CITER: Collaborative Inference for Efficient Large Language Model Decoding with Token-Level Routing

Wenhao Zheng[1*], **Yixiao Chen**[1*], **Weitong Zhang**[1], **Souvik Kundu**[2], **Yun Li**[1], **Zhengzhong Liu**[4], **Eric P. Xing**[3,4], **Hongyi Wang**[5], **Huaxiu Yao**[1†]
[1]The University of North Carolina at Chapel Hill, [2]Intel, [3]Carnegie Mellon University, [4]Mohamed bin Zayed University of Artificial Intelligence, [5]Rutgers University
wenhao@cs.unc.edu, huaxiu@cs.unc.edu

Abstract

Large language models have achieved remarkable success in various tasks but suffer from high computational costs during inference, limiting their deployment in resource-constrained applications. To address this issue, we propose a novel **C**ollaborative **I**nference with **T**oken-l**E**vel **R**outing (CITER) framework that enables efficient collaboration between small and large language models (SLMs & LLMs) through a token-level routing strategy. Specifically, CITER routes non-critical tokens to an SLM for efficiency and routes critical tokens to an LLM for generalization quality. We formulate router training as a policy optimization, where the router receives rewards based on both the quality of predictions and the inference costs of generation. This allows the router to learn to predict token-level routing scores and make routing decisions based on both the current token and the future impact of its decisions. To further accelerate the reward evaluation process, we introduce a shortcut which significantly reduces the costs of the reward estimation and improving the practicality of our approach. Extensive experiments on five benchmark datasets demonstrate that CITER reduces the inference costs while preserving high-quality generation, offering a promising solution for real-time and resource-constrained applications.

1 Introduction

Large language models (LLMs) have revolutionized various natural language processing tasks, from machine translation to context summarization and question answering (Coleman et al., 2024; Kamalloo et al., 2024; Eniser et al., 2024). However, their impressive performance comes with a substantial computational costs, particularly during inference. As these models grow in size, the costs of inference becomes a significant barrier to their practical deployment, especially in real-time applications. Therefore, there is an increasing need to reduce inference costs without compromising the quality of the generated outputs.

To address these issues, most existing approaches (Dao et al., 2022; Sanh et al., 2020; Kou et al., 2024; Anagnostidis et al., 2024) have proposed different methods to route different input queries to models of different sizes to reduce inference costs while maintaining output quality. Intuitively, small language models (SLMs) are assigned with simpler tasks demanding lower computational resources, while more complex cases are routed to LLMs to ensure response accuracy. However, most existing works only route queries to different models once, which means that either the LLM or the SLM will handle the entire response after routing. This one-step approach limits routing flexibility, as in many response, there is only few critical tokens need to be generated by LLM while the rest of tokens can be easily generated by SLM efficiently. As a result, simply routing these queries to LLM will significantly reduce the efficiency.

[*]Equal contribution.
[†]Corresponding author.

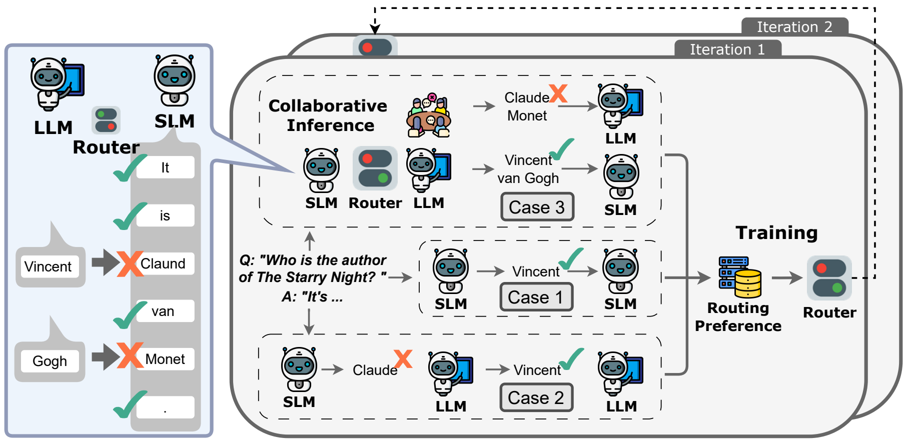


Figure 1: An overview of CITER. A router is leveraged to perform collaborative inference between the SLM and LLM. The router is trained using routing preference collected through three cases. **Case 1**: The SLM generates the correct token, the routing preference is assigned to the SLM. **Case 2**: The SLM generates an incorrect token, while the LLM generates the correct token, the routing preference is assigned to the LLM. **Case 3**: None of the SLM or the LLM generates the correct token, then the collaborative inference is conducted to obtain the completed response for assigning the routing preference.

To address this challenge, we present a novel framework, namely **C**ollaborative **I**nference with **T**oken-l**E**vel **R**outing (**CITER**). CITER introduces a token-level router which determines either LLM or SLM is used to generate each token. Specifically, many tokens in the response that are not important to the final prediction, can be routed and generated by SLM to reduce inference costs, while the LLM can be reserved to generate important tokens only. We propose optimizing this router using pairwise data by reinforcement learning, with the objective of minimizing the inference costs while preserving output quality. By employing this formulation, the router learns to predict token-level routing scores and make routing decisions not only based on the current token but also considering the impact of these decisions on future tokens. In order to further accelerate the estimation of the reward function defined by the accuracy of the response, we present a surrogate reward function as a shortcut, where the predictions from the SLM and LLM are leveraged to estimate the final reward without completing the whole generation process, accelerating the training process significantly. Through this framework, we enable the collaboration of SLM and LLM for effective and efficient autoregressive generation.

Our primary contribution is CITER, which reduces inference costs by employing a token-level router to select the appropriate model to generate each token. Experiments on five benchmark datasets demonstrate the effectiveness of our approach, making up to 30% fewer inference costs with comparable accuracy or delivering up to a 25% improvement in accuracy with the same cost compared to Co-LLM (Shen et al., 2024). Furthermore, our experiments in the ablation study also demonstrate that token-level routing offers more flexibility to achieve more promising results compared to query-level routing and that considering the long-term impact of routing decisions significantly boosts performance.

Notations. We denote $\pi_{\boldsymbol{\theta}}$ as the policy model parameterized by $\boldsymbol{\theta}$, x_i as the i-th token in the input prompt, y_j as the j-th token of the output response, $\mathbb{1}[\cdot]$ as the indicator function and $\oplus$ as the concatenate operation. All other notations are defined prior to their first usage.

2 Collaborative Inference with Token-lEvel Routing (CITER)

In this section, we describe our **C**ollaborative **I**nference with **T**oken-l**E**vel **R**outing (CITER) framework that uses token-level routing to accelerate the inference of language models. As illustrated in Figure 1, we introduce a router to facilitate collaborative inference between a powerful but computationally expensive LLM and a fast but potentially inaccurate SLM. Specifically, the router is used to predict the token-level routing score for each token, and a predefined threshold τ is used to determine which model should generate this token. The key challenge of our framework is the router training process. To feed the router with the knowledge on making the global optimal routing decisions not only based on the accuracy of the current token but also the long-term impact of its decision, we formulate the router training process as a preference-based policy optimization task, aiming to minimize the inference costs while maintaining the generation quality. To be more specific, we first formulate the RL problem and derive the reward function as token-wise routing preference, which should be computed to collect during the router training process. Subsequently, we introduce a shortcut for the reward function estimation, leveraging both the SLM and LLM's prediction to estimate the reward, significantly accelerating the collection process of the token-wise routing preference. Finally, we propose an iterative router training process to mitigate the potential inconsistencies of routing decisions in the preference collection phase and deployment. In the rest of this section, we will outline router training and collaborative inference processes in detail.

