Dipper: Diversity in Prompts for Producing Large Language Model Ensembles in Reasoning tasks

Gregory Kang Ruey Lau[*1,2], **Wenyang Hu**[*1], **Diwen Liu**[1], **Jizhuo Chen**[1],
See-Kiong Ng[1], **Bryan Kian Hsiang Low**[1]
[1]Department of Computer Science, National University of Singapore
[2]CNRS@CREATE, 1 Create Way, #08-01 Create Tower, Singapore 138602
{greglau,wenyang,lowkh}@comp.nus.edu.sg, seekiong@nus.edu.sg

Abstract

Large Language Models still encounter substantial challenges in reasoning tasks, especially for smaller models, which many users may be restricted to due to resource constraints (e.g. GPU memory restrictions). Inference-time methods to boost LLM performance, such as prompting methods to invoke certain reasoning pathways in responses, have been shown effective in past works, though they largely rely on sequential queries. The ensemble method, which consists of multiple constituent models running in parallel, is a promising approach to achieving better inference-time performance, especially given recent developments that enabled significant speed-ups in LLM batch inference. In this work, we propose a novel, training-free LLM ensemble framework where a single LLM model is fed an optimized, diverse set of prompts in parallel, effectively producing an ensemble at inference time to achieve performance improvement in reasoning tasks. We empirically demonstrate that our method leads to significant gains on math reasoning tasks, e.g., on MATH, where our ensemble consisting of a few small models (e.g., three Qwen2-MATH-1.5B-it models) can outperform a larger model (e.g., Qwen2-MATH-7B-it).

1 Introduction

While Large Language Models (LLMs) have demonstrated impressive capabilities in addressing a variety of tasks, they still encounter substantial challenges in reasoning tasks such as multi-step logical inference or problem-solving [1]. This is especially so for smaller models, which many users may be restricted to due to resource constraints (e.g. GPU memory restrictions), posing limitations on their utility in practice. Inference-time methods to boost LLM performance, especially for smaller models, hold promise in tackling these challenges [2]. However, many of these methods, such as Chain-of-Thought (CoT), Reflexion, and other techniques [3–6], have focused on *sequential* queries to an LLM to improve performance.

In contrast, ensemble methods, which involve the use of multiple constituent models in *parallel*, have been shown to improve models' performance and robustness in classical machine-learning settings [7] and are promising approaches to achieve better inference-time performance, although less well-studied in the LLM setting. The prospects of applying such methods to LLMs are increasingly attractive, given recent developments that have enabled significant speed-ups in parallel, LLM batch inference. These include methods to efficiently handle key-value cache memory [8] and prompt caching to efficiently reuse common prompts for multiple queries [9, 10], enabling sub-linear (in the number of queries) costs for batch inference.

[*] Equal contribution.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

However, a key challenge in achieving high performing ensembles is how diversity can be appropriately injected among its constituents [11, 12], and this applies to LLM ensembles as well. Recent works have explored how using hetereogenous model ensembles (i.e. consisting of different models types) could lead to improved performance [13, 14], although users may often prefer to or be restricted to using only a single type of LLM model in practice, making such methods not viable in those cases. While a single LLM may be sampled with the same query multiple times and rely on the stochasticity of the LLM response generation process [15] to essentially form a self-ensemble, this approach injects limited diversity to the ensemble which may limit performance improvements.

Instead, significantly more diversity could potentially be injected into an LLM ensemble by making use of LLMs' ability to produce diverse output for a given task with just different prompts. For example, adding simple prompt instructions on how the LLM should reason [16] has been shown to result in performance boosts. This leads to the interesting question of *how we could design ensemble methods that rely on just prompt diversity to produce significant performance boost* for a given LLM. Such ensemble methods could be applied during inference to improve the performance of any LLM (e.g. assessed via APIs), together with other types of inference-time methods.

In this work, we propose DIPPER, a novel, training-free LLM ensemble framework where a single LLM model type is fed an optimized, diverse set of reasoning prompts in parallel, effectively producing an ensemble at inference time to improve performance in reasoning tasks. This approach is simple but surprisingly effective and efficient, and could be implemented with any black-box LLM.

