bandits. In Arindam Banerjee and Kenji Fukumizu (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130, pp. 82–90. PMLR, April 2021.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2609–2634, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.147. URL https://aclanthology.org/2023.acl-long.147.

Xilu Wang, Yaochu Jin, Sebastian Schmitt, and Markus Olhofer. Recent advances in bayesian optimization. *ACM Computing Surveys*, 55(13s):1–36, 2023b. doi: 10.1145/3582078.

Xuezhi Wang and Denny Zhou. Chain-of-thought reasoning without prompting, 2024. URL https://arxiv.org/abs/2402.10200.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023c. URL https://openreview.net/forum?id=1PL1NIMMrw. Poster presentation.

Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando De Freitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55(1):361–387, 2016.

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.

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 *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9781713871088.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training, 2019. URL https://arxiv.org/abs/1908.04319.

Johnathan Xie, Annie S. Chen, Yoonho Lee, Eric Mitchell, and Chelsea Finn. Calibrating language models with adaptive temperature scaling. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 18128–18138. Association for Computational Linguistics, 2024.

Hanqi Yan, Qinglin Zhu, Xinyu Wang, Lin Gui, and Yulan He. Mirror: A multiple-perspective self-reflection method for knowledge-rich reasoning, 2024. URL https://arxiv.org/abs/2402.14963.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=5Xc1ecxO1h.

Jinghan Zhang, Fengran Mo, Xiting Wang, and Kunpeng Liu. Thought space explorer: Navigating and expanding thought space for large language model reasoning, 2024a. URL https://arxiv.org/abs/2410.24155.

Wenqi Zhang, Yongliang Shen, Linjuan Wu, Qiuying Peng, Jun Wang, Yueting Zhuang, and Weiming Lu. Self-contrast: Better reflection through inconsistent solving perspectives. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3602–3622, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.197. URL https://aclanthology.org/2024.acl-long.197/.

A Appendix

B Methodology Supplement

B.1 Technical Details on Optimisation Objective

Although we simply sum the two components, they naturally operate in a progressive, tie-breaking manner. If two candidate outputs differ in their verifier scores, the one with the higher r_{verifier} value immediately results in a higher $f(x)$. Only when the verifier scores are identical does $r_{\text{coherence}}$ serve to break the tie, making the process effectively hierarchical, even in this additive form.

B.2 Technical Details on Bayesian Optimisation

In constructing the prior distribution, for a finite collection of points $x_{1:n}$, the prior joint distribution on them is:

$$f(x_{1:n}) \sim \mathcal{N}(\mu_0(x_{1:n}), \Sigma_0(x_{1:n}, x_{1:n})),$$

where $f(x_{1:n}) = [f(x_1), \ldots, f(x_n)]^\top$, $\mu_0(x_{1:n}) = [\mu_0(x_1), \ldots, \mu_0(x_n)]^\top$, and the covariance matrix is:

$$\Sigma_0(x_{1:n}, x_{1:n}) = \begin{bmatrix} \Sigma_0(x_1, x_1) & \ldots & \Sigma_0(x_1, x_n) \\ \vdots & \ddots & \vdots \\ \Sigma_0(x_n, x_1) & \ldots & \Sigma_0(x_n, x_n) \end{bmatrix}.$$

We set $\mu_0(x) = 0$, indicating no additional preference for function values at the prior stage. For the covariance matrix $\Sigma_0(x_i, x_j)$, we use the Gaussian kernel with bandwidth ℓ:

$$\Sigma_0(x_i, x_j) = k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\ell^2}\right).$$

The kernel is chosen such that when x_i and x_j are close, the kernel value $k(x_i, x_j)$ approaches 1, indicating strong correlation, and when they are far apart, the kernel value approaches 0, reflecting weak correlation.

After observing $f(x_{1:k})$, we aim to infer the value of $f(x)$ at a new point x. Using Bayes' rule (Rasmussen & Williams, 2006), we update the posterior distribution of $f(x)$ conditioned on these observed values:

$$f(x) \mid f(x_{1:k}) \sim \mathcal{N}(\mu_k(x), \sigma_k^2(x)),$$

where the posterior mean and variance are given by

$$\begin{aligned}\mu_k(x) = {} & \Sigma_0(x, x_{1:k})\Sigma_0(x_{1:k}, x_{1:k})^{-1}(f(x_{1:k}) - \\ & \mu_0(x_{1:k})) + \mu_0(x), \\ \sigma_k^2(x) = {} & \Sigma_0(x, x) - \big[\Sigma_0(x, x_{1:k})\Sigma_0(x_{1:k}, x_{1:k})^{-1} \\ & \Sigma_0(x_{1:k}, x)\big].\end{aligned}$$

B.3 Maximising Expected Improvement

To select the point x that maximises the expected improvement $\text{EI}_k(x)$, we adopt a sampling-based approach. Specifically, we randomly sample a large number of candidate points from a standard normal distribution. In our experiments, we generate 5000 points $x_1, x_2, \ldots, x_{5000}$, where each point $x_i \in \mathbb{R}^D$ is sampled as:

$$x_i \sim \mathcal{N}(0, 1)^D, \quad i = 1, 2, \ldots, 5000.$$

For each sampled point, we compute the expected improvement $\text{EI}_k(x)$ using equation 3. After evaluating the expected improvement for all sampled points, we select the point with the maximum $\text{EI}_k(x)$ as the next candidate for evaluation. This method provides an efficient and practical way to approximate the global maximum of $\text{EI}_k(x)$ within the sampling budget.

B.4 Adaptive Expected Improvement

In defining $f(x)$, we assume an ideal verifier with perfect accuracy, meaning it provides an error-free assessment of correctness. However, in practice, the verifier's accuracy is less than 1, introducing uncertainty into its evaluations. This results in noisy observations, where the observed score o_k deviates from the true function value $f(x_k)$. We model this noise as:

$$o_k = f(x_k) + \eta_k, \quad \eta_k \sim \mathcal{N}(0, \lambda I),$$

where λ is an objective noise constant, determined by the inherent noise level. To address this noise in Bayesian optimisation, we use an adaptive version of the EI acquisition function that explicitly accounts for the uncertainty in observations:

$$\begin{aligned}\text{EI}_k(x) = \big[\mu_k(x) - f_k^*\big]^+ + \omega_k\sigma_k(x)\phi\left(\frac{\mu_k(x) - f_k^*}{\omega_k\sigma_k(x)}\right) \\ -\big|\mu_k(x) - f_k^*\big|\Phi\left(\frac{\mu_k(x) - f_k^*}{\omega_k\sigma_k(x)}\right),\end{aligned}$$

where $\omega_k = \sqrt{\gamma_k + 1 + \ln(1/\delta)}$ is the noise-adaptive scaling factor, and information gain term γ_k is defined as:

$$\begin{aligned}\gamma_k &= \max I(o(x_{1:k}); f(x_{i:k})) \\ &= \frac{1}{2}\log\det(I + \lambda^{-1}\Sigma_0(x_{1:k}, x_{1:k})).\end{aligned}$$

$I(o(x_{1:k}); f(x_{i:k}))$ represents the mutual information between the function values and the noisy observations, and δ is a hyperparameter in $(0, 1)$, controlling the balance between exploration and exploitation. This adaptation is inspired by the following observation (Vakili et al., 2021; Tran-The et al., 2022): For any choice of $\delta \in (0, 1)$, with probability at least $1 - \delta$, the cumulative regret R_T satisfies

$$R_T := \sum_{k=1}^{T} \big[f_k^* - f(x_k)\big] = O\big(\gamma_T \sqrt{T}\big).$$

This sublinear bound ensures that, with high probability, the regret grows at a slower rate than the number of iterations, thereby guaranteeing the convergence of the optimisation process.

