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 Labs, [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 (LLMs) have achieved remarkable success in natural language processing tasks but suffer from high computational costs during inference, limiting their deployment in latency-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 introduces a token-level routing mechanism, enabling efficient collaboration between small and large language models (SLMs & LLMs). Specifically, CITER enables routing non-critical tokens to an SLM to reduce computational overhead, while critical tokens are processed by an LLM to maintain generation quality. We formulate the training of the router as a reinforcement learning task, where the router receives rewards based on both the quality of predictions and the inference cost of generation. To further accelerate the reward evaluation process, we introduce a shortcut for reward function estimation, significantly reducing the cost of the reward estimation. Extensive experiments demonstrate that CITER reduces inference cost while preserving high-quality generation, offering a promising solution for real-time and resource-constrained applications.

1 Introduction

Large language models (LLMs) have revolutionized a wide range of natural language processing tasks, from machine translation to question answering (Coleman et al., 2024; Kamalloo et al., 2024). However, their impressive performance comes with a substantial computational cost, particularly during inference. As these models grow in size, the cost of inference becomes a significant barrier to their practical deployment, especially in real-time applications. Thus, there is a growing need for accelerating the inference process without compromising the quality of the generated outputs.

Among the strategies (Sanh et al., 2020; Anagnostidis et al., 2024) to reduce inference costs, routing tasks to models of different sizes is a promising approach to accelerating LLM inference while maintaining output quality. In this approach, small language models (SLMs) handle simpler tasks with lower computational overhead, while more complex cases are routed to LLMs to ensure response accuracy. However, while promising, existing works largely focus on routing entire user queries to different models for generation (Ong et al., 2024; Mohammadshahi et al., 2024), which limits routing flexibility and may reduce efficiency.

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 that predicts whether a token is important by estimating the token-level routing score, and routes it to the appropriate model to balance the efficiency and accuracy of generation. We formulate a reinforcement learning (RL) problem to train the router, with the objective of minimizing inference cost while preserving output quality. However, training the router using RL can be computationally expensive. To make it more practical,

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

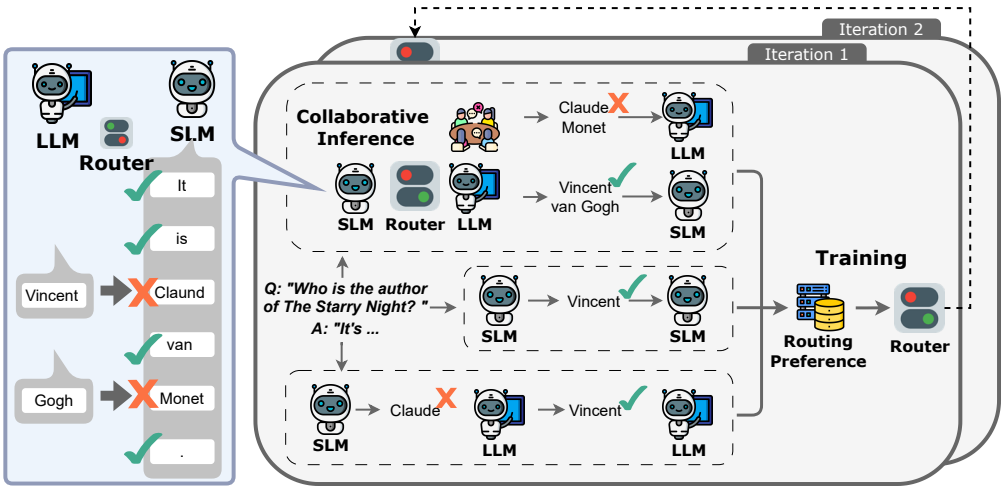


Figure 1: An overview of the CITER framework. A router is leveraged to perform collaborative inference between the SLM and LLM. The router is trained using routing preference collected through three cases. **Case 1**: The SLM generates the correct token, the routing preference is assigned to the SLM. **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.

we present a shortcut to the reward function, significantly accelerating the training process. Through this RL framework, the router learns an optimal token-level decision-making strategy, enabling an SLM and an LLM to collaborate for effective and efficient autoregressive generation.

