Representation-Aware Modularity: Efficient Cross-Task Generalization for Large Language Models

Anonymous ACL submission

Abstract

Cross-task generalization (CTG) enables large language models (LLMs) to handle unseen tasks proficiently, enhancing their adaptability in real-world scenarios. However, existing methods relying on per-token dynamic routing to multiple trained LoRA adapters face high computational and GPU memory costs. Recent Representation Fine-Tuning (ReFT) enhances efficiency for single-task adaptation by editing only prefix and suffix token representations. However, the semantic ambiguity of tokens and absence of a self-guided mechanism for parameter selection in unseen tasks limits their application to CTG. To this end, we propose RaMod, a Representation-Aware Modularity framework to extend the ReFT paradigm to CTG through two novel components: (i) Dual-Modular Representation & Parameter Fine-tuning, which manipulates only a strategically chosen subset of hidden representations with modular interventions to guide the model toward solving unseen tasks; and (ii) Asynchronous Orchestrator, which proactively allocates and releases GPU memory for selected interventions, thereby minimizing storage overhead. Extensive experiments demonstrate that RaMod not only achieves superior CTG performance but also substantially reduces the overhead of the latest CTG baseline, achieving 83%, 100%, and 79% reduction in its additional prefill time, generation delays, and memory consumption relative to the original LLM, respectively.

1 Introduction

The growing proliferation and accessibility of open-source large language models (LLMs) has necessitated the development of specialized variants adapted to particular downstream tasks, domain-specific applications, and diverse user requirements (Li and Liang, 2021; Liu et al., 2022b). Many existing approaches (Liu et al., 2022a; Zhang et al., 2023; Huang et al., 2025b) rely on an *adapter*-based framework, such as Low-Rank Adaptation (Hu et al., 2022), which employs parameter-efficient fine-tuning (PeFT), based on downstream training samples, to adapt a pre-trained LLM for the aforementioned process. Furthermore, real-world scenarios often require LLMs to effectively handle *unseen tasks* they have not encountered before, a capability called *cross-task generalization* (CTG) (Ye et al., 2021; Mishra et al., 2022)[1]. Prior CTG research has demonstrated that employing a mixture of trained adapters, e.s.p LoRAs, can enhance the model's generalization ability to unseen tasks (Belofsky, 2023; Agiza et al., 2024).

However, as illustrated in Figure 1, most existing CTG approaches (Xu et al., 2025; Lin et al., 2025; Huang et al., 2024; Belofsky, 2023; Ostapenko et al., 2024) exhibit inefficiency when handling unseen tasks. (i) **They typically require routing *each token* in new inputs to multiple trained adapters**, a process analogous to the Mixture of Experts (MoE) architecture (Shazeer et al., 2017), which suffers from load imbalance and inefficiency (Zhou et al., 2022). (ii) Since each token is dynamically routed to multiple trained adapters, this prevents the LoRA weights from being merged into the original model parameters as in single-task scenarios in the CTG context. Consequently, **all these trained adapters need to be stored in GPU HBM (High Bandwidth Memory) to maintain inference efficiency**, which imposes a heavier memory burden, thereby degrading throughput. A natural and fundamental question arises here: *how to enhance LLMs' cross-task generalization while mitigating adapter overhead, even achieving near-lossless latency comparable to original LLMs?*

The inefficiency inherent in per-token adaptation motivates a compelling alternative: if comparable accuracy can be achieved by fine-tuning only a subset of tokens, significant efficiency gains could

[1] We focus on *zero-shot* cross-task generalization, *i.e.*, without any exemplar data from the unseen tasks.

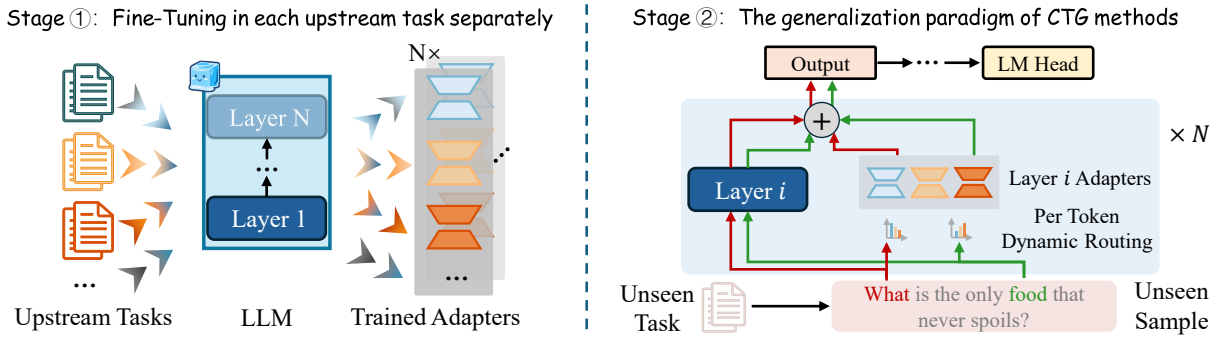

Figure 1: Existing CTG approaches (Xu et al., 2025; Huang et al., 2024; Belofsky, 2023; Ostapenko et al., 2024) typically begin by fine-tuning LoRA as task-specific adapters for each upstream task. When generalizing to unseen tasks, these methods dynamically compute routing weights for each token across all LoRA adapters, then aggregate the outputs through weighted summation to obtain updated token representations at each layer.

be realized. Recently proposed Representation Fine-Tuning (ReFT) methods (Wu et al., 2024b; Geiger et al., 2021) offer a promising direction by editing the representation space of only prefix and suffix tokens in the input prompt, thereby improving computational efficiency for single-task adaptation. Nevertheless, ReFT in its current form exhibits two major limitations when considered for CTG: (1) its static selection of prefix and suffix tokens is inflexible, potentially overlooking critical, context-dependent cues in the middle of the sequence that are vital for coherent generation. (2) ReFT assumes a known, single task, thus lacking any internal mechanism to self-guide the parameter selection for unseen tasks. This inherently limits its scalability and necessitates external guidance for each new scenario, contravening the goal of CTG.