We set $\delta = 0.1$, ensuring a 90% probability of convergence while balancing exploration and exploitation. A smaller δ (e.g., 0.01) strengthens theoretical guarantees but increases exploration, slowing convergence. A larger δ (e.g., 0.2) favours exploitation, accelerating convergence but weakening guarantees. Our choice provides stability and efficiency without excessive exploration.

B.5 Addressing the Curse of Dimensionality in Bayesian Optimisation

Traditional Bayesian optimisation struggles in high-dimensional spaces due to the curse of dimensionality, making it impractical for embedding vectors used in LLMs. To address this challenge, we leverage the following result from high-dimensional optimisation (Wang et al., 2016; Nayebi et al., 2019), which allows us to perform optimisation in a lower-dimensional space and to subsequently map points back to the original space.

Let D be the dimension of the embedding vectors. A function $f : \mathbb{R}^D \to \mathbb{R}$ is defined to have an *effective dimensionality* d_e, with $d_e \leq D$, if the following condition is satisfied: $\exists$ a subspace E of dimension d_e, such that $\forall x_E \in E \subset \mathbb{R}^D$ and $x_\perp \in E^\perp \subset \mathbb{R}^D$, where $E^\perp$ is the orthogonal complement of E, the function satisfies: $f(x_E + x_\perp) = f(x_E)$. In other words, d_e is the smallest dimension that retains all variability of f. Now, for $d \geq d_e$, consider a random matrix $A \in \mathbb{R}^{D\times d}$ with independent $\mathcal{N}(0, 1)$ entries. Then,

$$\forall x \in \mathbb{R}^D,\ \exists u \in \mathbb{R}^d \text{ such that } f(x) = f(Au).$$

This result implies that for any optimiser $x^* \in \mathbb{R}^D$, there exists a corresponding point $u^* \in \mathbb{R}^d$ such that $f(x^*) = f(Au^*)$. Therefore, instead of performing optimisation in the high-dimensional space, we can optimise the function $g(u) := f(Au)$ in the lower-dimensional space.

Suppose our initial sampled points and their evaluations are denoted by the set $\mathcal{U}_k = \{(u_1, g(u_1)), \ldots, (u_k, g(u_k))\}$, where each $u_i \in \mathbb{R}^d$ has independent standard normal entries. Similarly, we initialise $A \in \mathbb{R}^{D\times d}$ as a random matrix with independent $\mathcal{N}(0, 1)$ entries. We then find the next point to sample $u_{k+1} \in \mathbb{R}^d$ by optimising the acquisition function:

$$u_{k+1} = \underset{u\in\mathbb{R}^d}{\arg\max}\, EI_k(u \mid \mathcal{U}_k),$$

where $EI_k(\cdot \mid \mathcal{U}_k)$ represents the Expected Improvement conditioned on the current dataset $\mathcal{U}_k$ with the objective function being $g(\cdot)$ on $\mathbb{R}^d$. Then, we augment the dataset

$$\mathcal{U}_{k+1} := \mathcal{U}_k \cup \{(u_{k+1}, f(Au_{k+1}))\},$$

and iterate.

B.6 Setting the Convergence Threshold

The convergence threshold ϵ determines when the algorithm stops iterating. While smaller thresholds generally lead to more precise results, they can also increase computational costs due to additional iterations. We set $\epsilon = 0.01$ in our experiments, as this value strikes a balance between convergence quality and computational efficiency.

C More Experimental Details

C.1 Experiment Settings

For FIRE, CoT-Decoding, and RAP, we set the temperature to 0.8, following the settings of the respective baselines. We evaluate each approach on the four benchmark datasets (GSM8K, GSM-

Hard, SVAMP, and StrategyQA) by randomly sampling 200 test examples from each dataset. The generation process is cut off once the output reaches a maximum length of 300 tokens.

C.2 Additional Details on Neuron Activation Analysis

C.2.1 Overall Activation Rate

We analyse the neuron activations in the GLU-based MLP layers of the LLaMA model (Naik et al., 2024). Specifically, the hidden representation in the i-th layer is computed as:

$$h^i = \left(\text{act\_fn}(\tilde{h}^i W_1^i) \otimes \tilde{h}^i W_3^i\right) \cdot W_2^i, \tag{4}$$

where $\otimes$ denotes element-wise multiplication, and act_fn$(\cdot)$ is a non-linear activation function. We consider the j-th neuron inside the i-th FFN layer *activated* if its activation value $\left[\text{act\_fn}(\tilde{h}^i W_1^i)\right]_j$ exceeds zero.

We compare Self-Consistency (SC) (Wang et al., 2023c) with our Bayesian-optimisation-based perturbation method. For a single question (i.e., same prompt), we generate 200 samples using LLaMA and record neuron activations for each of the first 5 output tokens. We then visualise and compare the average activation rates between SC and our approach in Figure **??**.

C.2.2 Key Neuron Identification and Verification

We identify key neurons by analysing activation rates in SC-generated samples. We first separate these samples into correct and incorrect categories. For each neuron j in layer i, we calculate the activation difference:

$$\Delta_{i,j} = \text{avg}_{\text{correct}}(a_{i,j}) - \text{avg}_{\text{incorrect}}(a_{i,j}),$$

where $a_{i,j}$ denotes the activation value of the j-th neuron in the i-th layer. Neurons are then ranked by $\Delta_{i,j}$, and the top 5% with the largest positive values are selected as "key neurons."

To validate the identified key neurons under our method, we focus exclusively on those neurons marked as "key" by the SC analysis. Let $\text{avg}_{\text{SC, correct}}(\mathcal{N}_{\text{key}})$ denote the average activation of these key neurons in SC's correct samples. Likewise, $\text{avg}_{\text{Ours, correct}}(\mathcal{N}_{\text{key}})$ and $\text{avg}_{\text{Ours, wrong}}(\mathcal{N}_{\text{key}})$ denote the averages for our method's correct and incorrect samples, respectively. We find:

$$\frac{\text{avg}_{\text{Ours, correct}}(\mathcal{N}_{\text{key}})}{\text{avg}_{\text{SC, correct}}(\mathcal{N}_{\text{key}})} = 1.004,$$

indicating that in correct cases, our method activates these key neurons at a level comparable to SC. By contrast,

$$\frac{\text{avg}_{\text{Ours, wrong}}(\mathcal{N}_{\text{key}})}{\text{avg}_{\text{SC, correct}}(\mathcal{N}_{\text{key}})} = 0.794,$$

showing a markedly lower activation for incorrect samples under our method. These results confirm that the SC-derived key neurons remain crucial for correct reasoning, and our perturbation approach effectively utilises them while suppressing their activation in incorrect scenarios.

C.3 Prompts for Verifiers

We provide four prompt templates corresponding to the verifier strategies introduced in Section 5.2. These templates illustrate how each verifier approach is instantiated:

Single-Judge. This prompt asks the model to evaluate the correctness of a single final answer. The user provides a question, along with a final answer, and the verifier must decide whether that answer is correct.

Multi-Judge. This prompt provides multiple candidate answers and asks the verifier to assess their correctness collectively. The user includes each candidate's reasoning, and the verifier classifies which answers are correct.

Single-Generate. The prompt demonstrates a scenario where the verifier itself is prompted to *re-generate* the correct solution for one candidate. If the newly generated solution matches the candidate's answer, the candidate is deemed correct.

Multi-Generate (Ours). This prompt processes all candidate answers together and generates a new solution it believes to be correct. Any candidate matching this newly generated solution is labeled as correct. This leverages the model's generative capacity more thoroughly than pure classification, yielding better verification accuracy in practice.

Prompt for Single-Judge

Based on the given question and the previous answers, please provide your judgment on the correctness of the final answer.

Question:
Jessie currently weighs 9 kilograms. After she started to go jogging every day, she lost 62 kilograms in the first week and 140 kilograms in the second week. How much did she weigh before starting to jog?