Our primary contribution is CITER, which accelerates LLM inference by employing a token-level router to select the appropriate model for generating each token. Experiments on four benchmark datasets demonstrate the effectiveness of our approach, achieving a reduction in LLM inference cost while maintaining high output accuracy, with comparable accuracy while up to 30% fewer calls to LLM or delivering up to a 25% improvement in accuracy with the same ratio of calls to LLM compared to our strongest baseline, co-LLM (Shen et al., 2024). Additionally, our ablation study experiments also demonstrate that token-level routing offers more flexibility for achieving more promising results compared to query-level routing and that considering the long-term impact of routing decisions boosts the acceleration performance significantly.

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 leverages token-level routing to accelerate LLM inference. As illustrated in Figure 1, in CITER, we introduce a router to facilitate collaborative inference between a powerful but computationally expensive LLM and a fast but potentially inaccurate SLM. We formulate the training process of the router as a reinforcement learning problem, aiming to minimize the inference cost while maintaining the generation quality. Specifically, we first derive the reward function as token-wise routing preference. Subsequently, we introduce a shortcut for the reward function estimation to significantly accelerate the collection process. Finally, we propose an iterative training process to mitigate the potential inconsistencies of the routing decisions in the preference collection phase and deployment. Next, we will detail the router training and collaborative inference processes.

2.1 Router Training

2.1.1 Efficient Collection of Token-Wise Routing Preferences

To equip the router with the ability to predict token-level routing scores, we formulate the training process of the router as a reinforcement learning problem. Let the current state $\mathcal{T}_h$ be the historical

context input to the LLM before h-th token. At each step $h \in [H]$, the RL agent Π select either the LLM π_L and the SLM π_S with action $a = \{L, S\}$. In addition, we assume the reward $\mathcal{R}(\mathcal{T}_H, \mathcal{A}_H)$ comprehensively encodes the performance of the routing strategy, including the quality of generated response $\mathcal{T}_H$ and the computational cost required by performing $\mathcal{A}_H$. We further define the state-action value function as $Q_h^{\Pi}(\mathcal{T}_h, a) = \mathbb{E}[\mathcal{R}(\mathcal{T}_H, \mathcal{A}_{h:H})|\mathcal{T}_h, a, \Pi]$. The objective of the routing agent is to optimize the policy by

$$\Pi_h = \arg\max_{\Pi_h} \mathbb{E}_{a \sim \Pi_h(\cdot|\mathcal{T}_h)} Q_h^{\Pi}(\mathcal{T}_h, a) - \mathrm{KL}(\Pi_h \parallel \mu), \tag{1}$$

where μ is the pre-defined policy which randomly select the SLM and LLM. Omitting the mathematical derivation, which is detailed in Appendix B, we can obtain the objective function as

$$\mathcal{L} = \sum_{\mathcal{T}_h} \mathbb{1}_h[S \succ L|\mathcal{T}_h, \Pi] \log \Pi_h(S|\mathcal{T}_h) + \mathbb{1}_h[L \succ S|\mathcal{T}_h, \Pi] \log \Pi_h(L|\mathcal{T}_h), \tag{2}$$

where the $\mathbb{1}_h[S \succ L|\mathcal{T}_h, \Pi]$ is the routing preference and the $\mathbb{1}_h[L \succ S|\mathcal{T}_h, \Pi] = 1 - \mathbb{1}_h[S \succ L|\mathcal{T}_h, \Pi]$. To optimize equation 2, the routing preference $\mathbb{1}_h[S \succ L|\mathcal{T}_h, \Pi]$ is determined by whether the fully generated response, starting from state $\mathcal{T}_{h+1}$, is correct. The state $\mathcal{T}_{h+1}$ is reached by taking action S (selecting the SLM) from state $\mathcal{T}_h$. Specifically, If the state $\mathcal{T}_{h+1}$ is not a completed state (e.g., ending with an <EOS> token), the routing agent Π will be leveraged to process the collaborative inference between the SLM and the LLM, obtaining the completed state $\mathcal{T}_H$ at first. Then, the routing preference $\mathbb{1}_h[S \succ L|\mathcal{T}_h, \Pi]$ will be determined as whether the generated response $\mathcal{T}_H$ is correct. The whole process can be formulated as follows:

$$\begin{aligned}
\mathbb{1}_h[S \succ L|\mathcal{T}_h, \Pi] &= \mathbb{1}(\mathrm{Correct}(\mathcal{T}_H)), && \text{if } \mathcal{T}_{h+1} \text{ is completed,} \\
\mathbb{1}_h[S \succ L|\mathcal{T}_h, \Pi] &= \mathbb{1}(\Pi_{h+1}(S|\mathcal{T}_{h+1}) > \tau)\, \mathbb{1}_{h+1}[S \succ L|\mathcal{T}_{h+1}, \Pi] + && \\
&\quad\ \mathbb{1}(\Pi_{h+1}(S|\mathcal{T}_{h+1}) \leq \tau)\, \mathbb{1}_{h+1}[L \succ S|\mathcal{T}_{h+1}, \Pi], && \text{otherwise,}
\end{aligned} \tag{3}$$

where the $\mathrm{Correct}(\cdot)$ is used to determine whether the generated response is correct. The threshold τ is a hyperparameter that determines the routing score required to select the SLM.

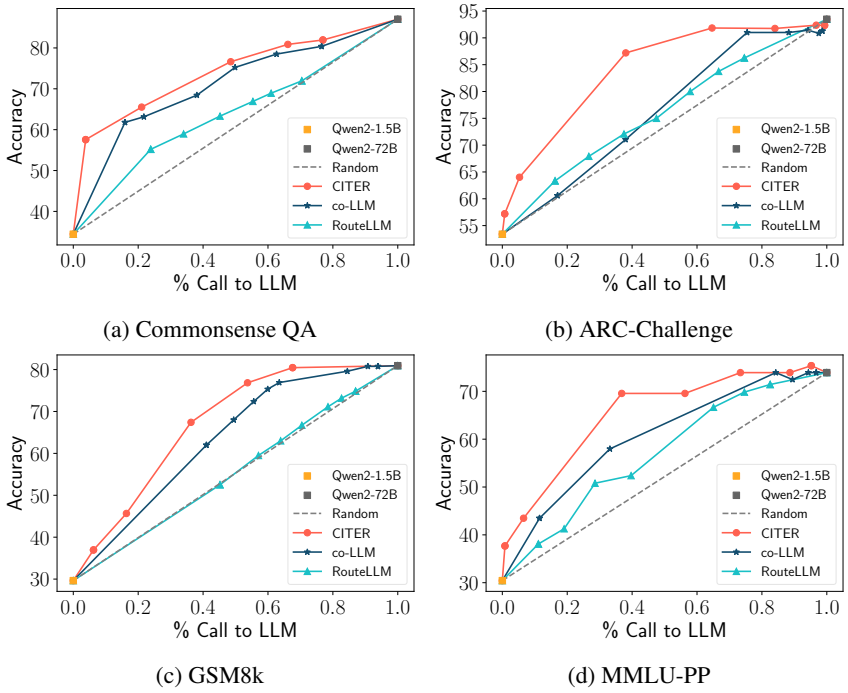
However, the second case in equation 3 requires generating the full response $\mathcal{T}_H$ to obtain the reward, which is computationally expensive. To mitigate this, we introduce a shortcut for estimating the reward, significantly reducing the cost of reward computation. Specifically, we first define the ground truth context before h-th token as $\mathcal{T}_h^*$, and y_h^* as the ground truth h-th token. Subsequently, we feed $\mathcal{T}_h^*$ to both the SLM and LLM to generate the next token. If the SLM can generate the correct token y_h^*, the SLM will be selected. Otherwise, if LLM can generate the correct token y_h^*, the LLM will be selected. 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. Empirically, about $80\% \sim 90\%$ of tokens can be correctly predicted by either the SLM or LLM, making the shortcut significantly reduce the computational cost of the reward function estimation. After collecting the preference, it will be used to train the router using equation 2.

2.1.2 Iterative Training Process

Ideally, we would expect the router Π used during preference data collection to make the same routing decision as the one used during deployment. However, as the router is updated throughout training using the collected data in Section 2.1.1, its behavior is likely to change, leading to potential inconsistencies. To address this issue, we propose a multi-iteration router training process. In each iteration k, the router Π_{k-1} from the previous iteration $k-1$ is used to collect routing preferences. These newly collected routing preferences are then utilized to train a new router Π_k. With each iteration, the router's behavior becomes more consistent, eventually reaching convergence. This iterative process continues until the collected routing preferences match those from the previous iteration or until a predefined number of iterations K is reached. The only exception occurs in the first iteration, where no trained router exists. In this case, a simple routing policy is employed, routing all tokens to the SLM to collect the initial preference data. By following this approach, we can train the router in a practical and efficient manner. The full process is outlined in Appendix G.

