G-Boost: Boosting Private SLMs with General LLMs

Anonymous ACL submission

Abstract

Due to the limited computational resources, most Large Language Model (LLM) developers can only fine-tune Small Language Models (SLMs) on their own data. However, these private SLMs typically have limited effectiveness. To enhance the performance of private SLMs, this paper proposes to ask general LLMs for help. The general LLMs can be APIs or larger LLMs whose inference cost the developers can afford. Specifically, we propose the G-Boost framework, in which a private SLM adaptively performs collaborative inference with a general LLM under the guidance of process reward. Experiments demonstrate that our framework can significantly boost the performance of private SLMs.

1 Introduction

Large Language Models (LLMs) have achieved remarkable performance across various natural language processing tasks (OpenAI, 2023; Yang et al., 2024; DeepSeek-AI et al., 2025). They demonstrate exceptional generalization and reasoning capabilities, making them effective in a wide range of applications. However, these capabilities come at a significant cost. Fine-tuning LLMs for specific domains often demands significant computational resources, which only a few organizations can afford. Additionally, data privacy concerns further prevent organizations from fully relying on external LLM services.

Due to resource constraints and privacy concerns, many developers can only fine-tune small language models (SLMs) on their private datasets. These fine-tuned SLMs can often adapt well to specific domains and sometimes even outperform general LLMs in domain-specific tasks. However, as shown in Fig. 1. limited by smaller parameter sizes, SLMs exhibit weaker general language capabilities. They often struggle with queries that require complex reasoning or deep understanding.

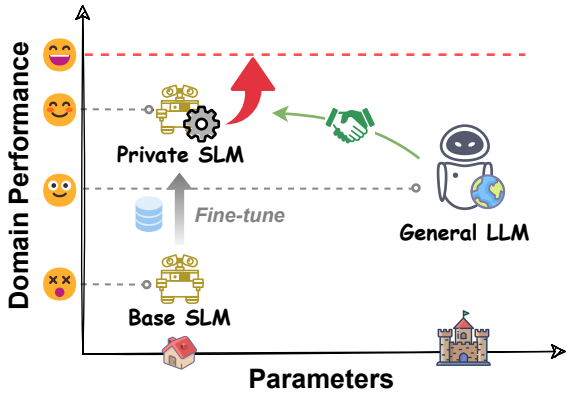

Figure 1: This figure highlights the performance constraints of private SLMs in domain-specific adaptation. Private SLMs are limited by their small parameter size, whereas general LLMs lack domain-specific expertise. G-Boost integrates their complementary strengths to enhance the performance of private SLMs.

Recently, various collaborative inference methods have been proposed to overcome the performance limitations of small models. A widely adopted approach involves combining small and large models to perform hybrid inference (Ong et al., 2024; MS et al., 2024; Zheng et al., 2025), where the two models dynamically alternate to generate outputs. This technique achieves near-LLM performance while reducing computational costs. Another strategy follows a two-stage process, where LLMs provide guidance and small models refine the predictions (Zhan et al., 2025; Yao et al., 2024). While effective to some extent, these methods often rely on the assumption that LLMs consistently outperform SLMs. However, in domain-specific tasks, general LLMs frequently lack the specialized knowledge required, which limits their ability to enhance private SLMs.

Researchers have also explored the integration of multiple LLMs to leverage the combined expertise of individual models, thereby improving