Answer:
$9 + 62 + 140 = 211$. So, Jessie weighed 211 kilograms after 2 weeks of jogging. Since she weighed 9 kilograms initially, she weighed $211 - 9 = 202$ kilograms before starting to jog. Answer: 202

Correct:
0

Question:
{User Question}

Answer:
{Previous answers}

Correct:

C.4 Effect of the Number of Samples k.

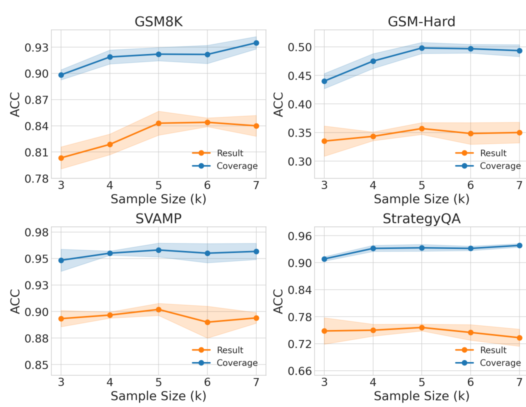


Figure 4: Accuracy (solid lines) and standard deviation (shaded areas) across different sample sizes k.

We investigate how the number of sampled embeddings k in each iteration influences final performance. Figure 4 shows that as k increases from 3 to 5, the final accuracy and coverage steadily rise for all evaluated tasks, although gains tend to plateau or fluctuate slightly after $k = 3$. This upward

Prompt for Multi-Judge

Based on the given question and the previous answers, please provide your judgment on the correctness of the final answer.

Question:
Jack is stranded on a desert island. He wants some salt to season his fish. He collects 2 liters of seawater in an old bucket. If the water is 20% salt, how many ml of salt will Jack get when all the water evaporates?

Your previous answers:
0. Thought: 1250 ml of water evaporates, leaving 1000 ml of salt. Answer: 1000
1. Thought: The total amount of water is 2 liters = 2000 ml. The amount of salt is 20% of 2000 ml = 0.20×2000 ml = $\langle\langle 0.20 \times 2000 = 400\rangle\rangle$400 ml. Answer: 400
2. Thought: 20% of 2 liters is $2 \times \frac{20}{100} = \langle\langle 2 \times 20/100 = 0.4\rangle\rangle 0.4$ liters. Since there are 1000 ml in 1 liter, 0.4 liters is $0.4 \times 1000 = \langle\langle 0.4 \times 1000 = 400\rangle\rangle 400$ ml. Answer: 400
3. Thought: 1 liter of seawater is 20% salt. So, 1 liter of seawater has $20\% \times 1$ liter = $\langle\langle 20 \times 0.1 = 0.2\rangle\rangle 0.2$ liters of salt. Since Jack has 2 liters of seawater, he will get $0.2 \times 2 = \langle\langle 0.2 \times 2 = 0.4\rangle\rangle 0.4$ liters of salt. Since there are 1000 ml in 1 liter, Jack will get $0.4 \times 1000 = \langle\langle 0.4 \times 1000 = 400\rangle\rangle 400$ ml of salt. Answer: 400
4. Thought: 20% of 2 liters is $2 \times \frac{20}{100} = \langle\langle 2 \times 20/100 = 0.4\rangle\rangle 0.4$ liters. There are 1000 ml in 1 liter, so 0.4 liters is $0.4 \times 1000 = \langle\langle 0.4 \times 1000 = 400\rangle\rangle 400$ ml. Answer: 400

Correct:
1, 2, 3, 4

Question:
{User Question}

Your previous answers:
{Previous answers}

Correct:

Prompt for Single-Generate

Based on the given question and the previous answers, please provide your analysis and final answer, starting the final answer with "Answer:"

Question:
Jack is stranded on a desert island. He wants some salt to season his fish. He collects 2 liters of seawater in an old bucket. If the water is 20% salt, how many ml of salt will Jack get when all the water evaporates?

Your previous answers:
1250 ml of water evaporates, leaving 1000 ml of salt. Answer: 1000

Analysis:
Let's think step by step. Jack has 2 liters of seawater, and 20% of it is salt. 2 liters = 2000 ml, so the amount of salt is 20% of 2000 ml = $0.20 \times 2000 = 400$ ml of salt.

Answer:
400

Question:
{User Question}

Your previous answers:
{Previous answers}

Analysis:

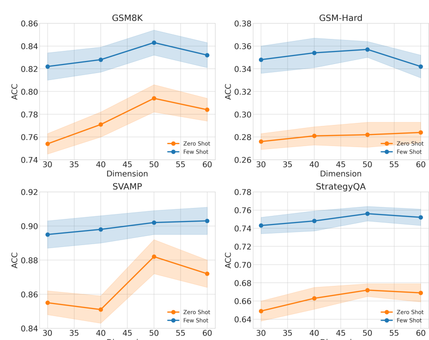


Figure 5: Accuracy (solid lines) and standard deviation (shaded areas) across reduced dimensions.

trend suggests that a moderate increase in k promotes better exploration of potentially correct solutions in the embedding space. Based on these observations, we adopt $k = 5$ as our default setting, balancing solution diversity with computational cost.

C.5 Additional Results

Tables 6 and 7 present the full experimental results for all methods and configurations across the four benchmarks and three LLMs (LLaMA3-8B-Instruct, Qwen2-7B-Instruct, and Mistral-7B-Instruct). These tables extend the summary reported in the main text, showing detailed accuracy, coverage, and standard deviations for zero-shot to 8-shot setups. They provide a comprehensive view of how each baseline and our approach perform under various hyperparameter and prompt configurations.

Prompt for Multi-Generate (ours)

Based on the given question and the previous answers, please provide your analysis and final answer, starting the final answer with "Answer:"

Question:
Artemis is making tea for a party. She knows her mom drinks an 8-ounce cup of tea and uses one ounce of tea. She will use this same ratio for the party. The party has 12 people there and each of them wants a 6-ounce cup of tea. How many ounces of tea does she need?

Your previous answers:
0. Thought: 8 ounces of tea for 1 cup, so 1 ounce of tea for $\frac{1}{8}$ of a cup. For 12 people, she needs $12 \times \frac{6}{8} = 9$ ounces of tea. Answer: 9
1. Thought: 6 ounces of tea is needed for each person. Since there are 12 people, $12 \times 6 = 72$ ounces of tea are needed. Since each ounce of tea is used for 1 cup, 72 ounces of tea will make 72 cups of tea. Answer: 72
2. Thought: 6 ounces of tea is $\frac{6}{8} = \frac{3}{4}$ of an 8-ounce cup. For 12 people, she needs $12 \times \frac{3}{4} = 9$ ounces of tea. Answer: 9
3. Thought: $12 \times 6 = 72$ ounces of tea needed. Since each ounce of tea is used for 1 cup, Artemis needs 72 ounces of tea. Answer: 72
4. Thought: 8 ounces of tea is used for 1 cup. So for 6 ounces of tea, she will use $\frac{6}{8} = \frac{3}{4}$ of the amount of tea. For 12 people, she will need $12 \times \frac{3}{4} = 9$ ounces of tea. Answer: 9

Analysis:
Let's think step by step. Artemis uses 1 ounce of tea for an 8-ounce cup, so for a 6-ounce cup, she will use $\frac{6}{8} = \frac{3}{4}$ of an ounce of tea. For 12 people, she needs $12 \times \frac{3}{4} = 9$ ounces of tea.

Answer:
9

Question:
{User Question}

Your previous answers:
{Previous answers}

Analysis:

Table 6: Performances of different reasoning methods on Accuracy (%) across all benchmarks.