To this end, we propose a Representation-Aware Modularity framework, called RaMod, which extends the ReFT paradigm to CTG while addressing these key limitations. Specifically, RaMod introduces a lightweight token selection mechanism that identifies semantically critical tokens in the input during intervention, instead of relying on fixed positional edits. Moreover, RaMod incorporates a training-free routing strategy termed FluidRoute derived from the Task-Projection Dominance theorem and employs rank-1 approximation, enabling the model to adaptively select modular interventions and address unseen tasks. Furthermore, we develop an Asynchronous Orchestrator to pre-allocate GPU memory to interventions in the subsequent layer that are most likely to be required and release the memory promptly after use, thereby reducing HBM usage. Our experiments demonstrate that RaMod not only achieves superior CTG performance but also drastically trims the overhead of the latest CTG baseline, reducing its excess prefill time, generation delays and HBM usage over the original LLM by 83%, 100%, 79%, respectively.

The detailed contributions of our work are:

- To the best of our knowledge, our work is the first to investigate the efficiency of cross-task generalization in LLMs and proposes an effective framework RaMod to address it.
- We introduce (i) a pivot token selection strategy to enhance ReFT for upstream tasks, (ii) Dual-Modular Representation & Parameter Generalization and (iii) Asynchronous Orchestrator to significantly reduce the computational overhead and HBM usage in the process of CTG.
- Our extensive experiments demonstrate the superior efficiency as well as outstanding generalization performance of RaMod framework[2].

2 Preliminary

Problem Statement. Assume we have N distinct *upstream* tasks $\mathbb{T} = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_N\}$ available for LLM fine-tuning, where each task $\mathcal{T}_i$ is associated with a dataset containing a set of samples $\mathcal{D}_i = \{(x_1, y_1), ..., (x_n, y_n)\}$. The union of the training sets comprise our multi-task dataset $\mathcal{D} = \{D_i\}_{i=1}^{N}$. For an unseen task or sample, *cross-task generalization* aims at adapting the model to accommodate the new task $\mathcal{T}'$ or sample $s' \in \mathcal{T}'$.

LoRA *versus* ReFT. Parameter-efficient Fine-Tuning (PeFT) reduces the number of trainable parameters for fine-tuning in downstream tasks. As a common choice, LoRA achieves competitive trade-offs between parameter efficiency and model

[2]Our code will be public upon acceptance.

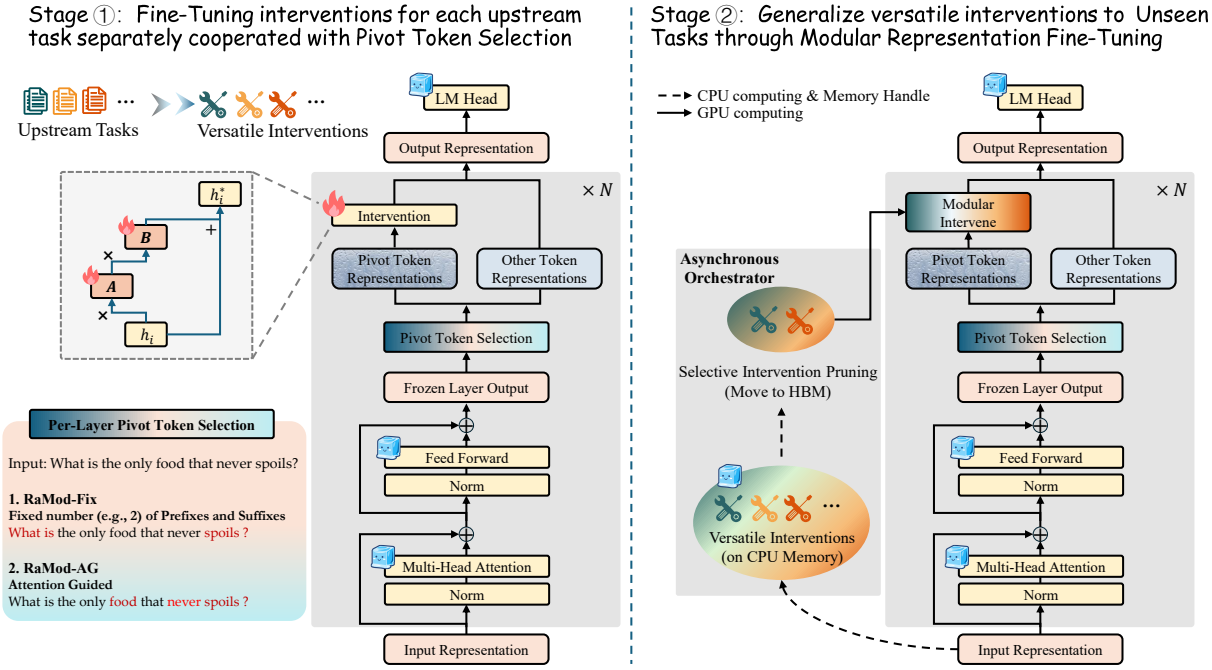

Figure 2: Our CTG framework RaMod consists of two stages. (i) In the first stage, we perform fine-tuning interventions separately for each upstream task. The intervention is applied at each layer to adapt the representations of pivot tokens. (ii) In the second stage, we generalize LLM with well-trained intervenes to unseen tasks. We propose a zero-shot intervention method for modular representation fine-tuning, enabling the model to adapt to unseen tasks effectively. Furthermore, we design an asynchronous orchestrator to dynamically allocate HBM as needed and proactively prune unnecessary interventions, thereby enhancing memory efficiency.