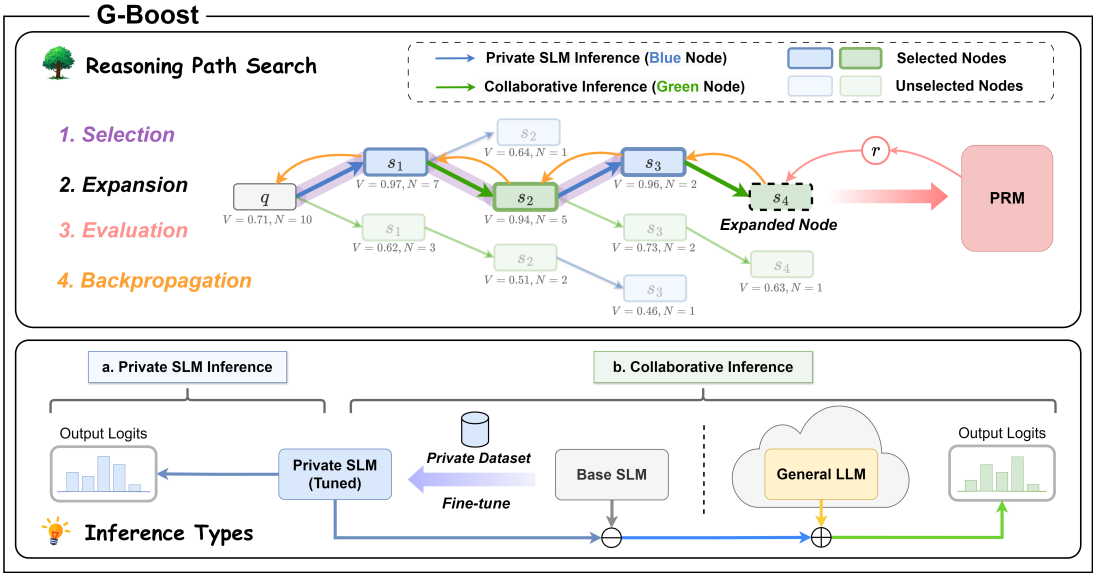

Figure 2: Overall Framework of G-Boost: G-Boost comprises two reasoning modes: Private SLM Inference and SLM-LLM Collaborative Inference. Guided by a process reward model, it employs tree search to adaptively integrate these modes, dynamically optimizing reasoning paths.

their cross-domain performance (Chen et al., 2024; Mohammadshahi et al., 2024; Jiang et al., 2023). Nevertheless, this approach yields limited benefits in single-domain tasks and rarely surpasses the capabilities of specialized domain experts. As a result, such methods also fail to significantly enhance domain-specific SLMs.

We propose the G-Boost framework, where the private LLM collaborates with the general LLM under the guidance of a Process Reward Model (PRM) (Lightman et al., 2024). This framework combines the domain knowledge of the private LLM with the broad language capabilities of the general LLM, thereby enhancing the private LLM's performance on domain-specific tasks. Specifically, the framework uses a logits fusion approach to integrate the private LLM's domain expertise with the general LLM's broad knowledge, enabling collaborative reasoning. Meanwhile, the PRM evaluates the logical consistency and quality of intermediate reasoning steps. Based on the reward value, Monte Carlo Tree Search (MCTS) (Coulom, 2006) is applied to dynamically optimize the reasoning path. By continuously adjusting the collaboration strategy during reasoning, G-Boost effectively leverages the strengths of both the private and general LLMs, enabling precise and reliable exploration of reasoning paths.

To validate the effectiveness of the proposed G-Boost framework, we conducted experiments on widely recognized benchmarks, including GSM8K (Cobbe et al., 2021) and MATH-500 (Lightman et al., 2024). The results demonstrate that G-Boost consistently enhances the performance of private SLM, surpassing both general LLMs and static collaborative inference methods. These findings highlight the potential of adaptive collaboration between private SLM and general LLM to achieve superior performance in domain-specific tasks.

2 Related Works

Collaborative Inference Between SLM and LLM. Recent research in collaborative inference between small and large language model has made significant progress, aiming to balance efficiency and performance. Routing mechanism is a common approach to dynamically select between SLM and LLM during inference. Query-level routing allocates the entire query to either SLM or LLM based on query complexity (Ong et al., 2024; Ding et al., 2024; Aggarwal et al., 2024), thereby reducing inference costs. However, this approach exhibits limited flexibility for fine-grained tasks. To further enhance efficiency, token-level routing assigns only critical tokens to LLM, while SLM generates the remaining tokens more efficiently (Zheng et al., 2025). Another category of methods is cascaded inference, in which LLM generates initial drafts or guiding prompts, and SLM refines or further elaborates on them (Zhan et al., 2025; Yao et al., 2024). This two-stage inference reduces cost while utiliz-

ing the generative capabilities of LLM. In contrast, speculative decoding (Leviathan et al., 2023; Chen et al., 2023a) accelerates inference by allowing SLM to generate draft tokens, which LLM subsequently verifies collectively in a single forward pass. This approach guarantees outputs identical to those of the original model, with no loss in effectiveness. Additionally, some methods enable black-box LLM adaptation through collaborative inference. For instance, CombLM (Ormazabal et al., 2023) and Proxy-Tuning (Liu et al., 2024) transfer domain-specific knowledge from SLM to LLM via logit arithmetic. While this improves domain performance to some extent, it can also introduce new bias, and the overall effectiveness remains limited. Existing collaborative inference methods between small and large models primarily focus on accelerating inference, aiming to achieve near-LLM performance at reduced costs. However, these methods often assume that LLM consistently outperforms SLM. In domain-specific tasks, general LLMs often lack the required expertise, limiting their ability to effectively enhance private SLMs.