2 Problem setting and related work

LLMs and prompts. Consider an LLM model M which for our purposes can be viewed as a black box that encodes a conditional probability distribution of text responses y over any text input q and additional prompt w, from which we can autoregressively sample responses $\hat{y}$ from, i.e.

$$\hat{y} \sim M(q, w) = p_M(y|q, w). \tag{1}$$

Examples of prompt w could include reasoning prompts such as "Let's think step by step" in CoT [17] that provide instructions on how the LLMs should derive answers for the query q.

LLM ensembles. Ensemble methods involve combining several models to produce a ensemble with better performance and lower variance. However, while commonly applied for a wide variety of machine learning models [7], ensemble methods for LLMs have remained relatively unexplored. Past works have focused on heterogeneous ensembles involving multiple types of models (e.g. different LLM API providers) [13], multi-agent LLM settings that focuses on interactions among agents [18–20], or homogeneous ensembles that rely only on stochastic sampling of model responses [15].

However, to the best of our knowledge, we are not aware of any work that focused on designing and analyzing homogeneous LLM ensembles where their *diversity is injected and optimized via prompts* to constituents with the same underlying LLM model. Our work's focus on such an approach exploits LLMs' unique capabilities of generating diverse output given only changes to its prompts, allowing for a simple but effective method to boost LLM performance using inference-time compute.

Problem formulation. Consider a task $\mathcal{T}$ that consists of instances described as tuples $t \coloneqq (q_t, c_t^*)$, where q_t can be represented as a text string query and c_t^* is the corresponding ground truth solution. We have access to a single LLM model M that when provided task queries and a prompt w, will provide a response $\hat{y}$ according to Eq. (1). This response will consist of (1) some reasoning output $\hat{r}$, and (2) the final answer $\hat{c}$ to the query, which we can denote as $\hat{y} \coloneqq \{\hat{r}, \hat{c}\}$. We evaluate the performance of the model with a specific prompt, denoted as $M(\cdot, w)$, on the task by computing its expected accuracy over the set of task instances $\mathcal{T}$, i.e., $F(M(\cdot, w); \mathcal{T}) \coloneqq \mathbf{E}_{t\sim\mathcal{T}}[\mathbb{I}\{\hat{c}_t = c_t^*\}]$, which in practice is computed over a representative test set.

We denote a homogeneous LLM ensemble as $\mathcal{E}(\cdot\,; M, n, \phi)$, consisting of n instances of the same model M and in general has an adjustable inference-time design parameter ϕ. The ensemble produces a final answer when provided a task query, i.e., $\mathcal{E}(q_t; M, n, \phi) \to \hat{c}_t$, and we can evaluate its performance based on its expected accuracy:

$$F(\mathcal{E}, \mathcal{T}) = \mathbf{E}_{t\sim\mathcal{T}}[\mathbb{I}\{\mathcal{E}(q_t; M, n, \phi) = c_t^*\}]. \tag{2}$$

Our objective is to design an ensemble framework with an appropriate design parameter ϕ such that given fixed M, n and a small labeled development set, we can efficiently maximize Eq. (2) by optimizing for ϕ to produce the best performing ensemble without additional training.

3 Method

Drawing inspiration from how using different prompts w would result in varying response distributions in Eq. (1) given the same model M, our DIPPER framework has the set of prompts $\{w_i\}_{i=1}^n$ fed into the ensemble of n LLM instances as the key ensemble design parameter ϕ. DIPPER consists of the following three components:

1. **Prompt Generator.** First, an LLM generates a large candidate pool of prompts (denoted as $\mathcal{W}$), which can be based on some description of the task and in-context prompt examples that we think may be effective, if such prior knowledge is available. The goal is for the prompts to invoke various types of reasoning pathways when addressing queries, hence injecting diversity into the ensemble. Additional details are in Appendix A.1.
2. **Prompt Selector.** Then, an optimization process is performed over the candidate pool of prompts $\mathcal{W}$ to select a subset of n prompts (i.e., $\{w_i \in \mathcal{W}\}_{i=1}^n$), based on a diversity metric that acts as an approximation of the relative performance of each subset (Section 3.1).
3. **Response Aggregator.** Finally, the responses from the n constituent LLMs are aggregated through a response aggregator operation $\mathcal{A}$ to produce a single final response for the ensemble (Section 3.2).