performance (Hu et al., 2022). Given the input hidden state $\boldsymbol{x} \in \mathbb{R}^{d_1 \times 1}$, LoRA modifies the output $\boldsymbol{o} \in \mathbb{R}^{d_2 \times 1}$ of a linear layer as follows:

$$\boldsymbol{o} = \boldsymbol{W}\boldsymbol{x} + \Delta\boldsymbol{W}\boldsymbol{x} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{A}\boldsymbol{x}, \quad \text{(LoRA)}$$

where $\boldsymbol{W}$ are the (frozen) weights of the linear layer, and $\boldsymbol{A} \in \mathbb{R}^{r \times d_1}, \boldsymbol{B} \in \mathbb{R}^{d_2 \times r}$ are two learnable low-rank parameters. LoRA achieves parameter efficiency since the rank $r \ll \{d_1, d_2\}$. Representation Fine-Tuning (ReFT) (Wu et al., 2024b) learns task-specific interventions on hidden representations[3]. As an instance of ReFT, DiReFT can be thought of as LoRA applied directly to hidden representation $\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} \in \mathbb{R}^{d_2 \times 1}$ at certain position set $\mathcal{P}$ in a prompt:

$$\boldsymbol{o} = \begin{cases} \boldsymbol{h} + \boldsymbol{B}(\boldsymbol{A}\boldsymbol{h}), & x \in \mathcal{P}, \\ \boldsymbol{h}, & x \notin \mathcal{P}, \end{cases} \quad \text{(DiReFT)}$$

where $\boldsymbol{A} \in \mathbb{R}^{r \times d_2}, \boldsymbol{B} \in \mathbb{R}^{d_2 \times r}$. In the implementation of DiReFT, the position set $\mathcal{P} = \mathcal{P}_{\text{Fix}}$ consists of a fixed number of p prefix $\{1, ..., p\}$ and s suffix $\{n - s + 1, ..., n\}$ positions in the prompt.

[3]For clarity, we will refer to the LoRA modules as adapters and the changed parameters in DiReFT as interventions. Together, these two will be collectively described as experts.

3 RaMod: Efficient Cross-Task Generalization via Modular ReFT

In this section, we illustrate RaMod, our cross-task generalization (CTG) framework extensively. RaMod comprises two stage, (i) pivot-representation fine-tuning in each upstream task and (ii) dual-modular parameter & representation generalization to unseen tasks, detailed in Sec. 3.1 and Sec. 3.2. At last, we introduce an asynchronous orchestrator for memory efficiency in Sec. 3.3. The overall architecture and algorithm of our framework are shown in Fig. 2 and Alg. 1, respectively.

3.1 Stage 1: Pivot-Representation Fine-tuning

LoRA's efficiency in single-task adaptation stems from the post-training merger of its parameters $(\boldsymbol{W} + \boldsymbol{B}\boldsymbol{A})$. However, when generalizing to cross-task scenarios, LoRA-based approaches employ MoE routing over multiple upstream adapters, and the required per-token adaptive computation precludes this merger, leading to inference inefficiency (Xu et al., 2025; Wu et al., 2024a; Ostapenko et al., 2024; Huang et al., 2024). To mitigate this issue, we apply Representation Fine-tuning (ReFT, Wu et al. (2024b)) to CTG.

A key advantage of ReFT over LoRA is its ability to achieve comparable performance by intervening on the representations of only a small subset of tokens. However, the original ReFT framework is constrained in its expressive power as it rigidly selects a fixed number of prefix and suffix tokens for this intervention. To enhance ReFT, we propose a pivot token selection strategy alternative.

Attention-Guided (AG) Pivot Token Selection. The attention score a token receives can be interpreted as the aggregated focus directed towards its representation by all other token representations in the sequence. A higher attention score implies that a token's representation is more influential in the context of the entire sequence (Ghaeini et al., 2018; Luo et al., 2018; Huang et al., 2025a). Consequently, intervening on the representations of such high-attention tokens is likely to have a more significant impact on the representations of other tokens. Based on this premise, we introduce the following metric to identify pivot tokens in the l-th layer:

$$\mathcal{P}_{\text{AG}}^{(l)} = \underset{j \in \{1,\ldots,|\mathcal{S}|\}}{\arg\text{top-}k_{\text{AG}}} \Big(\sum_{i=j+1}^{|\mathcal{S}|} \mathcal{A}_{ij}^{(l)} / (|\mathcal{S}| - j) \Big), \quad (1)$$

where $\mathcal{A}_{ij}^{(l)}$ denotes the attention score token t_i pays to token t_j in the l-th layer, and $|\mathcal{S}|$ represents the length of the sequence $\mathcal{S}$.

Optimize task-specific interventions. For each upstream task $\mathcal{T}_i \in \mathbb{T}$, we intervene on the per-layer pivot token representations at positions $\mathcal{P} = \mathcal{P}_{\text{AG}}^{(l)}$, which is an instance of our framework we refer to as **RaMod-AG**. Another instance of our framework, **RaMod-Fix**, performs interventions at fixed positions $\mathcal{P} = \mathcal{P}_{\text{fix}}$. The intervention parameters $\{\boldsymbol{A}_{\mathcal{T}_i}^{(l)}, \boldsymbol{B}_{\mathcal{T}_i}^{(l)}\}_{\mathcal{T}_i \in \mathbb{T}}^{l \in \mathbb{L}}$ for each upstream task are optimized by maximizing the likelihood of the target sequence following a language modeling objective, where $\mathbb{L}$ denotes the LLM layers.

3.2 Stage 2: Dual-Modular Parameter & Representation Generalization

After fine-tuning pivot representations for each upstream task, a critical challenge emerges: *how can these versatile, modular interventions be effectively applied to unseen tasks?* Although prior work has leveraged exemplar data (Huang et al., 2024) or task instructions (Charakorn et al., 2025) to fine-tune routing weights, the zero-shot setting, where no information about the unseen task is available, is still a nascent area of research. Existing approaches in this domain are hindered by inefficiencies, requiring the per-token dynamic routing to all trained experts. A comprehensive overview of these methods can be found in Sec. 5.

Based on the modular and versatile interventions trained in the first stage, we propose a dual-modular parameter & representation generalization method, as shown in the right of Fig. 2. First, we employ the same pivot token selection approach as in Stage 1 to identify tokens with significant influence at the l-th layer of the LLM. The representation for each of these pivot tokens is denoted as $\boldsymbol{h}_*$, where we drop the dependency on l for legibility.

FluidRoute: Training-Free & Efficient Routing. In the CTG setting, where no task-specific information is available for unseen tasks, a training-free routing method is required. This method must efficiently and automatically select the most appropriate intervention based on intermediate hidden states to adjust the pivot token representations $\boldsymbol{h}_*$, thus minimizing the impact on LLM inference speed. We first propose to quantify the relevance of each task to the given token using the l_2-norm of the task-parameterized transformation of $\boldsymbol{h}$:

$$\mathcal{R}_{\mathcal{T}_i}(\boldsymbol{h}) = \|\boldsymbol{B}_{\mathcal{T}_i}\boldsymbol{A}_{\mathcal{T}_i}\boldsymbol{h}\|_2. \quad (2)$$

We provide the theoretical guarantee following, and present the empirical analysis in Appendix A.2.