Multi-LLM Collaboration. Multi-LLM collaboration focuses on integrating the expertise of multiple LLMs to enhance performance in cross-domain or complex tasks. Unlike collaborative inference between small and large models, the primary motivation here is to leverage the complementary capabilities of multiple LLMs rather than optimizing costs. A key strategy in multi-LLM collaboration is routing mechanisms, which dynamically assign tasks to the most appropriate LLMs based on their capabilities. For example, RouterDC (Chen et al., 2024) trains a router using dual contrastive learning to select the best LLM for each query, while Routoo (Mohammadshahi et al., 2024) builds complementary model ensembles by predicting model performance. Similarly, (Chai et al., 2024) represents expert LLMs as tokens in a meta-LLM vocabulary, enabling dynamic routing to specialized LLMs during generation. In addition to routing, cascaded inference is often employed, where LLMs are invoked sequentially based on task complexity or resource constraints. For instance, FrugalGPT(Chen et al., 2023b) first uses lighter models for simpler tasks and escalates to more powerful LLMs only when necessary, ensuring efficiency while maintaining performance. Another strategy is model ensembling, which combines outputs of multiple LLMs to improve overall performance. For example, LLM-Blender (Jiang et al., 2023) ranks outputs of different LLMs and combines the top-ranked ones to produce higher-quality results.

3 Problem Setup

Consider a domain-specific task with a private training dataset D_p. Due to limited computational resources, the user fine-tunes an open-source SLM π_s^- locally on dataset D_p, resulting in a domain-adapted private SLM π_s^+. Both π_s^- and π_s^+ are deployed on the edge device. While the private SLM exhibits strong domain-specific capabilities, its language understanding and generalization abilities are limited due to its smaller parameter size.

To enhance the performance of the private SLM, the user can leverage a general LLM π_l, which shares the same vocabulary as the private SLM π_s^+. The general LLM π_l may be accessed via an inference API or deployed locally if computational resources permit. It provides strong in-context understanding capabilities but lacks domain-specific knowledge. The goal of our framework is to enable adaptive collaboration between the private SLM π_s^+ and the general LLM π_l to achieve better performance on domain-specific tasks.

4 Methodology

To leverage general LLM to enhance private SLM, we propose the G-Boost framework. It dynamically integrates the domain-specific expertise of private SLMs with the general language capabilities of LLMs, enabling collaborative reasoning for complex queries.

We model the collaborative reasoning process between the private SLM and the general LLM as a search problem in a tree-structured space. In this tree, the root node corresponds to the input query, while the remaining nodes represent individual reasoning steps. To define these steps, we divide the reasoning process into fixed-length segments of size L. A path from the root node to a leaf node represents a reasoning path up to a specific step. When a reasoning path reaches a final answer, the corresponding leaf node is designated as a termination node, marking the end of the reasoning process. Besides, each edge denotes an inference action, either by the private SLM alone or collaboratively with the general LLM. The objective of the framework is to construct an optimal reasoning path by dynamically selecting the most appropriate inference action at each reasoning step, thereby effectively combining the strengths of the private

SLM and the general LLM.

Specifically, the G-Boost framework employs MCTS to explore the collaborative reasoning space, systematically searching for the optimal sequence of reasoning actions. To guide MCTS, we introduce the PRM, which provides fine-grained feedback on the logical consistency and task relevance of intermediate reasoning steps. During the search process, PRM evaluates the quality of newly generated reasoning steps, enabling MCTS to prioritize promising reasoning paths while avoiding flawed paths. Under the guidance of process reward, the system dynamically refines the search tree. For a given query q, the iterative reasoning process builds a search tree, with each iteration consisting of four steps: selection, expansion, evaluation, and backpropagation.

4.1 Selection

At the start of each iteration, G-Boost identifies the most promising reasoning path to extend within the collaborative reasoning tree, where the private SLM π_s^+ and the general LLM π_l work together to address q. Starting from the root node, the system traverses the tree by iteratively selecting child nodes based on the Upper Confidence Bounds applied to Trees (UCT) algorithm (Kocsis and Szepesvári, 2006). This policy balances the exploitation of high-quality reasoning paths with the exploration of underexplored ones.