Putting everything together, our DIPPER framework characterizes an ensemble of size n via $\mathcal{E}(q_t; M, n, \{w_i\}_{i=1}^n) \coloneqq \mathcal{A}(\{M(q_t, w_i)\}_{i=1}^n) \to \hat{c}_t$, where the subset of prompts $\{w_i\}_{i=1}^n$ is chosen from a candidate pool $\mathcal{W}$ to optimize the expected ensemble performance $F(\mathcal{E}, \mathcal{T})$ for a task $\mathcal{T}$.

3.1 Prompt Selector

With our framework, the optimization problem in Eq. (2) reduces to an optimization to choose the best subset of prompts $\{w_i\}_{i=1}^n$ from the set of candidate prompts $\mathcal{W}$:

$$\mathrm{argmax}_{\{w_i \in \mathcal{W}\}_{i=1}^n} F(\mathcal{E}(q_t; M, n, \{w_i\}_{i=1}^n), \mathcal{T}). \tag{3}$$

Unfortunately, directly optimizing Eq. (2) is a combinatorial problem that is very challenging, even if a development/validation set is available for the task of interest. For example, selecting 5 prompts from a candidate pool of 200 prompts involves searching over $\binom{200}{5} \approx 2.5 \times 10^9$ candidates. Instead, we note that the best ensemble composition requires a balance of the two desiderata: fidelity and diversity. Hence, we propose optimizing Eq. (2) by considering how to prioritize the prompts that have the best predicted performance on the task $\mathcal{T}$, while maximizing the diversity of the selected set of prompts.

Prompt fidelity. First, we can approximate the predicted performance of each prompt by its average performance on a task development set $\mathcal{T}_d$[2]. Note that as inference using these various prompts on a small development set can be done in parallel, this process can in practice be significantly sped up by existing batch inference techniques such as those employed by vLLM [8]. Specifically, for a candidate pool of prompts $\mathcal{W}$ and development set $\mathcal{T}_d$, we can define a prompt fidelity mapping $u : \mathcal{W} \to [0, 1]$,

$$u(w) \coloneqq F(M(\cdot, w), \mathcal{T}_d), \tag{4}$$

where $M(\cdot, w)$ is the LLM model conditioned by prompt $w \in \mathcal{W}$, and F the expected accuracy defined in Section 2. In practice, for a candidate pool of size n, $u(w)$ can be represented as an $n \times 1$ column vector, with the elements representing each prompt's expected accuracy.

Semantic entropy. Instead, our approach involves prioritizing the prompts that have the best predicted performance on the task $\mathcal{T}$, while maximizing the diversity of the selected set of prompts. Then, we measure prompt diversity by considering how different the semantic meanings of the n role prompts are from each other. We represent each prompt's semantic meaning with a mapping R from its text representation w into a normalized continuous vector $s \in \mathbb{R}^p$ in a p-dimensional semantic embedding space $\mathcal{S}$ through a sentence embedding model M_s [21], i.e., $R(w) \coloneqq M_s(w)$. This mapping can be represented as an $n \times p$ prompt embedding matrix $R = [s_1, \cdots, s_n]$ where s is a $1 \times p$ row vector representing each prompt.

[2] Without such a development set, an uninformed prior on the performance (e.g. uniform distribution across roles), or an informed-prior based on domain knowledge, could also be used.

To quantify prompt diversity of a given set of prompts, we propose to compute the volume enclosed by the selected prompts in semantic space. Intuitively, for n fixed prompts, more diverse prompts point to more varied directions in semantic space, and enclose larger volume. Specifically, we define the semantic volume metric V as

$$V := \log\det(RR^T), \tag{5}$$

where we take the logarithm (for numerical stability) of the Gram matrix determinant[3].

Fidelity-adjusted semantic volume. To incorporate the prompts' expected accuracy information, we can compute the performance-adjusted prompt embedding matrix,

$$\tilde{R} := \exp(\frac{\alpha}{2}\operatorname{diag}(u))R, \tag{6}$$

where $\operatorname{diag}(u)$ is the diagonal matrix with its i^{th} diagonal element being the corresponding element u_i. This essentially scales each row s_i in R by an exponential factor based on its corresponding predicted accuracy, $\exp(\frac{\alpha}{2}u_i)$, where α is a scalar hyperparameter influencing the balance between diversity and expected performance. Intuitively, prompts with higher expected accuracy would then be able to support larger semantic volume and hence be prioritized for inclusion into the ensemble. The adjusted embedding matrix can then be used to compute the semantic volume in Eq. (5).