2.1 Reinforcement Learning for Router Optimization

We start by introducing the foundational concepts and notation for the Markov Decision Process for token-level routing. In particular, we formulate the token-level routing task as a Markov decision process (MDP) (Bellman, 1957) where state is a series of tokens $\mathbf{s}_h = (x_0, \cdots x_m, y_0, \cdots, y_h)$, including both the input prompt $(x_0, \cdots x_m)$ and the response $(y_0, \cdots, y_h)$. At each time step h, the agent selects its action from $\mathcal{A} = \{A_L, A_S\}$, which means generating a token from LLM (A_L) or SLM (A_S), respectively. Then we write the generation of the next token by the following transition kernel $\mathbb{P}(\mathbf{s}_{h+1}|\mathbf{s}_h, a_h)$ given by the dynamics of LLM and SLM. This generation process ends once the terminal token `<EOS>` is generated from either of these models. The generated token length is denoted as H, which can be flexible. The reward $r(\mathbf{s}_h, a_h)$ is then related to the generation cost and the accuracy of the final response $r(\mathbf{s}_H)$. The state-action value function is defined by

$$Q_h^{\pi}(\mathbf{s}, a) = \mathbb{E}\left[\textstyle\sum_{t=h}^{H} r(\mathbf{s}_t, a_t)\middle|\mathbf{s}_h = \mathbf{s}, a_h = a, \pi\right], \tag{1}$$

with the optimal state-action value function Q^* defined as $Q_h^*(\mathbf{s}, a) = \max_\pi Q_h^\pi(\mathbf{s}, a)$. The objective of the routing policy can be written by

$$\pi_h^*(a|\mathbf{s}) = \arg\max_{\pi} \mathbb{E}\left[Q_h^*(\mathbf{s}, a) - \beta \mathrm{KL}(\pi \parallel \mu)\right] \propto \mu(a|\mathbf{s}) \exp(\beta^{-1} Q_h^*(a|\mathbf{s})), \tag{2}$$

where μ is the prior policy related to the cost difference for evaluating LLM or SLM. The expectation is taken over the randomness of language models, policy π and the prompt $\mathbf{s}_0$.

2.2 Preference-based Token-level Policy Optimization

Generally, defining the reward $r(\mathbf{s}_h, a_h)$ as well as the state-action value function $Q_h(\mathbf{s}_h, a_h)$ is difficult and may result in reward hacking (Amodei et al., 2016; Leike et al., 2020). To tackle with this issue, similar with (Rafailov et al., 2023), we inject the pairwise preference $\mathbb{1}_h[a_S \succ a_L]$ following the Bradley–Terry model (Bradley & Terry, 1952) as:

$$\Pr_h(a_S \succ a_L|\mathbf{s}_h) = \sigma(\beta(Q_h^*(\mathbf{s}_h, a_S) - Q_h^*(\mathbf{s}_h, a_L))), \tag{3}$$

where $\sigma(z)$ is the sigmoid function. Following Rafailov et al. (2024), we have

$$\begin{aligned} Q_h^*(\mathbf{s}_h, a_L) - Q_h^*(\mathbf{s}_h, a_S) &= \beta \log \frac{\pi_h^*(a_L|\mathbf{s}_h)}{\mu(a_L)} - \beta \log \frac{\pi_h^*(a_S|\mathbf{s}_h)}{\mu(a_S)} \\ &= \beta \log \frac{\pi_h^*(a_L|\mathbf{s}_h)}{\pi_h^*(a_S|\mathbf{s}_h)} - \beta \log \frac{\mu(a_L)}{\mu(a_S)}, \end{aligned} \tag{4}$$

In the case of $\mu(a_L) = \mu(a_S)$ and $\beta = 1$, plugging equation 4 into equation 3 yields

$$\Pr(a_S \succ a_L|\mathbf{s}_h) = \frac{1}{1 + \frac{\pi_h^*(a_L|\mathbf{s}_h)}{\pi_h^*(a_S|\mathbf{s}_h)}} = \pi_h^*(a_S|\mathbf{s}_h),$$

Algorithm 1 Preference-based Router Optimization for CITER

```
1: Input: Training data $\mathcal{D} = \{\mathbf{x}, \mathbf{y}^*\}$, SLM and LLM $M_S, M_L$, number of rounds $K$.
2: Initialize training policy $\pi_{\boldsymbol{\theta}}$, preference set $\mathcal{P}_0 = \emptyset$
3: for round $k = 1, \cdots, K$ do
4:     Initialize preference set $\mathcal{P}_k = \emptyset$
5:     for prompt-response pair $\mathbf{x}, \mathbf{y}^*$ in $\mathcal{D}$ do
6:         Set $h = 0$, $\mathbf{s}_0 = \mathbf{x}$
7:         while $y_h$ is not <EOS> do
8:             if $M_S(\mathbf{s}_h) = y^*_{h+1}$ then Set $p_h = 1$ /* Case 1. $a_S \succ a_L$ */
9:             if $M_L(\mathbf{s}_h) = y^*_{h+1}$ then Set $p_h = 0$ /* Case 2. $a_L \succ a_S$ */
10:            else /* Case 3. */
11:            Generate new token: $\tilde{\mathbf{s}}_{h+1} = \mathbf{s}_h \oplus [M_S(\mathbf{s}_h)]$
12:            Generate $\tilde{\mathbf{s}} = \textbf{CITER}(\tilde{\mathbf{s}}_{h+1}, M_S, M_L, \pi_{\boldsymbol{\theta}}, \frac{1}{2})$
13:            $p_h = 1$ if $\tilde{\mathbf{s}}$ is correct else $p_h = 0$.
14:            Update $\mathcal{P}_k = \mathcal{P}_k \cup \{\mathbf{s}_h, p_h\}$
15:    /* Preference-based Optimization */
16:    Update $\boldsymbol{\theta}$ by minimizing loss $\mathcal{L}(\boldsymbol{\theta}) = -\sum_{(\mathbf{s},p)\in\mathcal{P}_k} p \log \pi_{\boldsymbol{\theta}}(a_S|\mathbf{s}) + (1-p)\log \pi_{\boldsymbol{\theta}}(a_L|\mathbf{s})$
17: Output: Routing policy $\pi_{\boldsymbol{\theta}}$.
```

where the latter equation is due to the fact that $\pi^*_h(a_S|\mathbf{s}_h) + \pi^*_h(a_L|\mathbf{s}_h) = 1$. Therefore, given a sequence of token $\mathbf{s}_h$, once we have labeled the preference $\mathbb{1}[a_S \succ a_L|\mathbf{s}_h]$, $\pi_h(a_S|\mathbf{s}_h)$, the routing agent π can be learned by minimizing the cross-entropy loss

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{\mathbf{s}_h}(\mathbb{1}_h[a_S \succ a_L|\mathbf{s}_h] \log \pi_h(a_S|\mathbf{s}_h, \boldsymbol{\theta}) + \mathbb{1}_h[a_L \succ a_S|\mathbf{s}_h] \log \pi_h(a_L|\mathbf{s}_h, \boldsymbol{\theta})), \quad (5)$$

where $\mathbb{1}[a_L \succ a_S|\mathbf{s}_h]$ indicates using large language model is preferred at state $\mathbf{s}_h$.

2.3 Acquiring Token-level Routing Preference

In this subsection, we describe our strategy to determine the preference label $\mathbb{1}[a_L \succ a_S|\mathbf{s}_h]$. For a state $\mathbf{s}_h$, we first generate the next token y_{h+1} with the small language model and then complete the whole trajectory $\mathbf{s}_H$ until the generation ends with <EOS> using the routing policy π. Compared to equation 1, the reward collected on this trajectory $\mathbf{s}_H$ is an unbiased estimation of $Q^\pi_h(\mathbf{s}_h, a_S)$. Intuitively, if using the small language model in the current step h can generate the correct response $\mathbf{s}_H$, then the small language model is preferred. Otherwise, we assign $a_L \succ a_S$ and assume that the large language model can generate the correct answer, as implemented in Line 9 in Algorithm 1.

However, generating and evaluating the final response $\mathbf{s}_H$ might be expensive or even infeasible when H is large. In order to further accelerate the token-level routing preference label, we introduce a shortcut by leveraging the ground truth response $\mathbf{s}^*_H$ provided in the dataset. As Line 8 in Algorithm 1 suggests, if the next token y^S_{h+1} generated by the small language model is exactly the same as the ground-truth token y^*_{h+1}, we assign $a_S \succ a_L$ since the behavior of the small language model is good enough to match the ground-truth model. In the case where the next token generated by the small language model does not match the ground truth, as carried out in Line 10, we check the next token generated by the large language model y^L_{h+1} and assign $a_L \succ a_S$ if $y^L_{h+1} = y^*_{h+1}$. Otherwise we will go back to the aforementioned case to evaluate the completed generated trajectory as conducted in Line 10 in Algorithm 1. We would like to highlight that only when both models fail to generate the correct token y^*_h based on ground truth context, the full response generation is required to compute the reward. This shortcut allows us to obtain routing preferences for most tokens without generating the full response. Empirically, we find that about 80% $\sim$ 90% of the tokens can be correctly predicted by either the SLM or LLM, which makes the shortcut significantly reduce the inference costs of estimating the reward function.