The UCT value of a node s is computed as:

$$\mathrm{UCT}(s) = V_s + C\sqrt{\frac{\ln N_{\text{parent}(s)}}{N_s}}, \quad (1)$$

where V_s is the value of the node (accumulated in the evaluation stage), N_s is the visit count of the node, $N_{\text{parent}(s)}$ is the visit count of its parent node, and C is the exploration constant that balances exploration and exploitation.

The trajectory from the root node to the current leaf node is defined as:

$$\tau = \{s_0, s_1, \ldots, s_d\}, \quad (2)$$

where s_0 is the root node and s_d is the current leaf node. Among the nodes in this trajectory, the system identifies candidate nodes for expansion based on whether they have unexpanded reasoning actions:

$$S_{\text{candidate}} = \{s \mid s \in \tau \text{ and } s \text{ has unexpanded actions}\}. \quad (3)$$

The node with the highest value from this set is selected for expansion:

$$s_k = \underset{s \in S_{\text{candidate}}}{\arg\max} V_s, \quad (4)$$

where k represents the index of the step in the reasoning path. By focusing on nodes with high potential values, the system ensures that promising reasoning paths are prioritized, while still exploring less-visited paths to avoid missing potential high-quality solutions.

4.2 Expansion

After selecting the leaf node s_k, the algorithm expands the search tree by generating a new reasoning step. The newly generated child node is denoted as s_{k+1}, which introduces new reasoning content and extends the search tree. We consider two modes for next step generation: **SLM-LLM collaborative inference** and **Private SLM inference.**

4.2.1 SLM-LLM Collaborative Inference

SLM-LLM collaborative inference leverages the complementary strengths of the private SLM and general LLM to enhance reasoning quality. The private SLM provides domain-specific expertise, while the general LLM contributes broad language capabilities. This collaboration aims to overcome the limitations of individual models, balancing generalization and specialization for more accurate reasoning.

Inspired by Proxy-Tuning (Liu et al., 2024), we propose a logit fusion strategy to integrate the output distributions of the private SLM and the general LLM. At each decoding step, the system refines the general LLM's logits by incorporating the logit offsets derived from the fine-tuned private SLM and its base version. This refinement aligns the general LLM's predictions with the domain-specific knowledge of the private SLM, while retaining the general LLM's robust in-context reasoning and generalization capabilities.

In this mode, the user invokes the general LLM π_l. The system generates step s_{k+1} by combining the logit distributions of the private SLM π_s^+ and the general LLM π_l. The probability distribution

for the next token is given by:

$$\begin{aligned}\tilde{P}(x_t \mid q, & s_{1:k}, s_{k+1}^{<t}) \\ &= \text{softmax}\Big(z_c(x_t \mid q, s_{1:k}, s_{k+1}^{<t}) \\ &\quad + z_e^+(x_t \mid q, s_{1:k}, s_{k+1}^{<t}) \\ &\quad - z_e^-(x_t \mid q, s_{1:k}, s_{k+1}^{<t})\Big),\end{aligned} \tag{5}$$

where $s_{k+1}^{<t}$ denotes the tokens already generated for step s_{k+1} up to position t.

The probability of generating the complete step s_{k+1} is computed as:

$$\tilde{\pi}(s_{k+1} \mid q, s_{1:k}) = \prod_{t=1}^{L} \tilde{P}(x_t \mid q, s_{1:k}, s_{k+1}^{<t}). \tag{6}$$

The next step s_{k+1} is sampled from this distribution:

$$s_{k+1} \sim \tilde{\pi}(\cdot \mid q, s_{1:k}), \tag{7}$$

where s_{k+1} represents the reasoning step generated exclusively by the collaborative probability distribution, based on the query q and the sequence of previously generated steps $s_{1:k}$.

4.2.2 Private SLM Inference

While the collaborative inference aims to leverage the complementary strengths of both models, the general LLM π_l often lacks domain-specific knowledge, which can lead to errors or reasoning instability. To address this, we incorporate private SLM inference as an alternative to generate the next step, relying solely on the fine-tuned SLM for independent reasoning.