2.2 Collaborative Inference

After router training, during inference, we aim to leverage the SLM to collaboratively generate tokens to reduce the number of tokens generated by the LLM, thereby improving the efficiency of



(a) Commonsense QA (b) ARC-Challenge

(c) GSM8k (d) MMLU-PP

Figure 2: The accuracy vs % calls to LLM curve of CITER and the baselines. Points closer to the top-left corner indicate better acceleration performance.

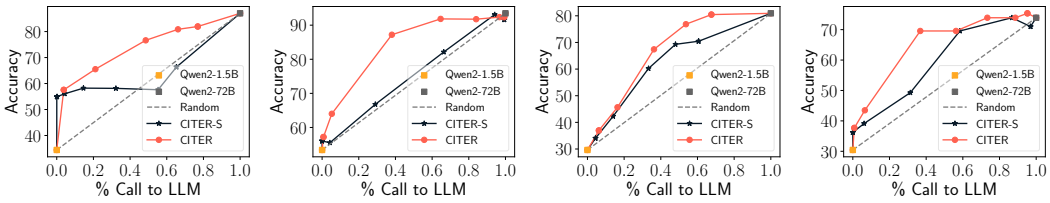
the inference process. Specifically, we start by feeding both the input prompt and the previously generated tokens into the SLM, obtaining the output token and corresponding hidden states. Then, the router Π trained in Section 2.1 is leveraged to predict the token-level routing score based on the hidden states from the SLM. During this process, the router considers both the current token and the historical context to make routing decisions based not only on the accuracy of the current token but also on the long-term influence of its decision on future token generation. Subsequently, a pre-defined threshold τ is used to determine whether the SLM or the LLM should handle the current token. If the routing score exceeds the threshold τ, indicating that the SLM is confident with its output. The output token from the SLM will be committed to the final response and the generation process will go on. Otherwise, the token will be routed to the LLM for re-generation and the SLM's output will be discarded. During the preference data collection process for router training, most tokens are assigned to the SLM, with only a few routed to the LLM through our shortcut. As a result, the SLM efficiently generates the majority of tokens. This process continues until an <EOS> token is produced by either the SLM or LLM. In this way, the router dynamically routes each token between the SLM and LLM, offloading non-critical tokens to the SLM to reduce computational overhead while utilizing the LLM's capabilities to maintain response quality.

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 computational cost and the quality of the generated response? (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?

3.1 Experimental Setup

Dataset and Evaluation. We evaluate CITER and our baselines on four widely-used academic benchmark datasets: the commonsense QA dataset (Talmor et al., 2019), the ARC-Challenge dataset (Clark et al., 2018), the GSM8k dataset (Cobbe et al., 2021) and the MMLU-Professional Psychology dataset (Hendrycks et al., 2021). More details of those datasets are presented in Appendix C. To



(a) Commonsense QA (b) ARC-Challenge (c) GSM8k (d) MMLU-PP

Figure 3: The accuracy vs % calls to LLM curve of CITER and the varient CITER-S. Points closer to the top-left corner indicate better acceleration performance.

compare the performance of CITER with the baselines, we plot the accuracy vs. the % calls to LLM curve 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 inference cost.

Baselines. We compare CITER with both a representative query-level routing method (RouteLLM (Ong et al., 2024)) and a token-level routing method (co-LLM (Shen et al., 2024)). RouteLLM makes routing decisions for entire queries, directing them to different models for generation, while co-LLM breaks down the generation process, routing each token to different models.

3.2 Overall Performance

We conduct extensive experiments to assess the performance of CITER across all benchmark datasets. The results are presented in Figure 2. Notably, all token-level routing methods significantly outperform the query-level routing method, across all datasets, reducing up to 30% calls to LLM while maintaining the same accuracy. This emphasizes the effectiveness of token-level routing, which provides enhanced flexibility in reducing computational costs while preserving response quality. Furthermore, CITER consistently surpasses co-LLM, achieving comparable accuracy with up to 27% fewer calls to LLM. 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.

3.3 Analysis of Long-Term Influence

We also conduct an ablation study on 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 calls to LLM by up to 42% while maintaining the same accuracy. These findings highlight the critical role of accounting for the long-term influence of routing decisions.