2.4 Proposed Algorithm

Finally we summarize the proposed algorithm as well as some implementation details in Algorithm 1 as an iterative update of the routing policy $\pi_{\boldsymbol{\theta}}$. In each iteration k, the router

Algorithm 2 **C**ollaborative **I**nference with **T**oken-l**E**vel **R**outing (**CITER**)

```
1: Input: Input prompt $\mathbf{s}$, SLM and LLM $M_S, M_L$, policy $\pi_{\boldsymbol{\theta}}$, threshold $\tau$
2: Let $\tilde{\mathbf{s}} = \mathbf{s}$
3: while True do
4:     Set $M = M_S$ if $\pi(a_S|\tilde{\mathbf{s}}) \geq \tau$ else $M = M_L$
5:     Generate next token and let $\tilde{\mathbf{s}} = \mathbf{s} \oplus \{M(\tilde{\mathbf{s}})\}$
6:     if $M(\tilde{\mathbf{s}})$ = <EOS> then break
7: Output: Generated response $\tilde{\mathbf{s}}$.
```

Table 1: The statistics of our evaluation datasets. The commonsense QA dataset and MMLU-Professional Psychology dataset are denoted as CS QA and MMLU-PP, respectively.

Dataset	Domain	Task	# choices	Train size	Test size
Commonsense QA (CS QA)	General	Multi-choice	5	9,741	1,221
ARC-Challenge	Reasoning	Multi-choice	4	1,119	299
MMLU-PP	Psychology	Multi-choice	4	612	69
GSM8k	Math	Question answering	N/A	7,473	1,319
MATH	Math	Question answering	N/A	7,500	5,000

$\pi_{\boldsymbol{\theta}_{k-1}}$ from the previous iteration is used to collect routing preferences $\mathcal{P} = \{\mathbf{s}, p\}$. Then iteration goes for at most K rounds but can also stop early when $\mathcal{P}_k = \mathcal{P}_{k-1}$ and thus the policy optimization converges. The preference $p \in \{0, 1\}$ is labeled through the three cases described in Subsection 2.3, where the algorithm calls the inference algorithm **CITER** when neither the LLM nor the SLM can predict the correct token.

The inference algorithm **CITER** is presented in Algorithm 2. In detail, **CITER** uses a deterministic policy where it chooses the small language model when $\pi(a_S|\mathbf{s}) \geq \pi(a_L|\mathbf{s})$ (i.e., $\pi(a_S|\mathbf{s}) \geq \frac{1}{2}$) and vice versa. To further investigate the balance between efficiency and precision by collaborating with LLM and SLM, we introduce another layer of prior policy $(\rho(a_S), \rho(a_L))$, where $\rho(a_S) + \rho(a_L) = 1$. Thus, the deterministic rule of selecting the SLM from the posterior policy distribution $\pi'(a|\mathbf{s}) \propto \pi(a|\mathbf{s})\rho(a)$ is that

$$\pi(a_S|\mathbf{s})\rho(a_S) \geq \pi(a_L|\mathbf{s})\rho(a_L) \Rightarrow \pi(a_S|\mathbf{s}) \geq \rho(a_L),$$

where we introduce $\tau := \rho(a_L)$ as a hyper parameter in the algorithm to probe this balance.

Remark 2.1. We maintain separate KV caches for SLM and LLM. When CITER switches between them, the previous KV cache is preserved, allowing it to be reused when switching back. This eliminates the need for redundant computations, improving efficiency.

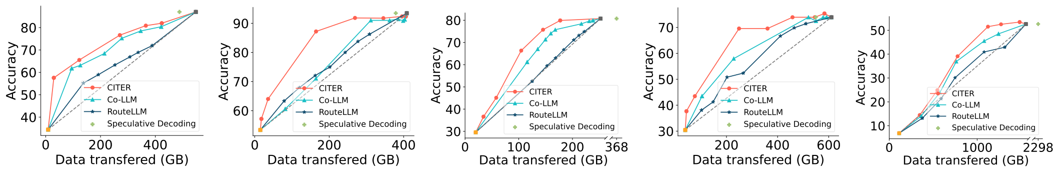
3 Experiments

In this section, we evaluate the performance of CITER aiming to answer the following questions: (1) Compared with the previous works on speeding up the inference of LLM, how does our framework perform in terms of the inference costs and the response quality? (2) Does the components we proposed in our framework boost the performance of the router? (3) Does the iterative training process of the router improve the performance of our framework? (4) How does the performance of our framework change with the size of the LLM? (5) Can the router distinguish the critical and non-critical tokens correctly?

3.1 Experimental Setup

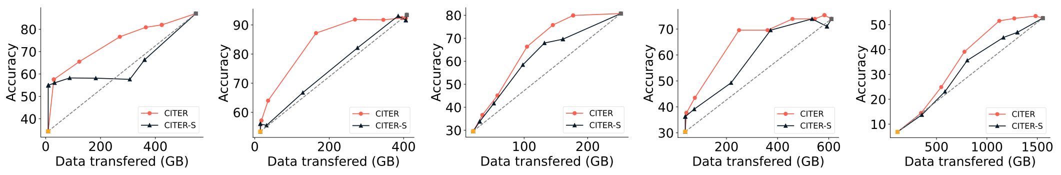
Dataset Description. We evaluate CITER and our baselines on five widely-used academic benchmark datasets: the commonsense QA dataset (Talmor et al., 2019) contains 12,102 questions requiring different types of commonsense knowledge to answer; the ARC-Challenge dataset (Clark et al., 2018), including 1,418 genuine grade-school level, multiple-choice science questions; the MMLU-Professional Psychology dataset (Hendrycks et al., 2021a), consisting of 874 multiple-choice questions on psychology; the GSM8k dataset (Cobbe et al., 2021) with 8.5K high quality linguistically diverse grade school math word problems and MATH dataset (Hendrycks et al., 2021b) with 12.5k problems from mathematics competitions respectively. The statistics of the datasets are in Table 1.

Evaluation. To evaluate the performance of CITER and the baseline methods, we use the accuracy of responses to reflect response quality and define the inference cost as the average



(a) CS QA (b) ARC-Challenge (c) GSM8k (d) MMLU-PP (e) MATH

Figure 2: The accuracy vs data transformation amount curve of CITER and the baselines. The yellow and grey squares represent the performance of slm and llm respectively. The grey line represents the random routing strategy. Points closer to the top-left corner indicate better acceleration performance.



(a) CS QA (b) ARC-Challenge (c) GSM8k (d) MMLU-PP (e) MATH

Figure 3: The accuracy vs data transformation amount curve of CITER and the varient CITER-S. The yellow and grey squares represent the performance of slm and llm respectively. The grey line represents the random routing strategy. Points closer to the top-left corner indicate better acceleration performance.

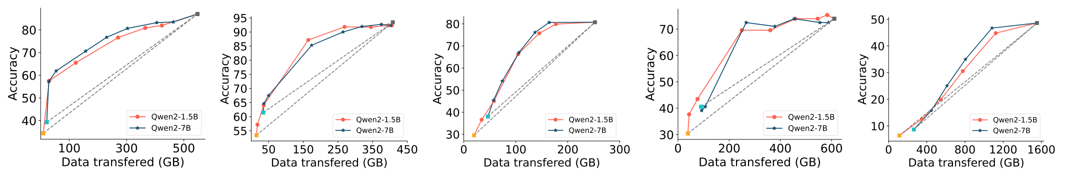
amount of data transformation per sample, mainly the KV cache that must be transferred from GPU HBM to the on-chip cache, since LLM generation is primarily memory-bound. Details on the data transformation calculations and an illustration of the memory-bound nature of LLM generation are provided in Appendix E. Additionally, for both query-level routing methods and token-level routing methods, a threshold τ is applied in each method to balance the trade-off between leveraging the LLM to improve the response quality or offloading to the SLM to reduce the overall inference costs. We then plot the accuracy curve versus the average amount of data transformation per sample to illustrate the acceleration performance of both CITER and the baselines. The optimal point is located in the top-left corner of the curve, corresponding to the highest accuracy with the lowest costs.