In this mode, the private SLM π_s^+, fine-tuned on the user's private dataset D_p, generates reasoning steps without involving the general LLM. While limited by its smaller model size, this approach avoids errors introduced by the general LLM and ensures stability in the reasoning process when collaborative inference proves unreliable, enriching the overall search process. The next reasoning step in this mode are generated according to the following distribution:

$$s_{k+1} \sim \pi_s^+(\cdot \mid q, s_{1:k}). \tag{8}$$

4.3 Evaluation

The evaluation stage estimates the value of the newly expanded node, assessing its potential quality within the reasoning process. In this work, we replace traditional rollout methods with PRM-based evaluation, which provides a more efficient and accurate assessment of the expanded node. Unlike rollouts, which simulate entire reasoning paths, PRM directly evaluating intermediate reasoning steps. This approach not only enhances the stability and accuracy of the evaluation but also reduces runtime overhead. Due to PRM's gradual adoption as a standard model for reasoning evaluation, we utilize an open-source PRM to evaluate the newly expanded node s_{k+1}. The PRM is specifically fine-tuned to deliver granular feedback for each reasoning step. It takes the query q and the sequence of reasoning steps $s_{1:k+1}$ as input. The reward for node s_{k+1} is then calculated as follows:

$$r = \text{PRM}(q, s_{1:k+1}), \tag{9}$$

Compared to traditional rollout methods, PRM directly evaluates intermediate reasoning steps in a fine-grained manner. This avoids randomness and cumulative error from simulating full paths, providing more stable and efficient quality feedback.

4.4 Backpropagation

After getting the reward, we propagate it from the newly expanded node to the root node, updating the statistical information of all nodes along the path. These updates refine the search tree and enable more accurate decision-making in future searches. Specifically, starting from the expanded node s_{k+1}, the reward r is backpropagated along the path to the root node. For a node s on the path, its value V_s is updated using the following formula:

$$V_s = \frac{(N_s - 1)V_s + r}{N_s}, \tag{10}$$

and its visit count is updated as:

$$N_s \leftarrow N_s + 1. \tag{11}$$

This update process ensures that the search gradually focuses on high-value paths, improving the overall quality of the reasoning path over time.

By leveraging PRM to guide reasoning steps and using MCTS to efficiently search the collaborative reasoning space, this method generates reasoning paths that are logically consistent and of high quality. This significantly improves the effectiveness of edge-cloud collaborative reasoning.

5 Experiments

5.1 Datasets

We conduct experiments focusing on mathematical reasoning tasks using two widely recognized bench-

Algorithm 1: G-Boost Framework

Input: Input query q, fine-tuned private SLM π_s^+, base SLM π_s^-, general LLM π_l, process reward model PRM, step length L, exploration constant C, maximum iterations T, collaboration probability p_{collab}

```
# Initialize the search tree with the root node
Initialize root node s_0 with V_{s_0} = 0, N_{s_0} = 0
# Main MCTS loop
for t = 1 to T do
    # Selection
    s ← s_0
    while s is not a leaf node do
        s ← argmax_{s' ∈ children(s)} ( V_{s'} + C sqrt( ln N_s / N_{s'} ) )
    end
    τ ← path from root to s
    S_candidate ← {s' | s' ∈ τ and s' has unexpanded actions}
    s_k ← argmax_{s' ∈ S_candidate} V_{s'}
    # Expansion
    if rand() < p_collab then
        # SLM-LLM collaborative inference
        Generate step s_{k+1} using π̃(s_{k+1} | q, s_{1:k}) as in Eq. (5)
    else
        # Private SLM inference
        Generate step s_{k+1} using π_s^+(· | q, s_{1:k})
    end
    Add s_{k+1} as a child of s_k with V_{s_{k+1}} = 0, N_{s_{k+1}} = 0
    # Evaluation
    r ← PRM(q, s_{1:k+1})
    # Backpropagation
    s ← s_{k+1}
    while s ≠ s_0 do
        N_s ← N_s + 1
        V_s ← ((N_s − 1) V_s + r) / N_s
        s_k ← parent(s)
    end
end
```

Output: Optimal terminate reasoning path τ^* with highest value

marks: GSM8K and MATH-500. Both datasets are in English. GSM8K contains over 1,000 grade-school-level word problems designed to test basic arithmetic and problem-solving skills, while MATH-500 is a subset of the MATH dataset, comprising 500 high-school-level problems spanning various mathematical domains. In addition, for fine-tuning SLMs on domain-specific tasks, we use MetaMathQA (Yu et al., 2024) as the private training dataset. It is a high-quality mathematical reasoning dataset in English, constructed by augmenting the training sets of GSM8K and MATH. MetaMathQA contains 395,000 examples, providing a substantial amount of data for fine-tuning and improving the performance of SLMs on mathematical reasoning tasks.