Optimization of semantic entropy. We can now recast Eq. (2) as an optimization of the fidelity-adjusted semantic volume metric $\tilde{V}$ evaluated over the set of candidate prompts. Note that instead of the expected ensemble performance $F(\mathcal{E})$, which is an objective that can only be optimized by blackbox optimization methods like Bayesian Optimization [22], our metric V can be approximated by efficient, well-established heuristics.

Specifically, as the semantic volume metric is submodular, we can optimize for the best subset of roles by incrementally building the subset with a greedy approach up to the desired size n and still be guaranteed a good approximation [23]. This allows us an efficient and theoretically-inspired approach to obtain the best ensemble prompts. Our full algorithm is outlined in Algorithm 1 in the Appendix.

3.2 Response Aggregator

Given the various constituent LLMs' responses, the aggregation method determines how much information is used to derive the final ensemble output. We consider two approaches:

Majority voting (MV). The first involves extracting the final answer $\hat{c}$ from each LLM response $\hat{y} = \{\hat{r}, \hat{c}\}$, and then selecting the answer that has been proposed the most number of times. This approach does not take into account the reasoning $\hat{r}$ output produced by the ensemble constituents, but is easily implementable.

LLM aggregation (LLMA). The second involves using another LLM instance to evaluate each constituent response, aggregate them, and generate a final answer for the task. This approach incurs additional LLM query cost and is dependent on the capabilities of the aggregator LLM, but has the advantage of potentially taking into account the various reasoning output $\hat{r}$ from the ensemble constituents to further improve overall performance (see Section 4.3 for details).

4 Experiments

Experimental set-up. We empirically evaluate our framework on mathematically reasoning tasks with the MATH [24], GSM8K, and MMLU-STEM datasets. We implement our framework by using the GPT-4o as our prompt generator and Qwen2-MATH-1.5B as the constituent model in the ensemble, where the ensemble constituents are run in parallel using vLLM [8] for fast batch inference. Further details of our experiments are in Appx. B.

Baselines. We evaluate our DIPPER framework by comparing it against the "Self-ensemble" baseline, which lacks prompt diversity but incorporates diversity through repeated response sampling [15]. We also compare our DIPPER implementation based on semantic volume ("Dipper") with two other variants: (1) a naïve implementation where prompts are sampled from the candidate pool based on

[3] We omit a factor of 2 which does not affect the optimization process. For our setting, we also have $n < p$ as the semantic embedding space is usually high dimensional.

their validation accuracy distribution ("Random+"), and (2) an ensemble using the "Top-n" prompts as evaluated on the validation set, which benefits from the diversity of prompts introduced by our prompt generation process but do not explicitly optimize for ensemble diversity otherwise.

4.1 Ensembles with fixed prompt methods

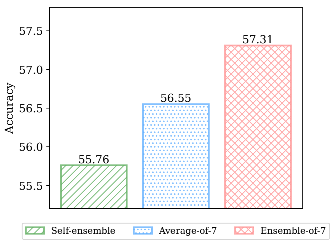

Figure 1: Comparison of different ensembles of 7 reasoning prompts on MATH.

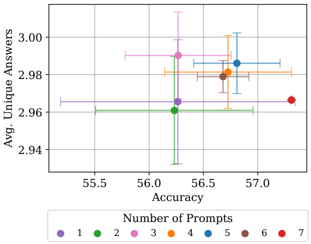

Figure 2: Plot of accuracy vs. average unique answers with ensembles of different numbers of prompts.

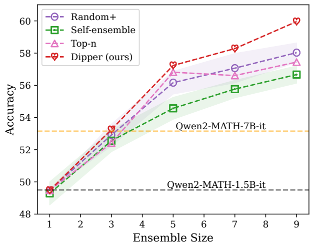

Figure 3: Comparison of different ensemble methods on MATH.

First, we demonstrate the effectiveness of our prompt-based ensemble method by considering a fixed list of 7 reasoning prompts inspired by existing works [25–27] on prompting methods to boost reasoning capabilities (details in Appx. B.1). Under a fixed ensemble size of 7, Fig. 1 shows that the ensemble using the 7 different prompts (57.31%) significantly outperforms the self-ensemble with no prompt ((55.76%)) and the average performance (56.55%) of self-ensemble using any single prompt.