Baselines. We compare CITER against three methods: a representative query-level routing approach (RouteLLM (Ong et al., 2024)), a token-level routing method (Co-LLM (Shen et al., 2024)), and a non-routing-based technique (Speculative Decoding (Leviathan et al., 2023)). RouteLLM makes routing decisions at the query level, directing entire queries to different models for generation, while Co-LLM operates at the token level, dynamically routing each token to different models throughout the generation process. In contrast, Speculative Decoding does not involve routing between models; instead, it leverages the SLM to propose a set of candidate tokens, and then verify them by the LLM.

Implementation Details We implement our framework using the Hugging Face Transformers library (Wolf et al., 2020). For the SLM and LLM, we utilize Qwen2-1.5b and Qwen2-72b, respectively. The router is implemented as a multilayer perceptron (MLP) network with three hidden layers, ReLU activation (Agarap, 2019), BatchNorm normalization (Ioffe & Szegedy, 2015), and a 0.1 dropout rate. It is trained using the Adam optimizer (Kingma & Ba, 2017) with a learning rate of 1×10^{-7}, betas of $(0.9, 0.99)$, and no weight decay. Training is performed on a single NVIDIA H100 GPU with a batch size of 80. The iterative training process runs for 2 rounds. We use the hidden state corresponding to the last generated token from the SLM as the input to our router. This approach enables the router to utilize the rich representations extracted by the SLM, allowing routing decisions to be informed not only by the current token but also by the broader context accumulated thus far.

3.2 Overall Performance

We conduct extensive experiments to assess the performance of CITER across all benchmark datasets, comparing it against baseline methods. The results are presented in Figure 2. Clearly, all token-level routing methods, including CITER and Co-LLM, significantly out-



(a) CS QA (b) ARC-Challenge (c) GSM8k (d) MMLU-PP (e) MATH

Figure 4: The accuracy vs data transformation amount curve of CITER with 1.5B SLM and CITER with 7B SLM. The yellow, blue and grey squares represent the performance of Qwen2-1.5B, Qwen2-7B and Qwen2-72B respectively. The grey line represents the random routing strategy. Points closer to the top-left corner indicate better acceleration performance.

perform the query-level routing method, RouteLLM, across all datasets, particularly on the Commonsense QA and GSM8k datasets, reducing up to 30% inference costs while maintaining the same accuracy or achieving up to 12% higher accuracy with the same cost. This emphasizes the effectiveness of token-level routing, which provides enhanced flexibility in reducing inference costs while preserving response quality. Notably, Speculative Decoding does reduce inference costs on some multiple-choice datasets. However, its verification mechanism requires the small model to produce outputs identical to those of the large model to maintain lossless output quality, which is overly stringent and limits the potential for further cost reduction on complex cases. As a result, on mathematical datasets, the acceptance rate of candidate tokens proposed by the small model is extremely low, leading to higher inference costs than simply using the large model alone, which is unacceptable. Furthermore, CITER consistently surpasses Co-LLM, achieving comparable accuracy with up to 27% fewer inference costs or delivering up to a 17% improvement in accuracy with the same cost. These findings demonstrate the success of our framework in accelerating LLM inference. This outcome is expected, as Co-LLM does not consider long-term information during the router training phase, which is crucial for token-level routing. In the following section, we present experiments to further demonstrate the importance of incorporating long-term information in router training.

3.3 Analysis of Long-Term Influence

In this section, we conduct an ablation study on a key component of our framework: the long-term influence of routing decisions, to evaluate its effectiveness. For this purpose, we design an ablation variant, CITER-S, where the SLM is selected if both the SLM and LLM provide incorrect predictions during the routing preference collection, disregarding the long-term impact of routing decisions. The results are shown in Figure 3. Clearly, CITER significantly outperforms the ablation variant CITER-S across all datasets, reducing inference costs by up to 42% while maintaining the same accuracy, or achieving up to a 23% accuracy improvement with the same cost. These findings highlight the critical role of accounting for the long-term influence of routing decisions.

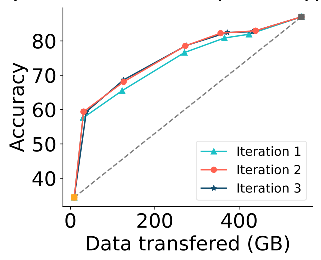
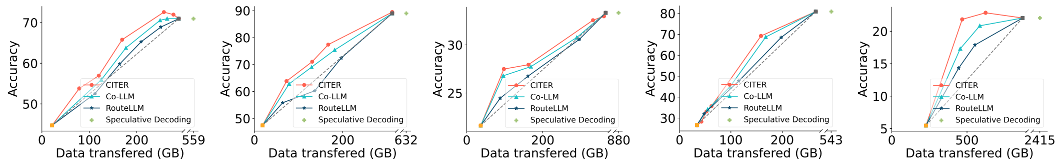


Figure 5: Accuracy vs. inference costs of CITER with router over the first three iterations on the commonsense QA datasets. Points closer to the top-left corner indicate better acceleration performance.

3.4 Analysis of Iterative Training Process

To highlight the importance of the iterative training process, we present the performance curve of CITER with the router over the first three iterations on the Commonsense QA dataset. As shown in Figure 5, the results demonstrate a clear improvement in performance in the first two iterations. In the second iteration, CITER reduces $\sim$ 5% inference costs while maintaining the same accuracy or achieves 2 $\sim$ 3% higher accuracy with the same cost compared to the first. This improvement underscores the effectiveness of our proposed iterative training process. Moreover, the performance curve of the third iteration closely follows that of the second, indicating that the router has already converged by the second iteration. The rapid convergence of the router emphasizes the robustness of our training strategy, suggesting that optimal performance can be achieved without excessive costs.



(a) CS QA (b) ARC-Challenge (c) GSM8k (d) MMLU-PP (e) MATH

Figure 6: The accuracy vs data transformation amount curve of CITER and the baselines with Llama3.1 series. The yellow and grey squares represent the performance of slm and llm respectively. The grey line represents the random routing strategy. Points closer to the top-left corner indicate better acceleration performance.

3.5 Results on Different Model Families

Additionally, we conduct experiments with Llama3.1 series models to demonstrate the compatibility of our framework. Specifically, we leverage the Llama3.1-70B model as the LLM and the Llama3.1-8B model as the SLM. The results are illustrated in Figure 6. Similarly to the results with Qwen series, CITER consistently outperforms all other baseline methods, achieving comparable accuracy with up to 32% fewer inference costs or providing up to a 5% improvement in accuracy with the same cost, compared to Co-LLM, the best baseline method. This result further demonstrates the effectiveness of our framework and additionally highlights the compatibility of our framework with different series of models.

3.6 Analysis of the Impact of SLM Model Size

We further scale up the SLM size from Qwen2-1.5B to Qwen2-7B, while keeping the LLM fixed to Qwen2-72B, to understand the scalability of our framework. As shown in Figure 4, the results clearly demonstrate that CITER reduces inference costs by up to 10% while maintaining the same level of accuracy or achieves up to 11% higher accuracy with the same cost when using Qwen2-7B as the SLM compared to Qwen2-1.5B, particularly on the commonsense QA and GSM8k datasets, underscoring our framework's scalability with larger SLMs. However, the performance gap is most noticeable when only very little tokens are generate by the LLM introducing a very small additional cost, and it gradually diminishes or even disappears as the cost further increases. This is expected, as the SLM's capacity limits its performance, and the quality of responses increasingly depends on the LLM as more calls are routed to it.

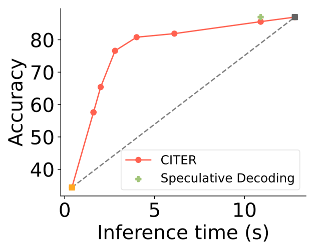


Figure 7: Accuracy vs. latency curve of CITER and Speculative Decoding.