Theorem 1 (Task-Projection Dominance)**.** *Let $P_{\mathcal{T}_i}$ denote the hidden representation distributions in task $\mathcal{T}_i$, the projection magnitude in expectation is maximized for the corresponding task $\mathcal{T}_i$. That is*

$$\mathbb{E}_{\boldsymbol{h}\sim P_{\mathcal{T}_i}} \left(\mathcal{R}_{\mathcal{T}_i}(\boldsymbol{h})\right) > \mathbb{E}_{\boldsymbol{h}\sim P_{\mathcal{T}_i}} \left(\mathcal{R}_{\mathcal{T}_j}(\boldsymbol{h})\right). \quad (3)$$

Proof. The proof is detailed in Appendix A.1. □

However, since computing $\mathcal{R}_{\mathcal{T}_i}(\boldsymbol{h})$ requires passing $\boldsymbol{h}$ through the parameters of each task to obtain the final result entailing substantial computational overhead, we propose an efficient alternative, which we term FluidRoute. FluidRoute begins by applying singular value decomposition (SVD) to the low-rank parameterization of each expert's intervention $\{\boldsymbol{A}_{\mathcal{T}_i}, \boldsymbol{B}_{\mathcal{T}_i}\}$:

$$\boldsymbol{U}_{\mathcal{T}_i}, \boldsymbol{\Sigma}_{\mathcal{T}_i}, \boldsymbol{V}_{\mathcal{T}_i} = \text{SVD}(\boldsymbol{B}_{\mathcal{T}_i}\boldsymbol{A}_{\mathcal{T}_i}). \quad (4)$$

From this decomposition, we extract the principal singular triplet for each expert's intervention: the first right singular vector $\boldsymbol{v}_{1,\mathcal{T}_i} \in \mathbb{R}^{d_2}$, the first left singular vector $\boldsymbol{u}_{1,\mathcal{T}_i} \in \mathbb{R}^{d_2}$ and the largest singular value $\lambda_{1,\mathcal{T}_i}$. These components possess distinct semantic interpretations: $\boldsymbol{v}_{1,\mathcal{T}_i}$ captures the principal input direction to which the expert is most sensitive, $\boldsymbol{u}_{1,\mathcal{T}_i}$ represents the principal output direction

Cateogry	MoE Routing & LoRA Adaptation	RaMod (Ours)
Fine-tuning token	All	A small portion
Modularity	θ	θ & $\boldsymbol{h}$
Affected stage	Prefill & Generation	Prefill
Expert loading	Synchronous	Asynchronous

Table 1: The comparison of RaMod with recent CTG methods (Xu et al., 2025; Lin et al., 2025; Wu et al., 2024a; Ostapenko et al., 2024; Huang et al., 2024) via MoE routing and LoRA adaptation. θ denotes model parameters, and $\boldsymbol{h}$ denotes hidden representations.

of the expert's transformation, and $\lambda_{1,\mathcal{T}_i}$ quantifies the magnitude of this transformation.

The product $\lambda_{1,\mathcal{T}_i}\boldsymbol{u}_{1,\mathcal{T}_i}\boldsymbol{v}_{1,\mathcal{T}_i}^T$ constitutes the best rank-1 approximation of the matrix product $\boldsymbol{B}_{\mathcal{T}_i}\boldsymbol{A}_{\mathcal{T}_i}$. According to the Eckart-Young-Mirsky theorem (Golub et al., 1987), this approximation minimizes the Frobenius norm of the error:

$$\min_{\text{rank}(\hat{\boldsymbol{M}})=1} \left\| \boldsymbol{B}_{\mathcal{T}_i}\boldsymbol{A}_{\mathcal{T}_i} - \hat{\boldsymbol{M}} \right\|_F^2 = \sum_{i=2}^{r} \lambda_{i,\mathcal{T}_i}^2, \quad (5)$$

where the minimum is achieved when $\hat{\boldsymbol{M}} = \lambda_{1,\mathcal{T}_i}\boldsymbol{u}_{1,\mathcal{T}_i}\boldsymbol{v}_{1,\mathcal{T}_i}^T$. To promise the precision and efficiency, we formulate the routing logit of FluidRoute for expert i given input token representation $\boldsymbol{h} \in \mathbb{R}^{d_2 \times 1}$ via $\boldsymbol{u}_{1,\mathcal{T}_i}$ and $\boldsymbol{v}_{1,\mathcal{T}_i}$:

$$\begin{aligned}\hat{\mathcal{R}}_{\mathcal{T}_i}(\boldsymbol{h}) &= \left\|\lambda_{1,\mathcal{T}_i}\boldsymbol{u}_{1,\mathcal{T}_i}\boldsymbol{v}_{1,\mathcal{T}_i}^T\boldsymbol{h}\right\|_2 = |\lambda_{1,\mathcal{T}_i}| \cdot \left\|\boldsymbol{u}_{1,\mathcal{T}_i}\boldsymbol{v}_{1,\mathcal{T}_i}^T\boldsymbol{h}\right\|_2 \\ &\overset{(i)}{=} |\lambda_{1,\mathcal{T}_i}| \cdot \left|\boldsymbol{v}_{1,\mathcal{T}_i}^T\boldsymbol{h}\right| \cdot \|\boldsymbol{u}_{1,\mathcal{T}_i}\|_2 \overset{(ii)}{=} |\lambda_{1,\mathcal{T}_i}| \cdot \left|\boldsymbol{v}_{1,\mathcal{T}_i}^T\boldsymbol{h}\right|,\end{aligned} \quad (6)$$

where step (i) is due to the $\boldsymbol{v}_{1,\mathcal{T}_i}^T\boldsymbol{h}$ being a scalar, and step (ii) is derived from each singular vector $\|\boldsymbol{u}\|_F = 1$. The routing logit of FluidRoute effectively estimates the alignment of the hidden state $\boldsymbol{h}_*$ with the directions specific to task $\mathcal{T}_i$.