To further investigate the impact of prompt diversity, we evaluated all combinations of the 7 prompts while maintaining a fixed ensemble size of 7. For combinations with fewer than 7 prompts, we randomly sampled responses to reach a total of 7 before applying majority voting. The result in Fig. 2 reveals that increasing the number of prompts in the ensemble generally leads to higher accuracy, reduced variance, and fewer unique answers. Especially, the 7-prompt ensemble shows the highest accuracy and lowest variance, which suggests that employing a diverse set of prompts in an ensemble can enhance performance and consistency, particularly when we do not know which prompt would perform best before evaluation.

4.2 Ensembles with optimized prompt diversity

Next, we consider our full DIPPER framework. We first generate a pool of prompt candidates ($|\mathcal{W}| = 200$) using the 7 reasoning prompts in the previous section as in-context exemplars (details in Appx. B.1) and then perform diversity optimization (Sec. 3.1) to select the best ensemble prompts. We refer the evaluation details to Appx. B.2. As shown in Fig. 3, our method achieves the highest accuracy compared to all baseline ensemble methods across various ensemble sizes. DIPPER also significantly outperforms the single LLM. For example, DIPPER with $n = 9$ has close to a 10%-pt increase (~20% accuracy gain) compared to the single LLM baseline. In fact, our ensemble that consists of just 3 Qwen2-MATH-1.5B model already slightly outperform the next model size class, the Qwen2-MATH-7B model. See more results on GSM8K and MMLU-STEM in Appx. C.2 where DIPPER is shown to be consistently effective.

4.3 LLM aggregation can do better

Finally, we analyze the effects of using Majority voting (MV) or LLM aggregation (LLMA) for our response aggregator component (see experimental details in Appx. B.3). We consider ensembles of size $n = 5$ with randomly selected prompts, and compare their performance on MATH when using either majority voting or LLM aggregation. Table 1 summarizes the results, showing that LLMA is more accurate than MV on average (i.e., higher $F(\mathcal{E})$). To better analyze the performance difference, we computed the "Override Ratio" which is how often a specific method is correct when the two methods disagree. Note that when MV and LLMA disagree, LLMA has a much higher ratio than MV which is only correct 8% of the time. We attribute LLMA's advantage to its capability of understanding the reasoning $\hat{r}$ in responses even when the ensembles do not have a majority for the final answers $\hat{c}$. This is corroborated when we look at the number of unique answers $|C|$ when only one specific method is accurate: $|C|$ for LLMA is higher than that of MV, which suggests that LLMA performs better than MV when the ensemble produces more unique answers, as expected.

Method	$F(\mathcal{E})$	Override Ratio	$\|C\|$
MV	56.63	0.08	3.16
LLMA	64.87	0.29	3.70

Table 1: Comparison between MV and LLMA. $F(\mathcal{E})$ is the test performance. Override ratio is how often a specific method is correct when the two methods disagree. $|C|$ is the number of unique answers when only one specific method is accurate.

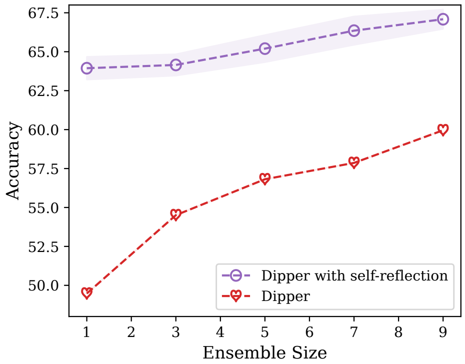

Figure 4: Comparison of naïve DIPPER and DIPPER with self-reflection on MATH.

4.4 DIPPER is a stackable framework

In addition, we also show that our ensemble framework DIPPER is orthogonal to other established prompting techniques (e.g. CoT and Reflexion [6]), allowing it to stack and bring greater performance. In our experiments, we first use DIPPER to select n agents and query each agent with the questions. Their initial responses will be self-reflected according to the method proposed in Reflexion [6], before being aggregated into the final answer with MV. The result in Fig. 4 shows that DIPPER coupled with reflection achieves much better results, suggesting that DIPPER has the potential to be extended further or combined with other methods.

5 Conclusion

In this work, we have proposed a novel framework, DIPPER, where a single LLM model type is fed an optimized, diverse set of reasoning prompts in parallel, effectively producing an ensemble at inference time to achieve performance improvement in reasoning tasks. Our empirical findings have demonstrated DIPPER's effectiveness in improving inference performance for a variety of reasoning tasks, which may inspire future works to investigate additional optimization methods for prompt-based inference-time ensembles to further improve performance gains.