3.7 Latency Analysis

In addition, we also evaluate the wall-clock latency of CITER against Speculative Decoding. The results are presented in Figure 7. The plot shows that CITER offers a flexible trade-off between latency and accuracy. For instance, CITER achieves an accuracy of 80.8% with a latency of only 4 seconds. While Speculative Decoding reaches a slightly higher accuracy of 87.0%, it requires a significantly longer latency of 10.9 seconds. CITER can also achieve 85.6% accuracy which is very similar to Speculative Decoding with the same latency. In addition, key advantage of CITER lies in the flexibility to operate at much lower latency points with only a minor compromise in accuracy, a feature not available with Speculative Decoding. This demonstrates the superior efficiency and adaptability of our framework.

3.8 Qualitative Analysis on the Router

Finally, we perform a case study to further analyze the decision-making process of the router in our framework. A selection of examples, along with their corresponding routing decisions, is shown in Figure 8. In the left example, it is clear that our router accurately identifies the critical tokens, including the first occurrence of the answer "Midwest" and the word "fertile," which describes the farmland in the Midwest, both crucial to the final answer. Moreover, most non-critical tokens are efficiently offloaded to the SLM, effectively reducing inference costs.

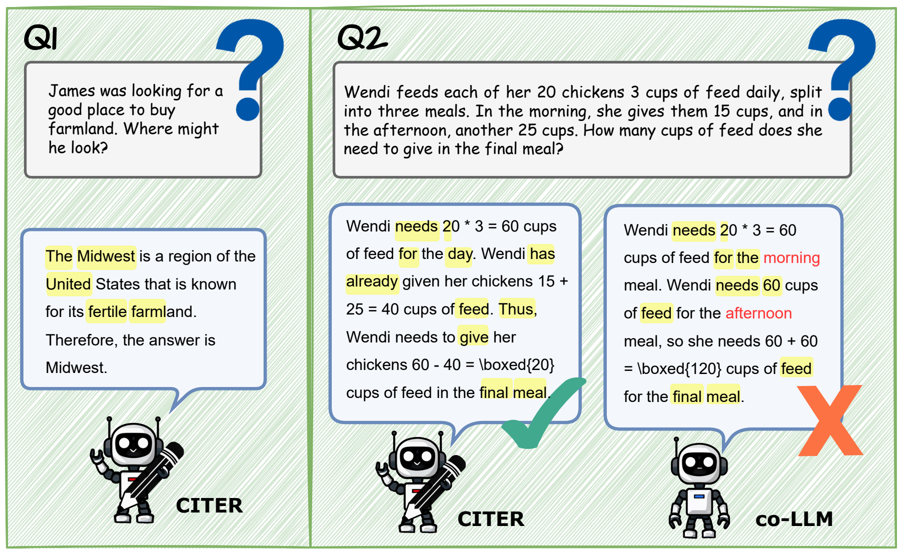

Q1
James was looking for a good place to buy farmland. Where might he look?
The Midwest is a region of the United States that is known for its fertile farmland. Therefore, the answer is Midwest.
CITER
Q2
Wendi feeds each of her 20 chickens 3 cups of feed daily, split into three meals. In the morning, she gives them 15 cups, and in the afternoon, another 25 cups. How many cups of feed does she need to give in the final meal?
Wendi needs 20 * 3 = 60 cups of feed for the day. Wendi has already given her chickens 15 + 25 = 40 cups of feed. Thus, Wendi needs to give her chickens 60 - 40 = \boxed{20} cups of feed in the final meal.
CITER
Wendi needs 20 * 3 = 60 cups of feed for the morning meal. Wendi needs 60 cups of feed for the afternoon meal, so she needs 60 + 60 = \boxed{120} cups of feed for the final meal.
co-LLM

Figure 8: The case study analysis of our router. The words highlighted with yellow background are generated by the LLM, while other words are generated by the SLM. The red-marked words are the mistakes in Co-LLM's response.

In the right example, we compare CITER with the token-level routing method Co-LLM. Clearly, our router outperforms Co-LLM by correctly identifying potential critical tokens, particularly time-related words. In Co-LLM's response, at the first red-marked word "morning," Co-LLM incorrectly routes the word "the" to the LLM while assigning the contextually important word "morning" to the SLM, leading to an initial error in the response. Similarly, Co-LLM routes the critical phrase "afternoon meal" to the SLM, resulting in the final incorrect prediction. In contrast, our router correctly identifies the critical word "day" and routes it to the LLM, followed by routing the phrase "has already" to accurately capture the reasoning process, ultimately leading to the correct prediction.

These examples illustrate that the router in CITER effectively distinguishes between critical and non-critical tokens, offloading non-critical tokens to the SLM to minimize inference costs, while leveraging the LLM to ensure the quality of the generated response.

4 Related Work

In this section, we conduct a literature review that mainly focuses on prior LLM inference acceleration methods, especially those that involve using routing mechanisms and collaborative inference between LLMs for inference acceleration.

Query-Level Routing Mechanisms. Previous routing methods (Jang et al., 2023; Chronopoulou et al., 2023; Diao et al., 2023; Lu et al., 2023; Cheng et al., 2024; Lu et al., 2024; Chen et al., 2023b; Wang et al., 2024b; Srivatsa et al., 2024; Stripelis et al., 2024) for efficient inference mainly focus on routing entire user queries to different models for generation. For example, Routoo (Mohammadshahi et al., 2024) proposes a performance predictor and a cost-aware decoder to route between LLMs, considering both performance and resource constraints; Hybird LLM (Ding et al., 2024) proposes a probabilistic router to select LLM backend for each query; RouteLLM (Ong et al., 2024) formulates the routing problem as a classification problem and employs a data augmentation framework to significantly expand the dataset used for training the router; Gupta et al. (2024) first leverages the small model (SLM) to generate the entire sequence, and then the decision to defer to the large model (LLM) is made based on the uncertainty (e.g., entropy or confidence derived from logits) during generation;FrugalGPT Chen et al. (2023b) formulates the routing problem as a constrained optimization problem, where the final generated quality is maximized under a budget or inference cost constraint. However, as highlighted in Section 1, routing at the query-level granularity may lead to suboptimal performance, as non-critical tokens in complex queries may be generated inefficiently, while critical tokens in simple queries may suffer from inaccuracy. In contrast, token-level routing methods offer more fine-grained control over the routing process, improving both inference costs and the quality of the generated response.

Token-Level Routing Mechanisms. Unlike query-level routing methods, previous token-level routing methods (Pfeiffer et al., 2021; Belofsky, 2023; Muqeeth et al., 2024; Wang et al., 2024a; Wu et al., 2024; Xu et al., 2024) mainly focus on routing input tokens to different specialized experts to enhance performance without considering the inference costs. For example, Arrow (Ostapenko et al., 2024) builds a mixture-of-experts (MoE) architecture with multiple LoRAs, dynamically routing inputs to different LoRAs during inference. Similarly, Branch-Train-MiX (Sukhbaatar et al., 2024) fine-tunes LLMs on different domains from a seed LLM, creating specialized experts to form an MoE framework. Besides these methods, Narasimhan et al. (2024) introduces token-level cascading and tackles the challenge of implementing deferral rules by leveraging speculative decoding, relying exclusively on the logits output by the model to determine when to defer to the LLM. Similarly, Co-LLM (Shen et al., 2024) introduces a router to route tokens to models of different sizes. However, they only consider the current outputs from SLM and LLM when generating ground truth labels to make the router decisions. This may lead to suboptimal performance since the influence of current decisions on future tokens is not considered. Moreover, similar to other token-level routing methods, they focus on enhanced response quality without taking the inference costs of the inference process into account. In contrast, our CITER framework considers both the current token and the future impact of each decision, enabling more accurate and efficient routing.