Through Theorem 3.1, we can directly estimate the intervention of the most relevant task from the hidden representations during the inference process of an LLM. We employ a top-k routing strategy from the routing logits of all intervenes $\hat{\mathcal{R}}(\boldsymbol{h}_*) = \{\hat{\mathcal{R}}_{\mathcal{T}_i}(\boldsymbol{h}_*)\}_{\mathcal{T}_i \in \mathbb{T}}$ and apply a softmax composition of target intervention parameters $\{\boldsymbol{B}_*, \boldsymbol{A}_*\}$ for each pivot token representation $\boldsymbol{h}_*$:

$$w_{\mathcal{T}_i}(\boldsymbol{h}_*) = \begin{cases} -\infty, & \hat{\mathcal{R}}_{\mathcal{T}_i}(\boldsymbol{h}_*) \notin \text{arg top-}k_\theta\, \hat{\mathcal{R}}(\boldsymbol{h}_*), \\ \hat{\mathcal{R}}_{\mathcal{T}_i}(\boldsymbol{h}_*), & \text{otherwise}, \end{cases} \quad (7)$$

$$\boldsymbol{A}_* = \sum\nolimits_{\mathcal{T}_i \in \mathbb{T}} \text{Softmax}(w_{\mathcal{T}_i}(\boldsymbol{h}_*))\boldsymbol{A}_{\mathcal{T}_i}. \quad (8)$$

The composition of $\boldsymbol{B}_*$ and $\boldsymbol{b}_*$ is defined similarly and omitted for brevity. Building on modular representation fine-tuning and modular intervention parameters, our method significantly improves the efficiency of generalizing to unseen tasks.

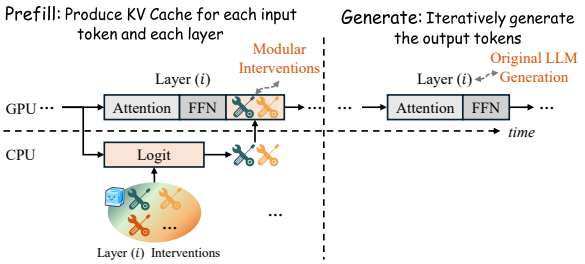

Figure 3: RaMod only intervenes during the prefill phase and leverages our Asynchronous Orchestrator to significantly reduce the memory footprint of modular parameters. The generation process utilizes the original LLM procedure, ensuring efficient generation.

Prefill-only Intervention. A key advantage is that our approach **only** requires modifications on the token representations during the *prefill* stage, *i.e.,* processing the input prompt. This avoids any latency impact on the *generation* process, *i.e.,* producing one token at a time. This is a significant efficiency gain over prior work that required routing for every single token in both prefill and generation phase. As demonstrated by the baseline `Arrow` in Table 2, the routing adaptation applied to each token during the generation process leads to a significant increase in latency, causing a per-token generation delay that is approximately 62 times greater (1.482s / 0.024s $\approx$ 62).

3.3 Asynchronous Orchestrator

Previous works on CTG generally require all experts to be stored in High Bandwidth Memory (HBM). This allows for the immediate determination of requisite experts based on input representations at each layer during inference, enabling subsequent computations. However, as the number of tasks increases, the linear relationship between experts and tasks leads to a significant HBM storage overhead. This, in turn, limits the maximum achievable batch size and adversely affects LLM inference efficiency metrics such as throughput. To mitigate this issue, some methods employ clustering to reduce the number of experts and facilitate parameter sharing (Gabrielsson et al., 2024; Ostapenko et al., 2024).

We propose an orthogonal approach to this problem, called the Asynchronous Orchestrator. The core principle of this approach is to store the parameters for the vast majority of experts in low-cost, high-storage CPU memory. By anticipating which experts will likely be required for upcoming inputs,

it proactively transfers the relevant parameters into HBM just before they are needed for computation. This strategy significantly mitigates the storage burden that expert parameters place on the HBM. This process is depicted in Figure 3.

Based on the common observation that hidden representations between adjacent layers exhibit a high degree of similarity (Lee et al., 2024; Liu et al., 2023b), we hypothesize that the experts required by these layers are also closely related. Therefore, we leverage the input pivot token representations $\boldsymbol{h}_*^{(l-1)}$ from the preceding layer instead of $\boldsymbol{h}_*^{(l)}$ to coarsely choose expert interventions with the top-K_θ ($K_\theta > k_\theta$) frequency ranked by FluidRoute:

$$\left\{\{\boldsymbol{A}_{\mathcal{T}_i}^{(l)}, \boldsymbol{B}_{\mathcal{T}_i}^{(l)}\} | \hat{\mathcal{R}}_{\mathcal{T}_i}(\boldsymbol{h}_*^{(l-1)}) \in \text{top-}K_\theta\ \hat{\mathcal{R}}(\boldsymbol{h}_*^{(l-1)})\right\}. \tag{9}$$

Concurrently with the current layer's computation, the parameters of these selected experts are asynchronously loaded into HBM for use in Stage 2. Upon completion of Stage 2, these parameters are evicted from HBM. This allows the data transfer to overlap with the model's dense computations (e.g., FFNs, attention), thereby reducing the substantial storage overhead of numerous expert parameters in CTG without compromising inference speed.

4 Experiments

In this section, we conduct empirical experiments to demonstrate the efficacy of our RaMod framework. We include additional details and hyperparameter analysis in Appendix A.3.

4.1 Setup

Benchmarks. We train our method and CTG baselines on 256 tasks from the Flan v2 dataset (Longpre et al., 2023). Following Ostapenko et al. (2024) and Wang et al. (2023), we sample 10,000 instances per task and use 1,000 for validation to ensure computational feasibility. For cross-task generalization, we evaluate these methods on 10 widely-used tasks in the zero-shot manner, including: (1) common-sense reasoning: WinoGrande (Sakaguchi et al., 2021), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020); (2) question answering: BoolQ (Clark et al., 2019), OpenbookQA (Mihaylov et al., 2018), ARC-Easy and ARC-Hard (Clark et al., 2018); (3) coding: HumanEval (Liu et al., 2023a) and MBPP (Austin et al., 2021); and (4) general-purpose reasoning: BHH (Suzgun et al., 2023). To ensure a valid evaluation of the LLM's zero-shot generalization capabilities, we remove any training data from the Flan v2 dataset that overlapped with these 10 unseen tasks.