Acknowledgments

This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG2-PhD/2023-01-039J). This research is part of the programme DesCartes and is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. This research is supported by the National Research Foundation Singapore and the Singapore Ministry of Digital Development and Innovation, National AI Group under the AI Visiting Professorship Programme (award number AIVP-2024-001).

References

[1] Jie Huang and Kevin Chen-Chuan Chang. Towards Reasoning in Large Language Models: A Survey, May 2023.

[2] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters, August 2024.

[3] Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. Reasoning with Language Model Prompting: A Survey, September 2023.

[4] Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. Progressive-Hint Prompting Improves Reasoning in Large Language Models, August 2023.

[5] 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, December 2023.

[6] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[7] M. A. Ganaie, Minghui Hu, A. K. Malik, M. Tanveer, and P. N. Suganthan. Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115:105151, October 2022. ISSN 0952-1976. doi: 10.1016/j.engappai.2022.105151.

[8] 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.

[9] Hanlin Zhu, Banghua Zhu, and Jiantao Jiao. Efficient Prompt Caching via Embedding Similarity, February 2024.

[10] In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt Cache: Modular Attention Reuse for Low-Latency Inference, April 2024.

[11] Anders Krogh and Jesper Vedelsby. Neural Network Ensembles, Cross Validation, and Active Learning. In *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994.

[12] Sheheryar Zaidi, Arber Zela, Thomas Elsken, Chris Holmes, Frank Hutter, and Yee Whye Teh. Neural Ensemble Search for Uncertainty Estimation and Dataset Shift. https://arxiv.org/abs/2006.08573v3, June 2020.

[13] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion, June 2023.

[14] Yichong Huang, Xiaocheng Feng, Baohang Li, Yang Xiang, Hui Wang, Bing Qin, and Ting Liu. Ensemble Learning for Heterogeneous Large Language Models with Deep Parallel Collaboration, May 2024.

[15] 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 *The Eleventh International Conference on Learning Representations*, 2023.

[16] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large Language Models are Zero-Shot Reasoners, January 2023.

[17] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, January 2023.

[18] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving Factuality and Reasoning in Language Models through Multiagent Debate, May 2023.

[19] Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. Dynamic LLM-Agent Network: An LLM-agent Collaboration Framework with Agent Team Optimization. October 2023.

[20] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors, October 2023.

[21] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[22] Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023.

[23] George Nemhauser, Laurence Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14:265–294, 12 1978. doi: 10.1007/BF01588971.

[24] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[25] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models, March 2023.

[26] Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. Rephrase and respond: Let large language models ask better questions for themselves. *arXiv preprint arXiv:2311.04205*, 2023.

[27] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[28] Qwen Team. Introducing Qwen2-Math. `https://qwenlm.github.io/blog/qwen2-math/`, 2024.

A Additional details on the DIPPER framework

A.1 Prompt Generator

The first component plays the important role of generating a large pool of candidate prompts with the following desiderata:

1. **Fidelity.** Each prompt should be able to influence the LLM into applying a certain type of reasoning approach to the task, and not have significant negative impact the LLM's performance on the task.

2. **Diversity.** The prompts should be sufficiently different from one another such that they elicit various reasoning pathways and provide a diverse pool to select from in the subsequent component.

We first show that LLMs are capable of generating prompts that meet this desideratum, via the most direct way of prompting it to generate a pool of candidate prompts while providing it with exemplars illustrating different reasoning prompts. To do so, we considered a list of 7 reasoning prompts inspired by existing works [25–27] on prompting methods to boost reasoning capabilities. Given these prompts as exemplars, we used GPT-4o to further generate a set of 200 different candidate prompts that each represent a different reasoning approach (details in Appx. B.1). Fig. 5 shows the distribution of average accuracy over a sampled test set of MATH [24] questions for each prompt, when used for the Qwen2-MATH-1.5B model (i.e., $F(M(\cdot, w); \mathcal{T})$ for $w_i \in \mathcal{W}$). Note that the distribution of accuracy is largely higher than that of the base model without prompts, and similar to the accuracies achieved by the reasoning prompt exemplars, demonstrating the fidelity requirement. Qualitatively, we see that the prompts are also relatively diverse – they generally specify certain reasoning approaches inspired by various subject domains (see Appx. **??**). We will quantify this diversity in Sec. 3.1 with our proposed metric.