5.2 Experimental Setup

In our experiments, we use two model pairs: Qwen2.5-1.5B with Qwen2.5-14B (Yang et al., 2024); TinyLlama-1B (Zhang et al., 2024) with LLaMA2-13B (Touvron et al., 2023). The smaller models, TinyLlama-1B and Qwen2.5-1.5B, serve as the base SLMs, while the larger models, LLaMA2-13B and Qwen2.5-14B, act as the general LLMs. To minimize potential data leakage and ensure the integrity of the evaluation, all models are based on their pre-trained versions. The base SLMs are fine-tuned on the MetaMathQA dataset to specialize in mathematical reasoning tasks, while the general LLMs are used in their pre-trained form. Besides, we adopt the open-source PRM Math-Shepherd (Wang et al., 2024) in the proposed framework, which is specifically designed to provide fine-grained feedback for mathematical reasoning processes.

To evaluate the effectiveness of the proposed G-Boost framework, we compare the performance of the pre-trained SLM, the fine-tuned SLM, and the fine-tuned SLM guided by PRM to perform MCTS. Moreover, we evaluate the collaborative inference method Proxy-Tuning, whose decoding approach is incorporated into G-Boost. However, Proxy-Tuning performs collaborative decoding throughout the entire inference process without dynamic adjustment capabilities. In contrast, G-Boost, guided by PRM, dynamically explores the action space of collaborative reasoning and adaptively optimizes the reasoning trajectory. All experiments are conducted on four NVIDIA GeForce RTX 4090 GPUs, each with 24 GB of memory. We set the sampling temperature to 0.9 and run each experiment three times with different random seeds, taking the average of the results.

5.3 Main Results

As shown in Tab. 1, the G-Boost framework consistently outperforms both the fine-tuned SLM and the general LLM across all tasks, demonstrating the effectiveness of leveraging general LLMs to enhance private SLMs. Compared to Tuned-MCTS, which relies solely on the fine-tuned SLM guided by PRM without collaborative inference, G-Boost achieves significant accuracy improvements of 8.1% and 5.0% on GSM8K for Qwen2.5 and LLaMA2, respectively, and 8.6% and 2.4% on MATH-500. Similarly, against Proxy-Tuning, which lacks the dynamic adjustment provided by PRM-guided MCTS,

Model	Task	SLM			LLM	SLM+LLM	
		Base	Tuned	Tuned-MCTS	Base	Proxy-Tuning	G-Boost
Qwen2.5	GSM8K	8.1	73.5	76.3	62.2	81.3	**84.4**
	MATH-500	27.2	33.6	35.8	31.8	36.8	**44.4**
LLaMA2	GSM8K	1.2	48.2	59.9	6.5	54.2	**64.9**
	MATH-500	1.4	12.6	16.8	2.4	14.8	**19.2**

Table 1: **Performance comparison on GSM8K and MATH-500.** SLM+LLM denotes collaborative inference methods, including Proxy-Tuning (static) and G-Boost (dynamic). Tuned-MCTS refers to private SLM guided by MCTS without collaboration. Bold values indicate the best performance per task.

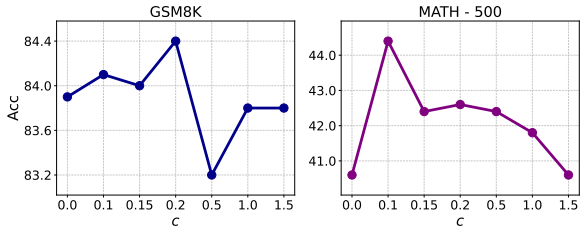

Figure 3: The impact of exploration constant in UCT.

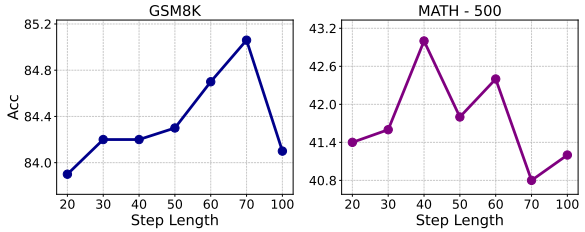

Figure 4: The impact of step length.

G-Boost improves accuracy by 3.1% and 10.7% on GSM8K, and 7.6% and 4.4% on MATH-500 for the two models. These results highlight the advantage of adaptively integrating the general LLM's reasoning capabilities with the private SLM's domain expertise, enabling precise and reliable exploration of reasoning paths.