Other Methods for LLM Inference Acceleration. In addition to routing methods, several approaches ranging from algorithmic to system optimizations (Miao et al., 2023; Kwon et al., 2023; Chen et al., 2024a) have been proposed to accelerate LLM inference. Speculative Decoding (Leviathan et al., 2023; Chen et al., 2023a) employs a small draft model to generate potential next tokens, which are concatenated with previously generated tokens. These guesses are then processed by the target LLM in parallel to verify their correctness. Tokens are only committed to the final output if confirmed by the target LLM. Although this approach reduces inference time by generating multiple tokens in a single forward pass, it does not lower the overall computational complexity (e.g., the total amount of FLOPs). Speculative Streaming (Bhendawade et al., 2024) addresses the computational overhead of Speculative Decoding by predicting n-grams instead of individual tokens in each forward pass. However, it requires redesigning the LLM architecture, necessitating re-pretraining, which is computationally prohibitive for many use cases. Medusa (Cai et al., 2024) mitigates the re-pretraining issue by adding auxiliary heads to the original LLM, allowing n-gram predictions without modifying the core model. These heads can be trained while keeping the original LLM frozen, thereby avoiding the need for re-pretraining. SpecInfer and Sequoia (Miao et al., 2023; Chen et al., 2024b) leverage tree-based parallelism for decoding and verification to further accelerate inference.

5 Conclusion

In this paper, we introduced CITER, a novel collaborative inference with token-level routing framework designed to reduce the inference cost of LLM while maintaining high-quality generation. By dynamically routing non-critical tokens to a SLM and reserving the LLM for critical tokens, CITER achieves an efficient balance between inference cost and generation quality. We formulated the routing problem as a policy optimization task, where the router learns to make token-level decisions based on both immediate token quality and long-term impact. Furthermore, we introduced a shortcut for reward estimation to enhance training efficiency. Experimental results across five benchmark datasets demonstrate that CITER significantly reduces inference costs while preserving accuracy, offering a promising solution for real-time and resource-constrained applications.

References

Abien Fred Agarap. Deep learning using rectified linear units (relu), 2019. URL https://arxiv.org/abs/1803.08375.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. arXiv preprint arXiv:1606.06565, 2016.

Sotiris Anagnostidis, Dario Pavllo, Luca Biggio, Lorenzo Noci, Aurelien Lucchi, and Thomas Hofmann. Dynamic context pruning for efficient and interpretable autoregressive transformers, 2024. URL https://arxiv.org/abs/2305.15805.

Richard Bellman. Dynamic Programming. Princeton University Press, 1957.

Joshua Belofsky. Token-level adaptation of lora adapters for downstream task generalization, 2023. URL https://arxiv.org/abs/2311.10847.

Nikhil Bhendawade, Irina Belousova, Qichen Fu, Henry Mason, Mohammad Rastegari, and Mahyar Najibi. Speculative streaming: Fast llm inference without auxiliary models, 2024. URL https://arxiv.org/abs/2402.11131.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. Biometrika, 39(3/4):324–345, 1952.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. arXiv preprint arXiv: 2401.10774, 2024.

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

Jian Chen, Vashisth Tiwari, Ranajoy Sadhukhan, Zhuoming Chen, Jinyuan Shi, Ian En-Hsu Yen, and Beidi Chen. Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. arXiv preprint arXiv:2408.11049, 2024a.

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

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable, robust, and hardware-aware speculative decoding. arXiv preprint arXiv:2402.12374, 2024b.

Feng Cheng, Ziyang Wang, Yi-Lin Sung, Yan-Bo Lin, Mohit Bansal, and Gedas Bertasius. Dam: Dynamic adapter merging for continual video qa learning, 2024. URL https://arxiv.org/abs/2403.08755.

Alexandra Chronopoulou, Matthew E. Peters, Alexander Fraser, and Jesse Dodge. Adaptersoup: Weight averaging to improve generalization of pretrained language models, 2023. URL https://arxiv.org/abs/2302.07027.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. arXiv:1803.05457v1, 2018.

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.

Jared Coleman, Bhaskar Krishnamachari, Khalil Iskarous, and Ruben Rosales. Llm-assisted rule based machine translation for low/no-resource languages. arXiv preprint arXiv:2405.08997, 2024.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

Shizhe Diao, Tianyang Xu, Ruijia Xu, Jiawei Wang, and Tong Zhang. Mixture-of-domain-adapters: Decoupling and injecting domain knowledge to pre-trained language models memories, 2023. URL https://arxiv.org/abs/2306.05406.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing, 2024. URL https://arxiv.org/abs/2404.14618.

Hasan Ferit Eniser, Hanliang Zhang, Cristina David, Meng Wang, Maria Christakis, Brandon Paulsen, Joey Dodds, and Daniel Kroening. Towards translating real-world code with llms: A study of translating to rust, 2024. URL https://arxiv.org/abs/2405.11514.

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Language model cascades: Token-level uncertainty and beyond. arXiv preprint arXiv:2404.10136, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. Proceedings of the International Conference on Learning Representations (ICLR), 2021a.

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

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. URL https://arxiv.org/abs/1502.03167.

Joel Jang, Seungone Kim, Seonghyeon Ye, Doyoung Kim, Lajanugen Logeswaran, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Exploring the benefits of training expert language models over instruction tuning. In International Conference on Machine Learning, pp. 14702–14729. PMLR, 2023.

Ehsan Kamalloo, Shivani Upadhyay, and Jimmy Lin. Towards robust qa evaluation via open llms. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '24, pp. 2811–2816, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657675. URL https://doi.org/10.1145/3626772.3657675.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Siqi Kou, Lanxiang Hu, Zhezhi He, Zhijie Deng, and Hao Zhang. Cllms: Consistency large language models, 2024. URL https://arxiv.org/abs/2403.00835.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In Proceedings of the 29th Symposium on Operating Systems Principles, pp. 611–626, 2023.

Jan Leike, Victoria Krakovna, et al. Specification gaming examples in ai. https://openai.com/research/specification-gaming, 2020.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2023. URL https://arxiv.org/abs/2211.17192.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. arXiv preprint arXiv:2311.08692, 2023.

Zhenyi Lu, Chenghao Fan, Wei Wei, Xiaoye Qu, Dangyang Chen, and Yu Cheng. Twin-merging: Dynamic integration of modular expertise in model merging, 2024. URL https://arxiv.org/abs/2406.15479.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. Specinfer: Accelerating generative large language model serving with tree-based speculative inference and verification. arXiv preprint arXiv:2305.09781, 2023.

Alireza Mohammadshahi, Arshad Rafiq Shaikh, and Majid Yazdani. Routoo: Learning to route to large language models effectively, 2024. URL https://arxiv.org/abs/2401.13979.

Mohammed Muqeeth, Haokun Liu, Yufan Liu, and Colin Raffel. Learning to route among specialized experts for zero-shot generalization, 2024. URL https://arxiv.org/abs/2402.05859.

Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Seungyeon Kim, Neha Gupta, Aditya Krishna Menon, and Sanjiv Kumar. Faster cascades via speculative decoding, 2024. URL https://arxiv.org/abs/2405.19261.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data, 2024. URL https://arxiv.org/abs/2406.18665.

Oleksiy Ostapenko, Zhan Su, Edoardo Maria Ponti, Laurent Charlin, Nicolas Le Roux, Matheus Pereira, Lucas Caccia, and Alessandro Sordoni. Towards modular llms by building and reusing a library of loras, 2024. URL https://arxiv.org/abs/2405.11157.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (eds.), Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pp. 487–503, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.39. URL https://aclanthology.org/2021.eacl-main.39.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In Advances in Neural Information Processing Systems, volume 36, 2023.

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

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. URL https://arxiv.org/abs/1910.01108.

Shannon Zejiang Shen, Hunter Lang, Bailin Wang, Yoon Kim, and David Sontag. Learning to decode collaboratively with multiple language models. arXiv preprint arXiv:2403.03870, 2024.

Kv Aditya Srivatsa, Kaushal Maurya, and Ekaterina Kochmar. Harnessing the power of multiple minds: Lessons learned from LLM routing. In Shabnam Tafreshi, Arjun Akula, João Sedoc, Aleksandr Drozd, Anna Rogers, and Anna Rumshisky (eds.), Proceedings of the Fifth Workshop on Insights from Negative Results in NLP, pp. 124–134, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.insights-1.15. URL https://aclanthology.org/2024.insights-1.15/.