Baselines. We verify the effectiveness of RaMod compared with the following zero-shot baselines: (1) `Base`: the base LLM without any adaptation; (2) `Shared`: a single LoRA expert finetuned on the joint training set of all 256 tasks; (3) `FullFT`: the base LLM is fully fine-tuned like `Shared`; (4) `Arrow` (Ostapenko et al., 2024): A zero-shot dynamic routing mechanism that directs each token to the most relevant parameters among a set of well-trained LoRA modules. Besides these baselines, we compare our method with `MeteoRA` (Xu et al., 2025), a method based on a MoE architecture that adaptively selects LoRA parameters, in order to evaluate the impact of an additional learnable gating mechanism. The gating parameters in MeteoRA are trained using a held-out set of 500 instances from each dataset.

Implementations. Following (Ostapenko et al., 2024), we use the Phi-2 (Javaheripi et al., 2023) and Mistral-7B (Jiang et al., 2024) as the base LLMs. Unless stated otherwise, we set the rank of 4, dropout of 0.05 and learning rate of 1e-4 for all expert modules, *i.e.,* interventions and LoRAs. For our RaMod method, we select the tokens from the calculation in Eqn. (1) to serve as pivot tokens, with k_{AG}=14. The original ReFT method uses a fixed set of 7 prefix and 7 suffix tokens. These two token selection strategies are configured as two variants of our method: **RaMod-AG** and **RaMod-Fix**, respectively. For selecting expert interventions (Eqn. (7)), we choose k_θ=4 expert parameters for each pivot token. Our asynchronous orchestrator uses the CPU to calculate the logits from Eqn. (9) and screens for the top-K_θ most likely to be utilized by the current layer, where K_θ=10. Results are averaged over three runs with distinct random seeds. All experiments were conducted on an Ubuntu 20.04.5 server, equipped with a 2.20GHz Intel Xeon Gold 5220R CPU with 1024GB memory, an NVIDIA A100 (80G) GPU.

4.2 Performance Comparison

The cross-task generalization results are enumerated in Table 2. Overall, our proposed RaMod-AG demonstrates superior performance in CTG across most datasets, achieving optimal average performance. When using Phi-2 as the backbone, RaMod-AG shows an average improvement of 1.0% compared to the second-best baseline, Arrow. With Mistral as the backbone, RaMod-AG improves by

Method \ Eval	Efficiency (s / GB)			Performance (%)										
	TTFT	TPOT	Mem.	WG	HSwag	PIQA	BoolQ	OQA	ARC-E	ARC-C	HE	MBPP	BHH	Avg.
Phi-2 (2.7B)	0.141	0.024	9.0	75.7	72.5	79.2	82.7	49.8	77.5	52.9	45.1	56.0	48.0	63.8
Shared	0.141	0.024	9.0	76.6	73.4	80.4	82.4	50.4	83.2	55.8	46.3	50.4	48.4	65.5
FullFT	0.141	0.024	9.0	77.0	73.2	80.3	80.8	48.0	83.5	57.9	50.0	57.2	47.7	65.6
Arrow	1.925	1.482	19.8	77.6	72.6	80.2	84.3	52.2	84.2	56.4	50.6	59.9	47.7	66.6
MeteoRA	2.169	1.253	20.7	76.1	73.8	79.6	83.2	**52.7**	83.7	56.0	49.3	57.9	48.1	66.0
RaMod-Fix	0.381	0.024	11.6	78.1	**75.2**	80.2	84.1	51.8	84.6	57.9	51.3	60.3	**48.9**	67.2
RaMod-AG	0.474	0.024	11.6	**78.4**	74.9	**80.9**	**84.6**	52.3	**85.3**	**58.3**	**51.7**	**60.8**	48.6	**67.6**
Mistral (7B)	0.162	0.032	14.9	66.5	78.8	81.1	82.2	44.6	68.9	49.6	28.0	47.5	47.9	59.5
Shared	0.162	0.032	14.9	**68.6**	79.5	50.4	84.6	50.4	84.8	60.0	24.4	47.5	49.2	63.1
Arrow	2.637	1.861	32.6	66.6	81.1	82.8	86.6	50.6	85.7	60.8	30.5	**49.4**	49.5	64.4
MeteoRA	3.096	1.803	33.8	67.3	81.2	81.9	86.3	48.7	85.3	60.1	29.1	48.5	49.1	63.8
RaMod-Fix	0.840	0.032	18.5	68.1	81.7	81.6	86.8	50.4	85.9	60.3	29.3	49.1	**50.0**	64.3
RaMod-AG	1.014	0.032	18.5	68.5	**81.9**	**82.9**	**87.4**	**50.9**	**86.1**	**61.3**	**31.4**	48.8	49.8	**64.9**

Table 2: The results of cross-task generalization performance and efficiency for Phi-2 and Mistral backbones. All experiments are conducted using the BF16 precision format. We evaluate efficiency by measuring time (s) and peak GPU memory (GB) with a batch size of 8, processing 2048 input tokens and generating up to 1024 tokens. Prefill is measured by Time-to-First-Token (TTFT), and generation by Time-Per-Output-Token (TPOT).

an average of 0.5% over Arrow and 1.1% over MeteoRA. This indicates that the additional parameters introduced by learnable gating mechanisms pose a higher risk of overfitting. In contrast, our proposed zero-shot routing only intervenes on the representations of a few tokens, which enhances the generalization performance of LLMs. Furthermore, RaMod-AG outperforms RaMod-Fix with a 0.5% improvement on the Phi-2 backbone and a 0.6% improvement on the Mistral backbone. This suggests that intervening on the representations of pivot tokens based on attention scores is more effective at reinforcing sample-specific task features, which improves generalization to unseen tasks.