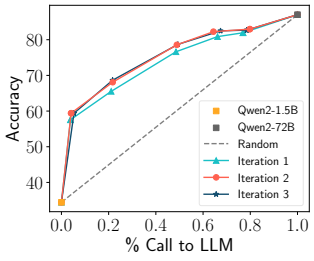


Figure 4: Accuracy vs. % calls to LLM curve of CITER with router over the first three iterations on the commonsense QA datasets. Points closer to the top-left corner indicate better acceleration performance.

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 4, the results demonstrate a clear improvement in performance from the first to the second iteration. In the second iteration, CITER reduces $\sim$ 5% calls to LLM while maintaining the same accuracy 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. The rapid convergence of the router emphasizes the robustness of our training strategy, suggesting that optimal performance can be achieved without excessive computational costs.

References

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

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.

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.

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.

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), 2021.

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.

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.

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.

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. 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.

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 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) 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; 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. 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 computational cost. For example, Arrow (Ostapenko et al., 2024) reuses a library of expert LoRAs to build a mixture-of-experts (MoE) architecture, 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. Among these methods, co-LLM (Shen et al., 2024) is the most relevant to our framework CITER, introducing a router to route tokens to models of different sizes. However, co-LLM only considers the current outputs from SLM and LLM when generating ground truth labels to train the router. 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, co-LLM focuses on enhanced response quality without taking the computational cost 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 (Leviathan et al., 2023; Miao et al., 2023; Kwon et al., 2023; Bhendawade et al., 2024; Cai et al., 2024; Chen et al., 2024b,a) 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.

B Proof of Objective Function

Let the current state $\mathcal{T}_h$ be the historical context input to the LLM before h-th token, i.e., $\mathcal{T}_h = \{\mathbf{x}, y_1, \cdots, y_{h-1}\}$, where $\mathbf{x}$ is the input prompts. At each step $h \in [H]$, the RL agent select either the LLM π_L and the SLM π_S with action $a = \{L, S\}$. Then the state $\mathcal{T}_{h+1}$ is composed by first sampling the next token $y_h \sim \pi_a(\cdot|\mathcal{T}_h)$ and then concatenating it to previous state as $\mathcal{T}_{h+1} = \mathcal{T}_h + \{y_h\}$. In addition, we assume the reward is defined by the whole trajectory by $\mathcal{R}(\mathcal{T}_H, \mathcal{A}_H)$ where $\mathcal{A}_H$ is the sequence of a routing policy, i.e. $\mathcal{A}_H = \{a_1, \cdots a_H\}$, $a_h = \{L, S\}$. Reward $\mathcal{R}$ comprehensively encodes the performance of the routing strategy, including the quality of generated response $\mathcal{T}_H$ and the computational cost required by performing $\mathcal{A}_H$. We further define the (meta) policy for the routing agent as $\Pi = \{\Pi_h(\cdot|\mathcal{T}_h)\}_h$ and the state-action value function as $Q_h^\Pi(\mathcal{T}_h, a) = \mathbb{E}[\mathcal{R}(\mathcal{T}_H, \mathcal{A}_{h:H})|\mathcal{T}_h, a, \Pi]$, where the expectation is taken over $\mathcal{T}_H$ and $\mathcal{A}_{h:H}$ given history context $\mathcal{T}_h$, action $a \in \{L, S\}$ and (meta) policy Π. The objective of the routing agent is to optimize the policy by

$$\Pi_h = \arg\max_{\Pi_h} \mathbb{E}_{a\sim\Pi_h(\cdot|\mathcal{T}_h)} Q_h^\Pi(\mathcal{T}_h, a) - \mathrm{KL}(\Pi_h \parallel \mu), \tag{4}$$

where μ is the pre-defined policy with $\mu(L) = \mu(S) = 0.5$, meaning that the SLM and LLM are randomly selected. The closed-form solution for equation **??** is therefore $\Pi_h(a|\mathcal{T}_h) \propto \mu(a)\exp(Q_h^\Pi(\mathcal{T}_h, a))$.

Generally, it is hard to evaluate the quantity of $\mathcal{R}(\mathcal{T}_H, \mathcal{A}_H)$ because it contains both the quality of $\mathcal{T}_H$ and the efficiency of $\mathcal{A}_H$. We inject the pairwise preference $\mathbb{1}_h[S \succ L]$ following the Bradley–Terry

Table 1: The statistics of our evaluation datasets.