Dimitris Stripelis, Zhaozhuo Xu, Zijian Hu, Alay Dilipbhai Shah, Han Jin, Yuhang Yao, Jipeng Zhang, Tong Zhang, Salman Avestimehr, and Chaoyang He. TensorOpera router: A multi-model router for efficient LLM inference. In Franck Dernoncourt, Daniel Preoţiuc-Pietro, and Anastasia Shimorina (eds.), Proceedings of the 2024 Conference on Empirical

Methods in Natural Language Processing: Industry Track, pp. 452–462, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-industry.34. URL https://aclanthology.org/2024.emnlp-industry.34/.

Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Rozière, Jacob Kahn, Daniel Li, Wen tau Yih, Jason Weston, and Xian Li. Branch-train-mix: Mixing expert llms into a mixture-of-experts llm, 2024. URL https://arxiv.org/abs/2403.07816.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421.

Hanqing Wang, Bowen Ping, Shuo Wang, Xu Han, Yun Chen, Zhiyuan Liu, and Maosong Sun. Lora-flow: Dynamic lora fusion for large language models in generative tasks, 2024a. URL https://arxiv.org/abs/2402.11455.

Hongyi Wang, Felipe Maia Polo, Yuekai Sun, Souvik Kundu, Eric Xing, and Mikhail Yurochkin. Fusing models with complementary expertise. In The Twelfth International Conference on Learning Representations, 2024b.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Xun Wu, Shaohan Huang, and Furu Wei. Mixture of lora experts, 2024. URL https://arxiv.org/abs/2404.13628.

Jingwei Xu, Junyu Lai, and Yunpeng Huang. Meteora: Multiple-tasks embedded lora for large language models, 2024. URL https://arxiv.org/abs/2405.13053.

A Acknowledgment

This work was supported in part by the National Institutes of Health (NIH) under grants R01AG079291, R01AG085581 and R01HL165061, National Science Foundation (NSF) under grants CNS2414087, NSF BCS2040381, NSF IIS2123952, NSF IIS1955532, NSF IIS2123952, NSF IIS2311990, and the Semiconductor Research Corporation (SRC) AIHW award 2024AH3210.

B Dataset Description

In this section, we describe our benchmark datasets with more details.

B.1 Commonsense QA

CommonsenseQA is a large-scale, multiple-choice question-answering dataset designed to challenge and evaluate systems on their ability to leverage commonsense knowledge. The dataset consists of 12,102 questions, each accompanied by one correct answer and four distractor (incorrect) options, requiring models to distinguish the correct answer by understanding various types of commonsense reasoning. What sets CommonsenseQA apart is its emphasis on requiring a broader array of everyday knowledge, involving not only basic facts but also causal, temporal, and conceptual reasoning.

B.2 ARC-Challenge

The AI2 ARC dataset is a comprehensive collection of 7,787 grade-school-level multiple-choice science questions, meticulously curated to stimulate advancements in question-answering systems. The dataset is strategically divided into two subsets: the ARC-Easy Set and the ARC-Challenge Set. The ARC-Challenge Set, which is the subset we utilized in our work, comprises a selection of particularly difficult questions. These questions were specifically included because they were misclassified by both a traditional retrieval-based algorithm and a word co-occurrence algorithm, making them a true test of a model's ability to understand and reason through complex scientific concepts. The ARC-Challenge subset serves as an ideal benchmark for testing sophisticated models, as it presents questions that require more than surface-level understanding or simple pattern matching.

B.3 MMLU-Professional Psychology

The MMLU dataset is a comprehensive multitask benchmark that comprises multiple-choice questions across a vast range of knowledge domains, including subjects in the humanities, social sciences, hard sciences, and other fields. It covers 57 distinct tasks such as elementary mathematics, U.S. history, computer science, law, and more, aimed at evaluating a model's general world knowledge and problem-solving capabilities.

In our work, we focused specifically on the "Professional Psychology" subset of MMLU. This subset contains questions rich in domain-specific terminology, including specialized terms related to psychology and, occasionally, biological concepts tied to psychological phenomena. It provides a robust test for assessing a model's proficiency in understanding and reasoning within a specialized academic field, thus offering insights into the model's capability to handle complex, domain-specific content.

B.4 GSM8k

GSM8k (Grade School Math 8k) is a dataset consisting of 8.5K high-quality, linguistically diverse grade school math word problems. Designed to evaluate and improve question-answering capabilities in basic mathematical problem-solving, this dataset emphasizes multi-step reasoning, requiring between 2 and 8 steps to arrive at the correct solution.

The problems involve a sequence of elementary calculations using basic arithmetic operations—addition, subtraction, multiplication, and division—along with some early Algebra concepts. However, the dataset ensures that all problems are approachable for a bright middle school student, avoiding the need for advanced mathematical tools like variable definitions in most cases.

One of the distinctive features of GSM8K is that the solutions are presented in natural language rather than purely in mathematical expressions. This design decision aligns with the dataset's goal to illuminate the reasoning capabilities of large language models (LLMs), specifically how they simulate an "internal monologue" when reasoning through problems. The dataset's natural language solutions provide a more interpretable and instructive resource for evaluating the logical progression of LLMs in real-world tasks.

B.5 MATH

The Mathematics Aptitude Test of Heuristics (MATH) dataset consists of an extensive set of 12,500 intricate mathematical problems curated from prestigious competitions, such as the AMC 10, AMC 12, and AIME Hendrycks et al. (2021b). Each problem is provided alongside a fully worked-out solution, offering step-by-step reasoning that facilitates both answer derivation and explanation generation. Covering a broad spectrum of mathematical topics—including Prealgebra, Algebra, Number Theory, Counting and Probability, Geometry, Intermediate Algebra, and Precalculus—the dataset serves as a rigorous benchmark for mathematical reasoning.

To enable a structured evaluation of model capabilities, problems are stratified into five difficulty levels (1 to 5), ensuring a progressive challenge across varying levels of complexity. The solutions are typeset in LaTeX, maintaining precision and coherence in mathematical notation. Notably, the MATH dataset emphasizes problems that extend beyond direct formula application, frequently requiring creative heuristics and nontrivial problem-solving strategies. This characteristic makes it particularly valuable for training and assessing models in advanced mathematical reasoning and structured solution generation.

C Prompt

In this section, we illustrate the prompt we used for each dataset.

Multiple-Choice Question Prompt

For multiple-choice question datasets, including the Commonsense QA dataset, the ARC-Challenge dataset, and the MMLU-Professional Psychology dataset, we leverage the following prompt to require LLMs to provide both an explanation and a final answer in a specific format.

Example:

```
Question: $question
Choices:
A. $choice_A
B. $choice_B
C. $choice_C
D. $choice_D
E. $choice_E

Please format your response in the following way: [Explanation]. Therefore,
the answer is answer (label). Ensure the final sentence includes the answer
followed by the label in parentheses.
Answer and Reasoning: $response
```

C.1 GSM8k Math Problem Prompt

For math problems in the GSM8k dataset and the MATH, the following prompt is employed to require LLMs to provide a step-by-step solution with clear reasoning in LaTeX format, with the final answer enclosed in a box using \boxed{}.

Example:

```
Solve the following math problem step by step. Ensure all reasoning is shown
clearly in LaTeX format, and the final answer should be enclosed using the
LaTeX command \boxed{} to display it in a box.
Question: $question
Solution: $solution
```

D Examples

In this section, we list examples of each dataset, including the questions, prompts, and responses from LLMs to clearly demonstrate the prompt we use and the task of each dataset.