Method	LLaMA3-8B-Instruct					Qwen2-7B-Instruct					Mistral-7B-Instruct				
	Shot 0	Shot 1	Shot 2	Shot 4	Shot 8	Shot 0	Shot 1	Shot 2	Shot 4	Shot 8	Shot 0	Shot 1	Shot 2	Shot 4	Shot 8
GSM8K															
COT	$53.0_{\pm 0.0}$	$73.0_{\pm 0.0}$	$73.5_{\pm 0.0}$	$79.0_{\pm 0.0}$	$77.4_{\pm 0.0}$	$64.5_{\pm 0.0}$	$70.0_{\pm 0.0}$	$66.5_{\pm 0.0}$	$81.5_{\pm 0.0}$	$82.5_{\pm 0.0}$	$42.0_{\pm 0.0}$	$43.5_{\pm 0.0}$	$53.5_{\pm 0.0}$	$54.5_{\pm 0.0}$	$54.0_{\pm 0.0}$
SC($\tau = 0.4$)	$73.0_{\pm 1.6}$	$75.4_{\pm 2.8}$	$75.7_{\pm 2.4}$	$80.7_{\pm 1.1}$	$80.4_{\pm 1.4}$	$81.2_{\pm 0.6}$	$82.4_{\pm 1.8}$	$83.7_{\pm 0.4}$	$86.9_{\pm 0.6}$	$85.7_{\pm 1.5}$	$52.9_{\pm 0.5}$	$53.5_{\pm 1.5}$	$59.9_{\pm 2.3}$	$60.4_{\pm 1.7}$	$58.3_{\pm 1.5}$
SC($\tau = 0.6$)	$73.6_{\pm 2.5}$	$73.4_{\pm 3.6}$	$72.6_{\pm 1.9}$	$80.3_{\pm 0.9}$	$80.6_{\pm 1.5}$	$80.2_{\pm 1.9}$	$84.5_{\pm 2.1}$	$84.2_{\pm 0.4}$	$86.5_{\pm 0.9}$	$85.4_{\pm 0.9}$	$55.1_{\pm 3.6}$	$56.5_{\pm 2.0}$	$59.6_{\pm 0.7}$	$60.5_{\pm 1.2}$	$57.4_{\pm 1.0}$
SC($\tau = 0.8$)	$65.0_{\pm 2.0}$	$74.5_{\pm 0.7}$	$66.8_{\pm 1.7}$	$80.9_{\pm 1.5}$	$81.1_{\pm 1.1}$	$80.0_{\pm 0.9}$	$82.7_{\pm 2.9}$	$82.6_{\pm 0.7}$	$85.4_{\pm 0.8}$	$85.1_{\pm 1.6}$	$50.2_{\pm 2.6}$	$51.4_{\pm 1.4}$	$56.4_{\pm 2.7}$	$59.8_{\pm 2.6}$	$57.7_{\pm 2.6}$
FIRE	$73.8_{\pm 2.3}$	$75.4_{\pm 1.8}$	$76.4_{\pm 2.0}$	$78.4_{\pm 3.2}$	$79.6_{\pm 2.9}$	$81.0_{\pm 1.8}$	$81.0_{\pm 2.9}$	$73.6_{\pm 2.2}$	$82.5_{\pm 2.1}$	$83.0_{\pm 1.3}$	$47.2_{\pm 2.9}$	$52.4_{\pm 2.3}$	$53.8_{\pm 1.7}$	$60.6_{\pm 0.9}$	$56.1_{\pm 3.2}$
CoT-Decoding	$73.9_{\pm 1.9}$	$76.7_{\pm 1.4}$	$78.6_{\pm 1.7}$	$81.4_{\pm 1.8}$	$80.3_{\pm 1.7}$	$82.0_{\pm 2.8}$	$81.6_{\pm 2.5}$	$76.2_{\pm 2.5}$	$85.7_{\pm 0.8}$	$84.5_{\pm 2.1}$	$47.3_{\pm 3.0}$	$56.4_{\pm 3.3}$	$57.3_{\pm 2.5}$	$57.6_{\pm 3.4}$	$58.2_{\pm 2.3}$
RAP	-	$77.4_{\pm 1.7}$	$79.4_{\pm 1.6}$	$80.4_{\pm 1.4}$	$80.7_{\pm 1.4}$	-	$84.7_{\pm 2.0}$	$85.7_{\pm 1.0}$	$87.4_{\pm 0.9}$	$86.2_{\pm 1.2}$	-	$57.6_{\pm 1.8}$	$57.1_{\pm 1.7}$	$59.8_{\pm 1.6}$	$58.6_{\pm 1.8}$
Ours	$\mathbf{79.4}_{\pm 1.2}$	$\mathbf{79.4}_{\pm 2.8}$	$\mathbf{83.0}_{\pm 0.8}$	$\mathbf{83.5}_{\pm 0.9}$	$\mathbf{84.3}_{\pm 1.4}$	$\mathbf{88.6}_{\pm 1.2}$	$\mathbf{88.8}_{\pm 1.4}$	$\mathbf{88.4}_{\pm 1.0}$	$\mathbf{89.8}_{\pm 2.5}$	$\mathbf{90.0}_{\pm 1.4}$	$\mathbf{61.4}_{\pm 2.5}$	$\mathbf{61.2}_{\pm 2.5}$	$\mathbf{61.2}_{\pm 0.9}$	$\mathbf{61.4}_{\pm 1.9}$	$\mathbf{62.7}_{\pm 1.0}$
GSM-Hard															
COT	$14.0_{\pm 0.0}$	$22.5_{\pm 0.0}$	$24.5_{\pm 0.0}$	$26.5_{\pm 0.0}$	$28.0_{\pm 0.0}$	$40.0_{\pm 0.0}$	$39.0_{\pm 0.0}$	$48.0_{\pm 0.0}$	$53.0_{\pm 0.0}$	$55.5_{\pm 0.0}$	$14.5_{\pm 0.0}$	$22.0_{\pm 0.0}$	$21.5_{\pm 0.0}$	$23.5_{\pm 0.0}$	$24.0_{\pm 0.0}$
SC($\tau = 0.4$)	$25.7_{\pm 0.4}$	$25.3_{\pm 0.4}$	$28.6_{\pm 1.2}$	$32.2_{\pm 0.4}$	$31.8_{\pm 1.8}$	$47.5_{\pm 1.4}$	$48.6_{\pm 1.4}$	$53.7_{\pm 1.3}$	$55.2_{\pm 1.5}$	$55.4_{\pm 0.7}$	$19.5_{\pm 1.0}$	$26.5_{\pm 1.6}$	$26.9_{\pm 1.4}$	$27.3_{\pm 1.0}$	$26.1_{\pm 1.5}$