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

model (Bradley & Terry, 1952) as:

$$\Pr_h(S \succ L|\mathcal{T}_h, \Pi) = \frac{1}{1 + \exp(Q^{\Pi}(\mathcal{T}_h, L) - Q^{\Pi}(\mathcal{T}_h, S))} \tag{5}$$

Following (Rafailov et al., 2024), we have that

$$Q_h^{\Pi}(\mathcal{T}_h, L) - Q_h^{\Pi}(\mathcal{T}_h, S) = \log \frac{\Pi_h(L|\mathcal{T}_h)}{\mu(L)} - \log \frac{\Pi_h(S|\mathcal{T}_h)}{\mu(S)} = \log \frac{\Pi_h(L|\mathcal{T}_h)}{\Pi_h(S|\mathcal{T}_h)}, \tag{6}$$

where the equation 6 is due to $\log(\mu(L)/\mu(S)) = 0$. Plugging equation 6 into equation 5 yields

$$\Pr(S \succ L|\mathcal{T}_h, \Pi)_h = \frac{1}{1 + \Pi_h(L|\mathcal{T}_h)/\Pi_h(S|\mathcal{T}_h)} = \Pi_h(S|\mathcal{T}_h), \tag{7}$$

where the last equation is due to the fact that $\Pi_h(S|\mathcal{T}_h) + \Pi_h(L|\mathcal{T}_h) = 1$. Therefore, given a context $\mathcal{T}_h$, once we have labeled the preference $\mathbb{1}_h[S \succ L|\mathcal{T}_h, \Pi]$, $\Pi_h(S|\mathcal{T}_h)$, the routing agent Π can be learned by minimizing the cross-entropy loss

$$\mathcal{L} = \sum_{\mathcal{T}_h} \mathbb{1}_h[S \succ L|\mathcal{T}_h, \Pi] \log \Pi_h(S|\mathcal{T}_h) + \mathbb{1}_h[L \succ S|\mathcal{T}_h, \Pi] \log \Pi_h(L|\mathcal{T}_h), \tag{8}$$

where the $\mathbb{1}_h[L \succ S|\mathcal{T}_h, \Pi]$ is defined similarly to $\mathbb{1}_h[S \succ L|\mathcal{T}_h, \Pi]$, but it takes action L (selecting the LLM) instead of S (selecting the SLM) at step h, conditioned on the state $\mathcal{T}_h$.

C Dataset Description

In this section, we describe our benchmark datasets with more details. The statistics of the datasets are summarized in Table 1.

C.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.

C.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.

C.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.

C.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.

D 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.

E Prompt

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

E.1 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
```

E.2 GSM8k Math Problem Prompt

For math problems in the GSM8k dataset, 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
```

F 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.

F.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).
```

F.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).

F.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).
```

F.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 = 1 3 >> 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}.
```

G The Iterative Training Process of our Router

Algorithm 1: The Iterative Training Process of our Router

Input: The training data $\mathcal{D} = \{(\mathcal{T}_h, y_h^*)\}_i$, the initial router Π_0, the SLM $\mathcal{M}_S$ and LLM $\mathcal{M}_L$, the pre-defined threshold τ and the number of iterations K

Output: The trained router Π

Initialization: iter $k \leftarrow 0$, previous routing preference $\mathcal{S}_0 \leftarrow \emptyset$
for $k = 1$ to K **do**
  $\mathcal{S}_k \leftarrow \emptyset$
  for $i = 1$ to $|\mathcal{D}|$ **do**
    $y_h^S \leftarrow \mathcal{M}_S(\mathcal{T}_h)$
    if $y_h^S == y_h^*$ **then**
      $\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{1\}$
      Continue
    $y_h^L \leftarrow \mathcal{M}_L(\mathcal{T}_h)$
    if $y_h^L == y_h^*$ **then**
      $\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{0\}$
      Continue
    Generate the full response $\mathcal{T}_H$ starting from $\mathcal{T}_h \cup y_h^S$ with router Π_{k-i} and threshold τ
    $\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{\mathbb{1}(\text{Correct}(\mathcal{T}_H))\}$
  if $\mathcal{S}_k == \mathcal{S}_{k-1}$ **then**
    break
  Train the router Π_k with the routing preference $\mathcal{S}_k$ and dataset $\mathcal{D}$ as the loss function equation 2
  $\Pi \leftarrow \Pi_k$
Return Π