Note that when generating the prompts, we did not pass any task description to the LLM prompt generator. We did so as the reasoning prompts can in general be task-agnostic, even if some may be inspired by some specific subject matter. In practice, the candidate pool of reasoning prompts need not be generated on-the-fly, but be drawn from a shared pool prepared beforehand by a more powerful LLM, to be used by ensembles consisting of much smaller LLMs, as we demonstrated. The actual selection of relevant prompts can be done by the prompt selector component, which we will described next in Sec. 3.1.

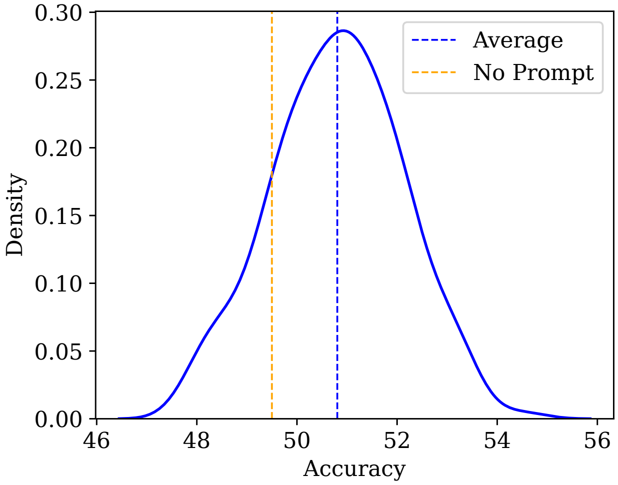

Figure 5: The accuracy distribution of 200 candidate prompts on MATH.

Algorithm 1 DIPPER semantic volume algorithm

```
Input: LLM model $M$, Initial candidate prompt set $\bar{\mathcal{W}}$, Semantic embedding model $M_s$,
    Development set $\mathcal{T}_d$, Ensemble size $n$, Fidelity-diversity hyperparam $\alpha$
Output: Ensemble prompt set $\mathcal{Z}$
$\mathcal{Z} \leftarrow \{\ \}$
$\bar{u}(w) \leftarrow [F(M(\cdot, w_i), \mathcal{T}_d)$ for $w_i \in \bar{\mathcal{W}}]$
$\mathcal{Z} \leftarrow \mathcal{Z} \cup \arg\max_w \bar{u}(w)$
$\mathcal{W} \leftarrow \bar{\mathcal{W}} \setminus \arg\max_w \bar{u}(w)$
for $j = 1, \ldots, n$ do
    $\tilde{\mathcal{V}} \leftarrow [\ ]$
    for $w_k \in \mathcal{W}$ do
        $\mathcal{P} \leftarrow \mathcal{Z} \cup w_k$
        $u(w) \leftarrow [F(M(\cdot, w_i), \mathcal{T}_d)$ for $w_i \in \mathcal{P}]$
        $R(w) \leftarrow [M_s(w_i)$ for $w_i \in \mathcal{P}]$
        $\tilde{V}_{w_k} \leftarrow \log\det(\exp(\alpha \operatorname{diag} u) R R^T)$
        $\tilde{\mathcal{V}}(w) \leftarrow [\tilde{\mathcal{V}}(w), \tilde{V}_{w_k}]$
    end for
    $\mathcal{Z} \leftarrow \mathcal{Z} \cup \arg\max_w \tilde{\mathcal{V}}(w)$
    $\mathcal{W} \leftarrow \mathcal{W} \setminus \arg\max_w \tilde{\mathcal{V}}(w)$
end for
return $\mathcal{Z}$
```

B Detailed Experimental Setting

B.1 Fixed 7 prompts and Prompt Generation

We consider 7 prompts inspired by existing works and list them in Tab. 2 below.

Table 2: The table of 7 basic reasoning prompts inspired by existing works.

Prompt
Let's think step-by-step to find the answer.
Reflect on the question carefully before answering.
Rephrase the question in your own words before responding.
Actively reason through the question and answer each part systematically.
Answer this question as a scientist would.
Eliminate the obviously incorrect answers first and then choose the most likely correct answer.
Analyze the context of the question and use relevant information to derive the answer.

We use the prompt template in Tab. 3 to generate 200 diverse prompts.

Table 3: The prompt template for generating more reasoning prompts based on the 7 prompts.