SC($\tau = 0.6$)	$24.5_{\pm 1.1}$	$25.7_{\pm 0.5}$	$28.4_{\pm 1.4}$	$32.0_{\pm 1.1}$	$31.2_{\pm 1.3}$	$46.2_{\pm 1.9}$	$50.1_{\pm 2.9}$	$54.2_{\pm 0.7}$	$56.0_{\pm 1.1}$	$53.4_{\pm 0.6}$	$20.7_{\pm 1.5}$	$25.7_{\pm 0.9}$	$27.8_{\pm 1.2}$	$31.0_{\pm 2.5}$	$25.3_{\pm 1.6}$
SC($\tau = 0.8$)	$21.8_{\pm 1.3}$	$24.8_{\pm 1.9}$	$28.2_{\pm 2.7}$	$30.6_{\pm 1.2}$	$30.8_{\pm 0.9}$	$47.3_{\pm 1.3}$	$47.8_{\pm 2.7}$	$54.8_{\pm 1.5}$	$55.7_{\pm 1.3}$	$55.4_{\pm 0.9}$	$19.1_{\pm 2.0}$	$26.8_{\pm 2.5}$	$26.6_{\pm 1.2}$	$29.8_{\pm 2.1}$	$26.6_{\pm 1.1}$
FIRE	$25.2_{\pm 3.0}$	$24.1_{\pm 1.5}$	$27.0_{\pm 1.5}$	$27.9_{\pm 1.8}$	$25.7_{\pm 2.1}$	$45.1_{\pm 2.0}$	$43.4_{\pm 2.5}$	$52.9_{\pm 1.9}$	$49.0_{\pm 2.5}$	$51.0_{\pm 1.8}$	$18.1_{\pm 1.9}$	$25.8_{\pm 1.3}$	$27.0_{\pm 1.7}$	$25.5_{\pm 2.8}$	$26.3_{\pm 1.4}$
CoT-Decoding	$24.8_{\pm 1.3}$	$27.3_{\pm 1.6}$	$28.0_{\pm 0.4}$	$31.0_{\pm 1.8}$	$30.3_{\pm 1.3}$	$46.7_{\pm 2.3}$	$44.1_{\pm 1.8}$	$53.9_{\pm 1.5}$	$50.1_{\pm 1.2}$	$52.1_{\pm 1.0}$	$16.6_{\pm 0.7}$	$26.6_{\pm 2.6}$	$28.0_{\pm 0.7}$	$26.0_{\pm 2.0}$	$27.4_{\pm 1.6}$
RAP	-	$26.7_{\pm 1.0}$	$31.4_{\pm 1.2}$	$32.4_{\pm 1.1}$	$32.7_{\pm 1.2}$	-	$53.2_{\pm 1.9}$	$55.1_{\pm 1.2}$	$55.9_{\pm 1.3}$	$56.2_{\pm 0.8}$	-	$27.4_{\pm 1.5}$	$28.0_{\pm 1.0}$	$28.4_{\pm 1.7}$	$27.6_{\pm 1.2}$
Ours	$\mathbf{28.2}_{\pm 1.8}$	$\mathbf{30.2}_{\pm 1.2}$	$\mathbf{35.3}_{\pm 1.4}$	$\mathbf{33.2}_{\pm 0.6}$	$\mathbf{35.7}_{\pm 1.0}$	$\mathbf{53.7}_{\pm 1.6}$	$\mathbf{57.4}_{\pm 0.7}$	$\mathbf{57.5}_{\pm 0.8}$	$\mathbf{57.4}_{\pm 1.2}$	$\mathbf{58.7}_{\pm 0.5}$	$\mathbf{25.8}_{\pm 1.8}$	$\mathbf{29.1}_{\pm 1.2}$	$\mathbf{32.3}_{\pm 0.4}$	$\mathbf{30.0}_{\pm 1.2}$	$\mathbf{32.5}_{\pm 1.5}$
SVAMP															
COT	$61.0_{\pm 0.0}$	$81.0_{\pm 0.0}$	$83.5_{\pm 0.0}$	$84.0_{\pm 0.0}$	$83.0_{\pm 0.0}$	$43.5_{\pm 0.0}$	$83.0_{\pm 0.0}$	$84.0_{\pm 0.0}$	$85.5_{\pm 0.0}$	$86.0_{\pm 0.0}$	$52.0_{\pm 0.0}$	$65.0_{\pm 0.0}$	$66.0_{\pm 0.0}$	$69.5_{\pm 0.0}$	$72.0_{\pm 0.0}$
SC($\tau = 0.4$)	$79.1_{\pm 1.2}$	$85.8_{\pm 1.5}$	$86.5_{\pm 0.7}$	$87.3_{\pm 0.8}$	$87.1_{\pm 1.0}$	$72.3_{\pm 2.0}$	$90.2_{\pm 1.0}$	$89.4_{\pm 0.5}$	$90.3_{\pm 0.9}$	$90.3_{\pm 1.2}$	$67.4_{\pm 2.5}$	$74.5_{\pm 1.7}$	$73.7_{\pm 1.0}$	$76.5_{\pm 1.5}$	$77.8_{\pm 1.0}$
SC($\tau = 0.6$)	$76.1_{\pm 3.9}$	$86.2_{\pm 0.5}$	$86.8_{\pm 1.7}$	$86.9_{\pm 1.2}$	$87.7_{\pm 1.2}$	$77.3_{\pm 1.2}$	$90.6_{\pm 0.9}$	$90.2_{\pm 0.8}$	$\mathbf{91.4}_{\pm 0.9}$	$90.4_{\pm 0.6}$	$69.7_{\pm 1.6}$	$75.8_{\pm 1.5}$	$75.6_{\pm 0.8}$	$75.8_{\pm 1.4}$	$78.4_{\pm 2.0}$
SC($\tau = 0.8$)	$69.6_{\pm 2.0}$	$86.3_{\pm 2.2}$	$86.9_{\pm 1.0}$	$87.4_{\pm 1.5}$	$87.4_{\pm 1.2}$	$78.6_{\pm 2.1}$	$90.3_{\pm 0.7}$	$90.1_{\pm 1.0}$	$90.8_{\pm 0.7}$	$90.6_{\pm 1.2}$	$68.3_{\pm 0.9}$	$75.1_{\pm 0.7}$	$76.6_{\pm 1.5}$	$76.9_{\pm 1.6}$	$77.6_{\pm 1.1}$
FIRE	$81.5_{\pm 0.8}$	$86.6_{\pm 1.8}$	$86.1_{\pm 1.3}$	$86.1_{\pm 1.4}$	$87.6_{\pm 2.0}$	$76.3_{\pm 2.2}$	$89.9_{\pm 1.4}$	$89.7_{\pm 0.8}$	$89.3_{\pm 0.9}$	$90.6_{\pm 0.2}$	$67.1_{\pm 1.9}$	$77.7_{\pm 1.1}$	$76.9_{\pm 2.7}$	$77.8_{\pm 1.2}$	$78.4_{\pm 1.2}$