5.4 Further Analysis

We perform additional analyses to gain deeper insights into the behavior of the proposed G-Boost framework and to further explore its underlying mechanisms. All experiments in this section are conducted on the Qwen2.5 models.

Exploration constant in UCT. The exploration constant c in the UCT formula balances exploration and exploitation during MCTS. We evaluate its impact on G-Boost across MATH-500 and GSM8K. As illustrated in Fig. 3, performance peaks at intermediate c values, where the framework effectively balances exploration of alternative reasoning paths and exploitation of high-quality ones. Smaller c values overly prioritize exploitation, potentially missing optimal paths, while larger values encourage excessive exploration, leading to inefficiency. The results suggest the importance of tuning exploration constant to achieve a robust trade-off, with the framework showing stability across a reasonable range of values.

Step length. The step length parameter determines the granularity of reasoning steps in the G-Boost framework, influencing both the efficiency and quality of collaborative inference. As shown in Fig. 4, shorter step lengths enable more detailed reasoning but may increase the complexity of the search space, while longer step lengths simplify the search process but risk overlooking nuanced reasoning paths. The framework exhibits relatively stable performance across a range of step lengths, suggesting its adaptability to different granularities. Moderate step lengths tend to yield better results, as they strike a balance between capturing sufficient reasoning detail and maintaining manageable computational complexity.

Maximum iteration number. The maximum iteration number T controls the depth of exploration in the MCTS process, influencing the quality of reasoning paths generated by the G-Boost framework. As demonstrated in Tab. 2, increasing T initially improves performance, as more iterations enable broader exploration of the reasoning space and help identify higher-quality paths. However, beyond a certain point, further increases in T provide only marginal gains, indicating that the framework tends to converge toward stable reasoning paths within a reasonable number of iterations. This behavior is consistent with the nature of MCTS, where early iterations play a more significant role in shaping the search tree, while additional iterations contribute less to further improvements. The framework ex-

hibits robustness across a range of T values, with performance remaining relatively stable even as T varies. This suggests that the G-Boost framework can achieve effective reasoning without requiring excessively large iteration counts.

Table 2: The impact of the maximum iterations

T	16	24	32	40	64
GSM8K	82.9	84.5	84.6	85.0	**85.7**
MATH-500	39.0	39.6	**42.4**	**42.4**	42.2

Collaboration probability The collaboration probability p_{collab} governs the decision-making process for invoking the general LLM during node expansion in the G-Boost framework. The results in Tab. 3 reveals that, the choice of p_{collab} significantly impacts the interplay between the private SLM's domain expertise and the general LLM's broader reasoning capabilities. At lower values of p_{collab}, the framework tends to rely more heavily on the private SLM, which may limit its ability to handle queries requiring general reasoning or complex logic. On the other hand, higher values of p_{collab} increase the involvement of the general LLM, but this can sometimes lead to suboptimal outcomes due to its lack of domain-specific knowledge. The framework achieves its best performance at intermediate values of p_{collab}, where it strikes an effective balance between specialization and generalization. This balance allows the framework to dynamically adapt to the strengths of each model, ensuring that domain-specific tasks benefit from both the private SLM's precision and the general LLM's versatility.

Table 3: The impact of collaboration probability

p_{collab}	0.1	0.3	0.5	0.7	0.9
GSM8K	83.7	84.4	84.1	**84.3**	84.2
MATH-500	41.8	42.2	**42.4**	41.8	41.2

Expand strategy The expand strategy in the G-Boost framework determines how nodes are extended during the tree search process, influencing both the efficiency and quality of reasoning. We compare two approaches: Single expansion (expanding one child node at a time) and Full expansion (expanding all child nodes simultaneously). According to Tab. 4, Single expansion consistently outperforms Full expansion. While Full expansion might theoretically accelerate convergence by exploring multiple branches concurrently, it risks allocating limited search budget to less promising or invalid branches, which can dilute the focus on high-quality reasoning paths. In contrast, Single expansion allows the framework to concentrate resources on the most promising steps, guided by the process reward model, leading to more efficient and reliable exploration. This suggests that, under constrained search budgets, a more selective expansion strategy better aligns with the framework's goal of balancing domain-specific expertise and general reasoning capabilities.

Table 4: The effect of different expand strategies.