4.3 Efficiency Gains

The efficiency evaluation of our method and the baselines is also included in Table 2. The evaluation primarily focuses on the maximum GPU memory usage and the time cost of the prefill and generation stages. Shared and FullFT incur identical storage overhead and maintain the same prefill and generation speeds as the original LLM, since each employs only a single expert that can be directly merged into the original model weights or fine-tuned in-place. In contrast, Arrow requires storing all expert parameters in HBM, and MeteoRA not only stores all experts but also trains an additional gating network. Moreover, computing logits for all experts introduces considerable computational overhead. For instance of Phi-2, MeteoRA increases TTFT by 15.4× and TPOT by 52.2× compared with the original LLM. Our approach, however, avoids interference during the generation phase, thus keeping TPOT consistent with the original model. During prefill, we intervene only on a small number of token representations, substantially reducing the memory usage associated with dynamic routing. Furthermore, by leveraging the *Asynchronous Orchestrator* to keep non-essential experts in CPU memory, RaMod-Fix drastically trims the overhead of MeteoRA, reducing its excess TTFT, TPOT and HBM usage over Phi-2 by 88.2%, 100%, 77.8%, and Mistral by 76.9%, 100%, 81.1%, respectively.

Method \ Eval	Efficiency (s / GB)			Avg. Perf.
	TTFT	TPOT	Mem.	(%)
RaMod-AG	0.474	0.024	11.6	67.6
w/o AG (Sec. 3.1)	0.381	0.024	11.6	67.2
w/o A.O. (Sec. 3.3)	0.849	0.024	20.1	67.7

Table 3: Ablation studies evaluating the contribution of key components of our method on Phi-2. A.O. denotes the Asynchronous Orchestrator in Sec. 3.3.

4.4 Ablation Study

In this subsection, we attempt to understand how the attention-guided (AG) pivot token selection and the asynchronous orchestrator (A.O.) contribute to the performance and efficiency of CTG. As shown in Table 3, RaMod-AG yields a notable improvement in CTG performance, achieving a 0.4% gain while reducing TTFT latency by less than 0.1 seconds compared to the variant without AG (i.e., RaMod-Fix in Table 2). Furthermore, by comparing RaMod-AG with the variant where the A.O. is removed, it can be observed that A.O. significantly reduces HBM usage. Benefiting from asyn-

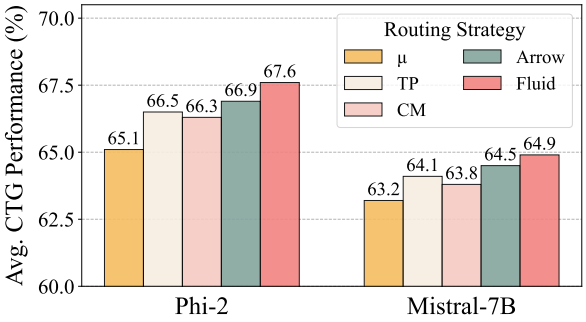

Figure 4: Comparison of routing strategies for average CTG performance across different backbones.

chronous scheduling and the involvement of fewer experts parameters in token representation intervention, the TTFT latency is also substantially reduced. This process filters out less critical experts while maintaining the model's generalization capability without significant loss.

4.5 Routing Strategy Comparison

The effectiveness of the FluidRoute logit proposed in Sec 3.2 is evaluated in Figure 4 by comparing CTG performance across different LLM backbones within the RaMod-AG framework using several routing strategies (extensive descriptions in Appendix A.3): μ-routing (uniform weight allocation), TP-routing (a prompt-dependent classifier trained with cross-entropy loss), CM-routing (prototype-based cosine similarity), and Arrow (using only the first right singular vector). Figure 4 demonstrates the superior CTG performance of our FluidRoute method, which outperforms the second-best baseline (Arrow) by an average of 0.7%. The key advantage of FluidRoute lies in its efficient and precise estimation of expert interventions per token, achieved via an optimal rank-1 SVD approximation and theoretical guarantee of Theorem 1.

5 Related Work

Parameter-efficient Fine-Tuning (PeFT). PeFT has emerged as a prominent approach to mitigate the high computational overhead associated with full fine-tuning of LLMs for downstream tasks (Houlsby et al., 2019; Liu et al., 2022a; Zhang et al., 2023; Huang et al., 2025b). LoRA (Hu et al., 2022) stands out as a prominent example of PeFT. It offers efficient adaptation by updating only injected low-rank matrices while keeping the parameters of the base LLM frozen. Recently, Representation Fine-tuning (Wu et al., 2024b) has been proposed as an alternative to LoRA, achieving high computational efficiency by directly editing a small subset of hidden representations from the LLM.

Cross-task generalization (CTG). The goal of CTG is for LLMs to exhibit strong performance on tasks they have not been explicitly trained on, also known as unseen tasks. CTG is typically divided into two settings depending on the requirement for exemplar data from unseen tasks: few-shot learning (Huang et al., 2024; Ponti et al., 2023; Vu et al., 2022), which uses a small number of labeled examples for alignment, and zero-shot learning (Sanh et al., 2022), which uses none. Wu et al. (2024a) treat each trained LoRA as an expert and integrate learnable gating function to learn composition weights for *each token* independently in the MoE's manner. Xu et al. (2025) further develop a GPU kernel operator for enhancing the efficiency of the naive MoE implementations. Ostapenko et al. (2024) decompose LoRA parameters through SVD and design a training-free routing mechanism based on the correlation between the SVD decomposition results and the token representations. Charakorn et al. (2025) use instruction tuning to generate LoRA parameters conditioned on task description embeddings from a separate, small sentence model. However, the efficiency of existing works is hampered by the need to activate different adapters on a token-by-token basis, the requirement to keep all trained adapters in GPU memory for immediate use, or the dependency on an additional sentence embedding model. To the best of our knowledge, our work is the first to leverage ReFT to enhance efficiency for CTG tasks.

6 Conclusion

This study investigated a crucial research question: efficient cross-task generalization for LLMs. The aim is to not only improve the generalization capabilities of LLMs but also reduce the memory usages of fine-tuned parameters and enhance the inference speed. To tackle this issue, we proposed a Representation-Aware Modularity framework, called RaMod. Within this framework, we developed a modular representation fine-tuning strategy to manipulate a minimal subset of token representations through a series of highly targeted interventions. Furthermore, we implemented an asynchronous orchestrator to dynamically allocate GPU memory for required interventions and promptly release memory upon intervention completion, thereby minimizing memory overhead. Our comprehensive experimental results demonstrated the efficiency and effectiveness of RaMod for LLM cross-task generalization.