CoT-Decoding	$83.2_{\pm 1.2}$	$87.8_{\pm 1.0}$	$87.5_{\pm 1.0}$	$87.5_{\pm 1.3}$	$88.2_{\pm 1.0}$	$78.6_{\pm 1.6}$	$90.3_{\pm 0.4}$	$90.0_{\pm 1.0}$	$90.3_{\pm 1.0}$	$89.7_{\pm 0.5}$	$69.4_{\pm 2.5}$	$77.8_{\pm 2.0}$	$77.7_{\pm 1.5}$	$76.9_{\pm 2.5}$	$78.6_{\pm 1.4}$
RAP	-	$78.4_{\pm 1.2}$	$87.4_{\pm 1.0}$	$86.8_{\pm 1.0}$	$87.9_{\pm 1.1}$	-	$90.8_{\pm 0.7}$	$91.2_{\pm 0.7}$	$90.1_{\pm 0.7}$	$90.8_{\pm 0.6}$	-	$0.0_{\pm 1.2}$	$0.0_{\pm 1.3}$	$78.4_{\pm 1.4}$	$79.4_{\pm 1.1}$
Ours	$\mathbf{88.2}_{\pm 1.3}$	$\mathbf{89.2}_{\pm 1.5}$	$\mathbf{88.8}_{\pm 1.0}$	$\mathbf{89.1}_{\pm 0.8}$	$\mathbf{90.2}_{\pm 0.6}$	$\mathbf{83.4}_{\pm 2.4}$	$\mathbf{92.3}_{\pm 0.8}$	$\mathbf{92.4}_{\pm 0.8}$	$90.8_{\pm 0.8}$	$\mathbf{92.2}_{\pm 0.8}$	$\mathbf{72.2}_{\pm 2.2}$	$\mathbf{79.0}_{\pm 0.9}$	$\mathbf{78.4}_{\pm 1.3}$	$\mathbf{80.0}_{\pm 1.4}$	$\mathbf{82.1}_{\pm 1.2}$
StrategyQA															
COT	$58.5_{\pm 0.0}$	$63.0_{\pm 0.0}$	$68.0_{\pm 0.0}$	$67.5_{\pm 0.0}$	$68.5_{\pm 0.0}$	$63.0_{\pm 0.0}$	$54.5_{\pm 0.0}$	$63.5_{\pm 0.0}$	$66.0_{\pm 0.0}$	$70.0_{\pm 0.0}$	$62.0_{\pm 0.0}$	$62.5_{\pm 0.0}$	$63.5_{\pm 0.0}$	$68.5_{\pm 0.0}$	$69.0_{\pm 0.0}$
SC($\tau = 0.4$)	$64.7_{\pm 0.7}$	$68.4_{\pm 2.4}$	$68.6_{\pm 1.7}$	$69.2_{\pm 1.2}$	$71.6_{\pm 0.8}$	$67.1_{\pm 1.5}$	$67.4_{\pm 2.0}$	$66.5_{\pm 1.2}$	$69.1_{\pm 1.2}$	$71.1_{\pm 1.6}$	$63.9_{\pm 1.5}$	$57.6_{\pm 2.0}$	$65.9_{\pm 0.7}$	$70.2_{\pm 1.9}$	$72.6_{\pm 1.2}$
SC($\tau = 0.6$)	$59.9_{\pm 2.0}$	$69.9_{\pm 1.2}$	$68.2_{\pm 0.8}$	$70.7_{\pm 2.2}$	$71.3_{\pm 1.5}$	$67.5_{\pm 0.7}$	$66.5_{\pm 1.7}$	$67.4_{\pm 2.6}$	$67.7_{\pm 1.4}$	$69.1_{\pm 1.2}$	$64.2_{\pm 1.0}$	$58.7_{\pm 1.3}$	$64.8_{\pm 0.7}$	$69.6_{\pm 1.2}$	$71.7_{\pm 0.8}$
SC($\tau = 0.8$)	$54.4_{\pm 2.6}$	$68.0_{\pm 2.0}$	$67.9_{\pm 0.7}$	$70.4_{\pm 0.9}$	$72.7_{\pm 1.2}$	$67.0_{\pm 1.0}$	$68.0_{\pm 1.9}$	$68.3_{\pm 1.2}$	$67.7_{\pm 2.5}$	$70.1_{\pm 0.8}$	$64.9_{\pm 1.0}$	$59.9_{\pm 1.2}$	$66.5_{\pm 0.4}$	$69.9_{\pm 1.4}$	$72.1_{\pm 1.5}$
FIRE	$63.0_{\pm 3.7}$	$70.8_{\pm 1.6}$	$71.8_{\pm 1.3}$	$70.2_{\pm 2.0}$	$72.8_{\pm 1.5}$	$67.6_{\pm 0.8}$	$68.4_{\pm 0.8}$	$68.4_{\pm 1.8}$	$67.3_{\pm 1.2}$	$68.1_{\pm 0.8}$	$64.2_{\pm 1.0}$	$66.5_{\pm 1.5}$	$68.2_{\pm 1.4}$	$72.6_{\pm 2.1}$	$71.0_{\pm 2.2}$
CoT-Decoding	$64.6_{\pm 1.6}$	$71.1_{\pm 3.1}$	$71.4_{\pm 2.0}$	$70.5_{\pm 2.0}$	$73.3_{\pm 1.8}$	$65.9_{\pm 1.5}$	$68.3_{\pm 2.5}$	$68.7_{\pm 0.7}$	$67.5_{\pm 1.5}$	$69.5_{\pm 2.1}$	$63.5_{\pm 1.5}$	$68.2_{\pm 0.4}$	$68.6_{\pm 0.7}$	$70.6_{\pm 1.6}$	$72.7_{\pm 2.1}$
RAP	-	$\mathbf{71.6}_{\pm 1.7}$	$70.6_{\pm 1.1}$	$71.5_{\pm 1.4}$	$73.4_{\pm 1.1}$	-	$67.5_{\pm 1.5}$	$68.2_{\pm 1.3}$	$68.5_{\pm 1.3}$	$\mathbf{71.3}_{\pm 1.1}$	-	$68.5_{\pm 1.1}$	$70.6_{\pm 0.7}$	$72.1_{\pm 1.4}$	$72.4_{\pm 1.3}$
Ours	$\mathbf{67.2}_{\pm 0.7}$	$71.0_{\pm 0.9}$	$\mathbf{72.3}_{\pm 1.2}$	$\mathbf{73.8}_{\pm 1.2}$	$\mathbf{75.6}_{\pm 0.8}$	$\mathbf{68.1}_{\pm 1.5}$	$\mathbf{69.6}_{\pm 1.6}$	$\mathbf{69.7}_{\pm 1.2}$	$\mathbf{68.9}_{\pm 0.5}$	$70.3_{\pm 1.3}$	$\mathbf{66.1}_{\pm 1.9}$	$\mathbf{70.1}_{\pm 0.7}$	$\mathbf{71.2}_{\pm 1.4}$	$\mathbf{72.7}_{\pm 1.0}$	$\mathbf{72.8}_{\pm 1.5}$