Expand	Single	Full
GSM8K	**84.4**	84.3
MATH-500	**44.4**	42.0

6 Conclusions

This paper presents G-Boost, a novel framework for enhancing the performance of private SLM by adaptively collaborating with general LLM under the guidance of process reward. G-Boost addresses the limitations of private SLMs, such as weaker general reasoning capabilities, by dynamically integrating the domain-specific expertise of private SLMs with the broad language understanding of general LLMs. Through a tree-structured search process guided by PRM, G-Boost dynamically explores the collaborative reasoning space, balancing specialization and generalization to achieve superior performance on domain-specific tasks. Extensive experiments on mathematical reasoning benchmarks, demonstrate that G-Boost significantly outperforms both fine-tuned private SLMs and general LLMs, as well as static collaborative inference methods like Proxy-Tuning. This work highlights the potential of reward-guided adaptive collaboration for boosting private LLMs and opens new avenues for exploring efficient and flexible edge-cloud reasoning frameworks in specialized tasks.

Limitations

While the G-Boost framework demonstrates significant improvements in enhancing private SLMs, it has certain limitations. For instance, the efficiency of the collaborative inference process could be improved. Future work could focus on optimizing

computational workflows to reduce latency. Additionally, while the framework has shown promising results, its effectiveness across a broader range of domains remains to be fully explored. Future research could prioritize expanding evaluations to diverse domains to validate and potentially extend the applicability of the framework.

Ethical Considerations

This framework aims to improve the performance of private SLMs for users facing resource or privacy constraints, promoting more accessible and privacy-aware language model applications. All datasets utilized in this work are under the MIT License. Our approach is based on open-source language models and also involves interaction with general language models, which may include those accessed via API. Our work did not explicitly handle any bias that exists in the aforementioned pretrained models or datasets.

References

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. 2024. Automix: Automatically mixing language models. In *NeurIPS*.

Ziwei Chai, Guoyin Wang, Jing Su, Tianjie Zhang, Xuanwen Huang, Xuwu Wang, Jingjing Xu, Jianbo Yuan, Hongxia Yang, Fei Wu, and Yang Yang. 2024. An expert is worth one token: Synergizing multiple expert llms as generalist via expert token routing. In *ACL*.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023a. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023b. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James T. Kwok, and Yu Zhang. 2024. Routerdc: Query-based router by dual contrastive learning for assembling large language models. In *NeurIPS*.

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. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Rémi Coulom. 2006. Efficient selectivity and backup operators in monte-carlo tree search. In *Computers and Games, 5th International Conference, CG 2006, Turin, Italy, May 29-31, 2006. Revised Papers*, volume 4630 of *Lecture Notes in Computer Science*, pages 72–83.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. 2024. Hybrid LLM: cost-efficient and quality-aware query routing. In *ICLR*.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *ACL*.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *ECML*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *ICML*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *ICLR*.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. 2024. Tuning language models by proxy. *arXiv preprint arXiv:2401.08565*.

Alireza Mohammadshahi, Arshad Rafiq Shaikh, and Majid Yazdani. 2024. Routoo: Learning to route to

large language models effectively. *arXiv preprint arXiv:2401.13979*.

Adarsh MS, Jithin VG, and Ditto PS. 2024. Efficient hybrid inference for llms: Reward-based token modelling with selective cloud assistance. *arXiv preprint arXiv:2409.13757*.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. 2024. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Aitor Ormazabal, Mikel Artetxe, and Eneko Agirre. 2023. Comblm: Adapting black-box language models through small fine-tuned models. In *EMNLP*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *ACL*.

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

Yao Yao, Zuchao Li, and Hai Zhao. 2024. GKT: A novel guidance-based knowledge transfer framework for efficient cloud-edge collaboration LLM deployment. In *ACL*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *ICLR*.

Huiyou Zhan, Xuan Zhang, Haisheng Tan, Han Tian, Dongping Yong, Junyang Zhang, and Xiang-Yang Li. 2025. PICE: A semantic-driven progressive inference system for LLM serving in cloud-edge networks. *arXiv preprint arXiv:2501.09367*.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

Wenhao Zheng, Yixiao Chen, Weitong Zhang, Souvik Kundu, Yun Li, Zhengzhong Liu, Eric P Xing, Hongyi Wang, and Huaxiu Yao. 2025. Citer: Collaborative inference for efficient large language model decoding with token-level routing. *arXiv preprint arXiv:2502.01976*.