D.1 Commonsense QA

```
Question: What do people aim to do at work?
Choices:
A. complete job
B. learn from each other
C. kill animals
D. wear hats
E. talk to each other

Please format your response in the following way: [Explanation]. Therefore,
the answer is answer (label). Ensure the final sentence includes the answer
followed by the label in parentheses.
Answer and Reasoning: The answer should be something that people aim to do at
work. People generally aim to complete jobs. Therefore, the answer is complete
job (A).
```

D.2 ARC-Challenge

```
Question: Juan and LaKeisha roll a few objects down a ramp. They want to see
which object rolls the farthest. What should they do so they can repeat their
investigation?
Choices:
A. Put the objects in groups.
B. Change the height of the ramp.
C. Choose different objects to roll.
D. Record the details of the investigation.

Please format your response in the following way: [Explanation]. Therefore,
the answer is answer (label). Ensure the final sentence includes the answer
followed by the label in parentheses.
Answer and Reasoning: To repeat their investigation, Juan and LaKeisha should
record the details of the investigation. This will involve noting down the
objects they rolled, the height of the ramp, and the distance each object
rolled. This way, they can replicate the exact conditions and compare the
results of their repeated investigation. This is the crucial step in the
scientific method, as it ensures that the results are reliable and can be
trusted. Therefore, the final answer is record the details of the
investigation (D).
```

D.3 MMLU-Professional Psychology

```
Question: Justice primarily involves ___________.
Choices:
A. Recognizing fairness for all persons
B. Promoting accuracy and truthfulness
C. Benefitting others and do no harm
D. Respecting cultural and individual difference

Please format your response in the following way: [Explanation]. Therefore,
the answer is answer (label). Ensure the final sentence includes the answer
followed by the label in parentheses.
Answer and Reasoning: Justice primarily involves recognizing fairness for all
individuals. It is about ensuring that everyone is treated equally and fairly,
without any discrimination or bias. Therefore, the answer is Recognizing
fairness for all persons (A).
```

D.4 GSM8K

```
Solve the following math problem step by step. Ensure all reasoning is shown
clearly in LaTeX format, and the final answer should be enclosed using the
LaTeX command \boxed{} to display it in a box.
Question: Janet's ducks lay 16 eggs per day. She eats three for breakfast
every morning and bakes muffins for her friends every day with four. She sells
the remainder at the farmers' market daily for \$2 per fresh duck egg. How
much in dollars does she make every day at the farmers' market?
Solution: 16 - 3 = << 16 - 3 = 13 >> 13 eggs are left after breakfast. 13 - 4
= << 13 - 4 = 9 >> 9 eggs are left after baking. 9 * 2 = << 9 * 2 = 18 >> 18
dollars she makes at the farmers' market. The final answer is: \\boxed{18}.
```

D.5 MATH

```
Solve the following math problem step by step. Ensure all reasoning is shown
clearly in LaTeX format, and the final answer should be enclosed using the
LaTeX command \boxed{} to display it in a box.
Question: How many vertical asymptotes does the graph of
$y=\\frac{2}{x^2+x-6}$ have?
Solution: The vertical asymptotes of a rational function occur at the values
of $x$ for which the denominator of the function is equal to zero. \n\nSo, to
find the vertical asymptotes of the function $y=\\frac{2}{x^2+x-6}$, we need
to solve the equation $x^2+x-6=0$ for $x$.\n\nThis is a quadratic equation,
and we can solve it by factoring. The equation $x^2+x-6=0$ can be factored
into $(x-2)(x+3)=0$.\n\nSetting each factor equal to zero gives the solutions
$x=2$ and $x=-3$.\n\n Therefore, the graph of $y=\\frac{2}{x^2+x-6}$ has $2$
vertical asymptotes, at $x=2$ and $x=-3$.\n The final answer is: \\boxed{2}
```

E Data Transformation Calculation and the Memory-bound LLM Generation

In this appendix, we first analyze the computational characteristics of large language models (LLMs) during the decoding phase and then conclude that this phase is fundamentally memory-bound on modern hardware, such as NVIDIA H100 GPUs. Finally, we illustrate the calculation of the data transformation required for each token during the decoding process.

To analyze the generation process of LLMs, we begin by illustrate their structures. Each Transformer block in a decoder-only LLM consists of the following components:

- **LayerNorm**
- **Multi-Head Self-Attention (MHSA)**: includes linear projections for queries (Q), keys (K), and values (V), scaled dot-product attention, and an output projection.
- **Residual Connection**
- **LayerNorm after Self-Attention**
- **Feedforward Network (FFN)**: typically two linear layers with an activation in between, often of shape $d \to 4d \to d$.
- **Residual Connection**

During decoding, tokens are generated one at a time. To avoid recomputation of attention over previous tokens, modern implementations cache the key and value projections from previous steps in GPU memory, referred to as the *KV cache*. Moreover, *FlashAttention* is employed to efficiently compute attention within a single fused kernel, minimizing data movement and maximizing usage of on-chip memory.

Because of these optimizations, each new token only requires computing its query vector and performing attention against cached keys and values. This reduces both computation and data movement compared to training or prompt processing.

Let d be the hidden dimension, l the number of layers, m the length of the current context (i.e., number of cached tokens), and assume float16 precision (2 bytes per element). We now analyze the compute and memory access for each component in a single Transformer layer during decoding of one token:

- **LayerNorm**: Requires reading and writing a d-dimensional vector, calculating the mean and variance and used them for nomalization.
  Memory: read d inputs, wriet d outputs, $2d$ memory access in total.
  Compute: $4d$ FLOPs.
- **Q projection**: Matrix-vector product ($1 \times d$ multiply $d \times d$).
  Memory: read d inputs and weights (d^2), write d outputs, $d^2 + 2d$ memory access in total.
  Compute: $2d^2$ FLOPs
- **K/V projection**: Not needed during decoding, as keys/values are cached.
- **Attention (FlashAttention)**:
  - Read $m \cdot d$ cached keys and $m \cdot d$ cached values.
  - Compute attention scores and weighted sum over m past tokens.

  Memory: read $2md$ inptus, wriet d outputs, $2md + d$ memory access in total.
  Compute: $4md + 2m$ FLOPs (QK matmul + attention weighted sum)
- **Output projection**: Matrix-vector product ($1 \times d$ multiply $d \times d$).
  Memory: read d inputs and weights (d^2), write d outputs, $d^2 + 2d$ memory access in total.
  Compute: $2d^2$ FLOPs
- **FFN**: Two linear layers: $d \to 4d \to d$ with an activation in between
  Memory: read $d + 4d + 4d$ inputs and weights ($8d^2$), write $4d + 4d + d$ outputs, $8d^2 + 18d$ memory access in total.
  Compute: $8d^2 + 4d + 8d^2 = 16d^2 + 4d$ FLOPs

Summing over all components, the total computation and memory per layer per token is:

$$\begin{aligned}\text{FLOPs per layer} &= 4d + 2d^2 + 4md + 2m + 2d^2 + 16d^2 + 4d \\ &= 20d^2 + 4md + 8d + 2m \text{ FLOPs}\end{aligned} \tag{6}$$

$$\begin{aligned}\text{Memory access per layer} &= 2 * (2d + d^2 + 2d + 2md + d + d^2 + 2d + 8d^2 + 18d) \\ &= 20d^2 + 4md + 50d \text{ bytes}\end{aligned} \tag{7}$$

Assume a typical setup with $d = 8192$, $m = 1024$ (context length), float16 (2 bytes). The per layer FLOPs will be $20d^2 + 4md + 8d + 2m \approx 1.28$GFLOPs and the memory accessed per layer will be $20d^2 + 4md + 50d \approx 1.28$GB. Thus, the compute-to-memory ratio is ≈ 1 FLOPs per byte

On the other hand, an NVIDIA H100 GPU has Peak FP16 Tensor Core throughput: ~ 2000 TFLOPs/s and peak memory bandwidth: ~ 3 TB/s. Thus, the compute-to-memory ratio: ~ 666.67 FLOPs per byte.

The actual compute-to-memory ratio of decoding is much lower (~ 1 FLOPs/byte) than what the H100 GPU hardware is capable of (~ 666.67 FLOPs/byte). Therefore, decoding in LLMs is significantly **memory-bound**: performance is bottlenecked by memory bandwidth rather than compute throughput. This suggests that optimizations that reduce memory movement can have a substantial impact on inference speed and that the memory movement amount can be a great metric for the theoretical analysis of inference speed.

Therefore, in our experiments, we record the generated source of each token for both the query-level routing methods and the token-level routing methods and leverage Equation 7 to calculate the data transformation amount that occurred during the whole generation process. Similarly, for speculative decoding, the generated source of each token is also recorded, and a similar equation, where the output token of one forward pass is changed from a single token to multiple tokens, is employed to calculate the data transformation amount. All the additional data transformation introduced by additional structure (such as the router) in those methods are also included properly. In addition, we deploy the SLM and the LLM on the same device, so there is no switch cost and additional data transformation when we switch between those two models. Finally, we employ the data transformation amount to indicate the computation cost of the generation process. This improves the reproducibility of our experimental results and avoids the result deviation caused by different hardware devices and experimental environments due to the selection of indicators such as inference time.