Table 7: Coverage rates of correct answers across different models (%) on all benchmarks.

Method	LLaMA3-8B-Instruct					Qwen2-7B-Instruct					Mistral-7B-Instruct				
	Shot 0	Shot 1	Shot 2	Shot 4	Shot 8	Shot 0	Shot 1	Shot 2	Shot 4	Shot 8	Shot 0	Shot 1	Shot 2	Shot 4	Shot 8
							GSM8K								
SC(τ=0.4)	$79.8_{\pm 0.8}$	$69.2_{\pm 2.8}$	$75.7_{\pm 2.4}$	$89.5_{\pm 1.4}$	$91.0_{\pm 1.5}$	$93.1_{\pm 0.4}$	$88.9_{\pm 1.7}$	$91.4_{\pm 1.0}$	$93.5_{\pm 0.7}$	$94.5_{\pm 0.6}$	$73.2_{\pm 3.0}$	$69.3_{\pm 1.6}$	$75.5_{\pm 2.2}$	$74.9_{\pm 1.9}$	$73.4_{\pm 1.0}$
SC(τ=0.6)	$78.4_{\pm 2.2}$	$66.6_{\pm 3.6}$	$72.6_{\pm 1.9}$	$90.6_{\pm 1.3}$	$91.1_{\pm 0.8}$	$94.2_{\pm 0.7}$	$88.5_{\pm 1.5}$	$92.4_{\pm 0.7}$	$93.8_{\pm 0.7}$	$94.2_{\pm 1.3}$	$73.0_{\pm 1.3}$	$72.7_{\pm 2.7}$	$76.2_{\pm 1.6}$	$76.5_{\pm 2.5}$	$73.5_{\pm 2.0}$
SC(τ=0.8)	$71.0_{\pm 1.2}$	$62.1_{\pm 0.7}$	$66.8_{\pm 1.7}$	$90.4_{\pm 1.6}$	$90.2_{\pm 1.6}$	$92.9_{\pm 1.2}$	$88.6_{\pm 1.5}$	$91.6_{\pm 1.1}$	$93.1_{\pm 0.9}$	$93.8_{\pm 0.7}$	$71.2_{\pm 1.8}$	$70.1_{\pm 2.0}$	$74.3_{\pm 2.8}$	$74.5_{\pm 2.8}$	$73.2_{\pm 1.4}$
FIRE	$84.5_{\pm 1.3}$	$83.9_{\pm 2.0}$	$88.8_{\pm 2.0}$	$89.4_{\pm 1.1}$	$88.2_{\pm 0.6}$	$86.8_{\pm 1.8}$	$78.1_{\pm 2.6}$	$84.2_{\pm 1.3}$	$90.6_{\pm 1.9}$	$90.8_{\pm 0.9}$	$63.1_{\pm 2.5}$	$70.4_{\pm 2.5}$	$70.9_{\pm 1.6}$	$76.7_{\pm 2.2}$	$73.9_{\pm 2.8}$
CoT-Decoding	$85.3_{\pm 1.7}$	$88.1_{\pm 1.2}$	$90.6_{\pm 1.1}$	$90.2_{\pm 0.8}$	$90.5_{\pm 0.9}$	$88.4_{\pm 2.1}$	$81.2_{\pm 2.0}$	$85.5_{\pm 2.4}$	$92.0_{\pm 0.9}$	$91.7_{\pm 0.8}$	$63.2_{\pm 2.4}$	$72.6_{\pm 2.7}$	$75.5_{\pm 2.5}$	$74.3_{\pm 2.3}$	$76.6_{\pm 1.2}$
RAP	-	$87.6_{\pm 0.9}$	$89.1_{\pm 1.3}$	$88.9_{\pm 1.2}$	$89.5_{\pm 0.8}$	-	$88.3_{\pm 1.2}$	$92.4_{\pm 1.2}$	$93.4_{\pm 1.1}$	$92.3_{\pm 0.7}$	-	$72.4_{\pm 2.1}$	$74.5_{\pm 1.3}$	$75.8_{\pm 1.5}$	$75.6_{\pm 1.3}$
Ours	$\mathbf{91.8}_{\pm 1.4}$	$\mathbf{91.1}_{\pm 0.8}$	$\mathbf{92.4}_{\pm 1.1}$	$\mathbf{92.6}_{\pm 1.4}$	$\mathbf{92.2}_{\pm 0.8}$	$\mathbf{95.9}_{\pm 0.1}$	$\mathbf{96.8}_{\pm 0.6}$	$\mathbf{96.4}_{\pm 1.0}$	$\mathbf{96.8}_{\pm 1.4}$	$\mathbf{96.6}_{\pm 0.7}$	$\mathbf{85.4}_{\pm 1.4}$	$\mathbf{79.0}_{\pm 2.4}$	$\mathbf{82.3}_{\pm 1.0}$	$\mathbf{81.9}_{\pm 1.4}$	$\mathbf{82.5}_{\pm 0.9}$
							GSM-Hard								
SC(τ=0.4)	$27.3_{\pm 0.4}$	$31.7_{\pm 0.5}$	$38.6_{\pm 1.9}$	$41.6_{\pm 1.8}$	$43.4_{\pm 0.4}$	$62.6_{\pm 1.6}$	$53.4_{\pm 1.7}$	$62.4_{\pm 1.2}$	$62.6_{\pm 1.7}$	$64.3_{\pm 0.7}$	$31.1_{\pm 1.1}$	$37.5_{\pm 1.6}$	$40.9_{\pm 1.2}$	$39.9_{\pm 1.6}$	$38.2_{\pm 1.5}$
SC(τ=0.6)	$28.2_{\pm 1.6}$	$33.0_{\pm 0.8}$	$38.0_{\pm 1.4}$	$39.9_{\pm 2.2}$	$43.0_{\pm 1.1}$	$63.3_{\pm 2.0}$	$56.2_{\pm 2.2}$	$62.4_{\pm 1.1}$	$65.3_{\pm 0.7}$	$65.8_{\pm 1.4}$	$30.2_{\pm 1.7}$	$37.3_{\pm 0.9}$	$38.8_{\pm 1.3}$	$41.3_{\pm 1.4}$	$39.1_{\pm 2.5}$
SC(τ=0.8)	$25.1_{\pm 0.9}$	$32.3_{\pm 2.4}$	$37.8_{\pm 1.2}$	$39.3_{\pm 1.2}$	$42.2_{\pm 1.1}$	$61.8_{\pm 0.5}$	$54.9_{\pm 1.9}$	$64.1_{\pm 1.0}$	$63.8_{\pm 1.8}$	$65.1_{\pm 1.0}$	$31.7_{\pm 1.4}$	$38.2_{\pm 1.6}$	$39.2_{\pm 1.0}$	$40.9_{\pm 1.8}$	$37.0_{\pm 1.1}$
FIRE	$32.0_{\pm 2.3}$	$33.1_{\pm 2.5}$	$38.3_{\pm 1.4}$	$39.9_{\pm 1.1}$	$40.6_{\pm 2.0}$	$54.4_{\pm 2.1}$	$49.3_{\pm 2.3}$	$56.7_{\pm 2.1}$	$56.8_{\pm 2.1}$	$60.6_{\pm 1.2}$	$26.2_{\pm 2.5}$	$37.4_{\pm 2.1}$	$39.8_{\pm 1.8}$	$39.1_{\pm 2.7}$	$35.8_{\pm 1.6}$
CoT-Decoding	$33.1_{\pm 0.9}$	$33.6_{\pm 1.9}$	$39.8_{\pm 1.4}$	$42.3_{\pm 1.2}$	$42.2_{\pm 1.5}$	$55.1_{\pm 1.0}$	$49.3_{\pm 2.2}$	$56.6_{\pm 1.3}$	$57.4_{\pm 0.6}$	$61.5_{\pm 1.4}$	$27.7_{\pm 1.2}$	$36.9_{\pm 1.0}$	$40.3_{\pm 1.6}$	$38.5_{\pm 1.4}$	$38.4_{\pm 0.7}$
RAP	-	$33.4_{\pm 1.6}$	$40.2_{\pm 1.5}$	$41.9_{\pm 1.0}$	$43.9_{\pm 1.3}$	-	$53.7_{\pm 1.8}$	$60.2_{\pm 1.2}$	$62.1_{\pm 1.1}$	$64.2_{\pm 1.2}$	-	$37.6_{\pm 1.5}$	$40.6_{\pm 1.2}$	$40.3_{\pm 1.7}$	$38.4_{\pm 1.4}$
Ours	$\mathbf{37.0}_{\pm 1.5}$	$\mathbf{37.0}_{\pm 0.8}$	$\mathbf{46.4}_{\pm 2.5}$	$\mathbf{47.3}_{\pm 2.2}$	$\mathbf{49.8}_{\pm 1.0}$	$\mathbf{69.4}_{\pm 1.7}$	$\mathbf{67.8}_{\pm 0.9}$	$\mathbf{70.7}_{\pm 1.0}$	$\mathbf{71.1}_{\pm 1.1}$	$\mathbf{71.1}_{\pm 1.1}$	$\mathbf{40.7}_{\pm 1.2}$	$\mathbf{43.3}_{\pm 1.2}$	$\mathbf{44.4}_{\pm 1.5}$	$\mathbf{45.9}_{\pm 1.1}$	$\mathbf{45.2}_{\pm 1.9}$