Prompt Generation Template
Here are some instruction examples: {7 reasoning prompts} Study the above examples and brainstorm 200 similar instructions with detailed descriptions of different reasoning behaviors that are helpful for reasoning. Those 200 proposed instructions should be diverse enough.

B.2 Evaluation

We primarily consider three datasets in our paper. For MATH, we randomly 10% test samples from each category and form a fixed subset of size 500. We uniformly randomly sample 20 samples from this subset to form a validation dataset and use the rest 480 samples as the hold-out test dataset. For

GSM8K and MMLU-STEAM, we use their official split of test data and uniformly randomly sample 20 samples to form a validation dataset for each task, and use the rest samples as the hold-out test data.

In the inference evaluation, we use 4-shot exemplars for MATH, 8-shot for GSM8K, and 5-shot for MMLU-STEM. Those exemplars are adopted from the evaluation setting in Qwen2-MATH [28] and fixed for all questions and all methods.

B.3 LLM aggregation

We perform LLM aggregation with the same Qwen2-MATH-1.5B-it model, by feeding the question context and the responses from LLM agents into the designed template shown below in the bounding box:

Table 4: The prompt template for LLM aggregation.

⟨**System Prompt**⟩: You are a helpful assistant. You do not directly answer a question, but examine the reasoning and correctness of responses from different experts and provide a final answer.

The question is:
⟨**QUESTION**⟩

There are some responses:
⟨**RESPONSES**⟩

Examine those responses and provide the final answer.

C Additional Results

C.1 Performance-adjusted embedding

To study the effect of accuracy on the performance-adjusted prompt embedding matrix, we report the Spearman correlation between logdet V and the ensemble performance $F(\mathcal{E})$ under different choices of α. We observe that when $\alpha = 0$, the correlation is 0.18, and it increases as α becomes larger. The positive correlation justifies our approach to maximize prompt diversity. The increasing correlation justifies our approach of incorporating validation accuracy into the prompt semantic embedding.

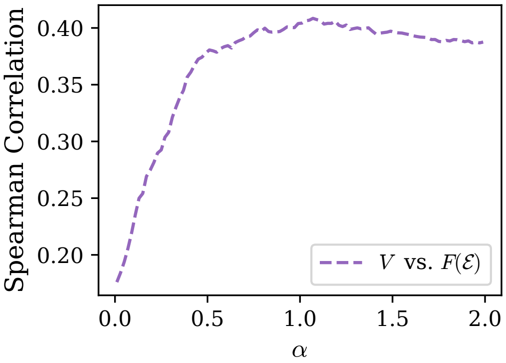

Figure 6: Spearman correlation between diversity V and ensemble accuracy $F(\mathcal{E})$ on MATH.

C.2 Results on More Datasets

We also evaluate the performance of DIPPER on GSM8K and MMLU-STEM. The results in Fig. 7 and Fig. 8 demonstrate that our proposed method DIPPER can achieve superior or comparable results against the strongest baseline, which suggests that diversity optimization is a promising approach to improve inference performance.

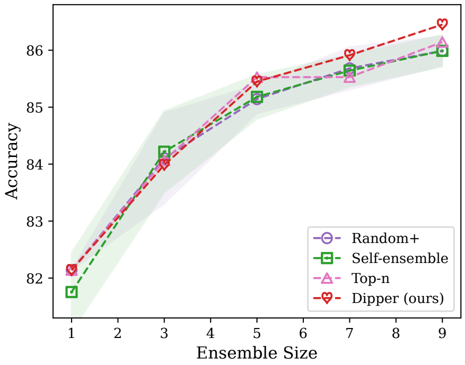

Figure 7: Comparison of different ensemble methods on GSM8K.

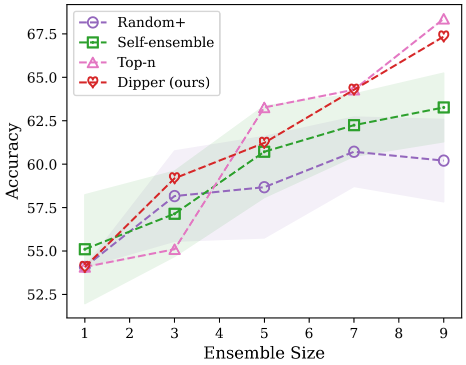

Figure 8: Comparison of different ensemble methods on MMLU-STEM.