							SVAMP								
SC(τ=0.4)	$84.6_{\pm 1.1}$	$91.8_{\pm 0.5}$	$91.6_{\pm 1.6}$	$92.8_{\pm 0.4}$	$92.9_{\pm 0.6}$	$91.7_{\pm 1.8}$	$94.3_{\pm 0.6}$	$93.9_{\pm 0.7}$	$93.7_{\pm 0.4}$	$94.0_{\pm 0.9}$	$61.9_{\pm 1.7}$	$85.2_{\pm 1.3}$	$84.5_{\pm 1.1}$	$86.3_{\pm 1.3}$	$87.7_{\pm 0.4}$
SC(τ=0.6)	$82.1_{\pm 2.7}$	$92.3_{\pm 0.2}$	$92.7_{\pm 1.0}$	$93.0_{\pm 0.6}$	$93.8_{\pm 0.4}$	$92.3_{\pm 1.5}$	$94.4_{\pm 0.6}$	$94.6_{\pm 0.6}$	$94.3_{\pm 0.5}$	$94.0_{\pm 1.0}$	$67.5_{\pm 1.4}$	$85.7_{\pm 0.5}$	$86.3_{\pm 1.2}$	$87.9_{\pm 1.4}$	$88.9_{\pm 1.4}$
SC(τ=0.8)	$73.8_{\pm 2.6}$	$91.7_{\pm 1.7}$	$93.3_{\pm 0.5}$	$93.5_{\pm 0.8}$	$94.6_{\pm 0.7}$	$92.5_{\pm 1.1}$	$94.5_{\pm 0.7}$	$94.5_{\pm 0.5}$	$94.1_{\pm 0.9}$	$94.0_{\pm 0.9}$	$72.1_{\pm 2.3}$	$86.5_{\pm 0.8}$	$87.1_{\pm 1.1}$	$88.3_{\pm 1.0}$	$88.2_{\pm 1.2}$
FIRE	$89.9_{\pm 0.5}$	$93.7_{\pm 0.3}$	$93.5_{\pm 0.9}$	$92.9_{\pm 1.2}$	$93.8_{\pm 1.3}$	$91.5_{\pm 0.5}$	$93.9_{\pm 1.1}$	$94.7_{\pm 1.0}$	$94.5_{\pm 1.8}$	$95.1_{\pm 0.7}$	$68.2_{\pm 1.4}$	$87.9_{\pm 1.3}$	$89.9_{\pm 1.7}$	$90.2_{\pm 1.5}$	$90.1_{\pm 1.6}$
CoT-Decoding	$90.8_{\pm 0.8}$	$93.5_{\pm 1.2}$	$94.2_{\pm 0.8}$	$93.8_{\pm 0.8}$	$93.9_{\pm 1.3}$	$91.2_{\pm 1.0}$	$94.4_{\pm 0.5}$	$94.2_{\pm 0.7}$	$94.6_{\pm 1.0}$	$93.4_{\pm 0.9}$	$67.0_{\pm 2.9}$	$89.5_{\pm 0.5}$	$88.7_{\pm 1.3}$	$88.9_{\pm 1.7}$	$90.4_{\pm 1.2}$
RAP	-	$93.6_{\pm 1.1}$	$93.8_{\pm 0.8}$	$93.4_{\pm 0.9}$	$94.1_{\pm 1.1}$	-	$94.3_{\pm 1.8}$	$94.7_{\pm 0.9}$	$94.7_{\pm 1.2}$	$94.8_{\pm 0.7}$	-	$88.7_{\pm 1.2}$	$89.7_{\pm 1.4}$	$89.7_{\pm 1.3}$	$90.7_{\pm 1.0}$
Ours	$\mathbf{93.8}_{\pm 0.4}$	$\mathbf{95.5}_{\pm 0.9}$	$\mathbf{95.5}_{\pm 0.8}$	$\mathbf{95.1}_{\pm 0.7}$	$\mathbf{95.8}_{\pm 1.2}$	$\mathbf{97.0}_{\pm 0.7}$	$\mathbf{96.8}_{\pm 0.3}$	$\mathbf{95.6}_{\pm 0.8}$	$\mathbf{96.0}_{\pm 1.2}$	$\mathbf{97.8}_{\pm 0.5}$	$\mathbf{78.1}_{\pm 0.8}$	$\mathbf{91.2}_{\pm 1.7}$	$\mathbf{90.5}_{\pm 1.4}$	$\mathbf{90.6}_{\pm 1.0}$	$\mathbf{91.0}_{\pm 0.6}$
							StrategyQA								
SC(τ=0.4)	$85.3_{\pm 0.2}$	$85.8_{\pm 1.5}$	$84.7_{\pm 1.0}$	$84.0_{\pm 0.3}$	$85.7_{\pm 1.0}$	$83.5_{\pm 1.3}$	$84.7_{\pm 0.8}$	$85.7_{\pm 1.1}$	$85.0_{\pm 1.7}$	$86.7_{\pm 1.7}$	$73.8_{\pm 1.6}$	$70.5_{\pm 1.1}$	$74.9_{\pm 0.6}$	$84.4_{\pm 0.9}$	$85.4_{\pm 1.9}$
SC(τ=0.6)	$84.1_{\pm 1.7}$	$89.2_{\pm 2.0}$	$86.3_{\pm 2.0}$	$87.1_{\pm 2.3}$	$88.5_{\pm 1.3}$	$83.2_{\pm 1.4}$	$85.4_{\pm 1.4}$	$86.4_{\pm 2.1}$	$85.9_{\pm 1.4}$	$87.5_{\pm 1.3}$	$75.0_{\pm 0.8}$	$73.6_{\pm 1.2}$	$76.9_{\pm 1.0}$	$86.1_{\pm 1.0}$	$88.0_{\pm 1.0}$
SC(τ=0.8)	$86.6_{\pm 2.9}$	$88.9_{\pm 2.7}$	$87.1_{\pm 0.9}$	$88.1_{\pm 0.9}$	$89.6_{\pm 1.2}$	$84.8_{\pm 1.7}$	$84.9_{\pm 2.8}$	$86.9_{\pm 1.3}$	$85.2_{\pm 1.2}$	$88.2_{\pm 1.5}$	$76.0_{\pm 1.4}$	$74.1_{\pm 0.9}$	$77.2_{\pm 1.6}$	$87.8_{\pm 1.2}$	$89.8_{\pm 0.8}$
FIRE	$91.2_{\pm 1.5}$	$92.6_{\pm 1.0}$	$90.1_{\pm 2.1}$	$89.1_{\pm 2.1}$	$90.6_{\pm 1.1}$	$84.4_{\pm 1.6}$	$87.0_{\pm 1.7}$	$88.4_{\pm 2.2}$	$88.1_{\pm 2.3}$	$89.4_{\pm 1.4}$	$74.9_{\pm 1.6}$	$81.0_{\pm 1.0}$	$79.6_{\pm 1.4}$	$86.4_{\pm 1.7}$	$89.7_{\pm 1.5}$
CoT-Decoding	$92.4_{\pm 0.4}$	$93.4_{\pm 1.1}$	$87.4_{\pm 1.4}$	$89.1_{\pm 0.8}$	$90.6_{\pm 1.6}$	$84.7_{\pm 2.3}$	$87.3_{\pm 1.8}$	$86.9_{\pm 1.1}$	$86.8_{\pm 0.8}$	$88.9_{\pm 2.4}$	$75.6_{\pm 0.8}$	$82.2_{\pm 2.8}$	$79.7_{\pm 1.4}$	$87.3_{\pm 1.4}$	$89.0_{\pm 1.3}$
RAP	-	$93.5_{\pm 1.2}$	$89.6_{\pm 1.4}$	$89.3_{\pm 1.5}$	$91.1_{\pm 1.3}$	-	$88.6_{\pm 1.4}$	$87.6_{\pm 0.9}$	$88.6_{\pm 1.2}$	$88.7_{\pm 1.6}$	-	$81.6_{\pm 1.5}$	$79.2_{\pm 1.3}$	$86.6_{\pm 1.2}$	$89.3_{\pm 1.2}$
Ours	$\mathbf{93.7}_{\pm 1.4}$	$\mathbf{94.0}_{\pm 1.3}$	$\mathbf{93.5}_{\pm 1.2}$	$\mathbf{90.9}_{\pm 1.3}$	$\mathbf{93.3}_{\pm 1.8}$	$\mathbf{88.4}_{\pm 0.7}$	$\mathbf{89.0}_{\pm 0.8}$	$\mathbf{89.8}_{\pm 0.8}$	$\mathbf{91.6}_{\pm 0.8}$	$\mathbf{90.4}_{\pm 1.0}$	$\mathbf{81.2}_{\pm 1.3}$	$\mathbf{84.1}_{\pm 0.7}$	$\mathbf{82.4}_{\pm 1.6}$	$\mathbf{87.3}_{\pm 0.9}$	$\mathbf{90.2}_{\pm 1.4}$