Limitations

To ensure a fair comparison with the LoRA-based CTG baselines, our current approach primarily focuses on utilizing a DiReFT transformation, which is very similar to LoRA calculation, as the pivot token representation intervention. While other representation learning methods exist, such as distributed interchange intervention (Geiger et al., 2024) and low-rank linear subspace ReFT (Wu et al., 2024b), we have chosen DiReFT for this study due to its simplicity and effectiveness. As our work is focused on the performance and efficiency of LLMs for the CTG task, we will explore the performance and efficiency of other types of interventions in our future work.

References

Ahmed Agiza, Marina Neseem, and Sherief Reda. 2024. Mtlora: Low-rank adaptation approach for efficient multi-task learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16196–16205.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Joshua Belofsky. 2023. Token-level adaptation of lora adapters for downstream task generalization. In *Proceedings of the 2023 6th Artificial Intelligence and Cloud Computing Conference*, pages 168–172.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. 2025. Text-to-loRA: Instant transformer adaption. In *Forty-second International Conference on Machine Learning*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *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)*, pages 2924–2936.

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

Rickard Brüel Gabrielsson, Jiacheng Zhu, Onkar Bhardwaj, Leshem Choshen, Kristjan Greenewald, Mikhail Yurochkin, and Justin Solomon. 2024. Compress then serve: Serving thousands of lora adapters with little overhead. In *Forty-second International Conference on Machine Learning*.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. *Advances in Neural Information Processing Systems*, 34:9574–9586.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah Goodman. 2024. Finding alignments between interpretable causal variables and distributed neural representations. In *Causal Learning and Reasoning*, pages 160–187. PMLR.

Reza Ghaeini, Xiaoli Fern, and Prasad Tadepalli. 2018. Interpreting recurrent and attention-based neural models: a case study on natural language inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4952–4957.

Gene H Golub, Alan Hoffman, and Gilbert W Stewart. 1987. A generalization of the eckart-young-mirsky matrix approximation theorem. *Linear Algebra and its applications*, 88:317–327.

Gene H Golub and Charles F Van Loan. 2013. *Matrix computations*. JHU press.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2024. Lorahub: Efficient cross-task generalization via dynamic lora composition. In *First Conference on Language Modeling*.

Chenxi Huang, Shaotian Yan, Liang Xie, Binbin Lin, Sinan Fan, Yue Xin, Deng Cai, Chen Shen, and Jieping Ye. 2025a. Enhancing chain-of-thought reasoning with critical representation fine-tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23173–23195.

Weizhong Huang, Yuxin Zhang, Xiawu Zheng, Jing Lin, Yiwu Yao, Rongrong Ji, and 1 others. 2025b. Dynamic low-rank sparse adaptation for large language models. In *The Thirteenth International Conference on Learning Representations*.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, and 1 others. 2023. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 1(3):3.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. 2024. {InfiniGen}: Efficient generative inference of large language models with dynamic {KV} cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 155–172.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

Tianwei Lin, Jiang Liu, Wenqiao Zhang, Yang Dai, Haoyuan Li, Zhelun Yu, Wanggui He, Juncheng Li, Jiannan Guo, Hao Jiang, Siliang Tang, and Yueting Zhuang. 2025. Teamlora: Boosting low-rank adaptation with expert collaboration and competition. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 13622–13637. Association for Computational Linguistics.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022a. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023a. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36:21558–21572.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022b. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, and 1 others. 2023b. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, and 1 others. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pages 22631–22648. PMLR.

Ling Luo, Xiang Ao, Feiyang Pan, Jin Wang, Tong Zhao, Ningzi Yu, and Qing He. 2018. Beyond polarity: Interpretable financial sentiment analysis with hierarchical query-driven attention. In *IJCAI*, pages 4244–4250.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487.

Oleksiy Ostapenko, Zhan Su, Edoardo Ponti, Laurent Charlin, Nicolas Le Roux, Lucas Caccia, and Alessandro Sordoni. 2024. Towards modular llms by building and reusing a library of loras. In *International Conference on Machine Learning*, pages 38885–38904. PMLR.

Edoardo Maria Ponti, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy. 2023. Combining parameter-efficient modules for task-level generalisation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 687–702.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, and 1 others. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny

Zhou, and 1 others. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *ACL (Findings)*.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2022. Spot: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5039–5059.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, and 1 others. 2023. How far can camels go? exploring the state of instruction tuning on open resources. *Advances in Neural Information Processing Systems*, 36:74764–74786.

Xun Wu, Shaohan Huang, and Furu Wei. 2024a. Mixture of lora experts. In *The Twelfth International Conference on Learning Representations*.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. 2024b. Reft: Representation finetuning for language models. *Advances in Neural Information Processing Systems*, 37:63908–63962.

Jingwei Xu, Junyu Lai, and Yunpeng Huang. 2025. Meteora: Multiple-tasks embedded lora for large language models. In *The Thirteenth International Conference on Learning Representations*.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. Crossfit: A few-shot learning challenge for cross-task generalization in nlp. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7163–7189.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, and 1 others. 2022. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114.

A Appendix

A.1 Theoretical Proof

As established in Sec. 3.2, Theorem 1 guarantees that given an input representation for a specific positional task, our proposed routing mechanism exhibits a higher probability of selecting the intervention corresponding to the most analogous task. We proceed to demonstrate the derivation of this theorem through mathematical expectation. Let a pre-trained weight matrix be denoted by $W \in \mathbb{R}^{d\times m}$. For a task $\mathcal{T}_k$, its adaptation is governed by a low-rank matrix $\Delta \boldsymbol{W}_{\mathcal{T}_k} = \boldsymbol{B}_{\mathcal{T}_k}\boldsymbol{A}_{\mathcal{T}_k}$, where $\boldsymbol{B}_{\mathcal{T}_k} \in \mathbb{R}^{d\times r}$, $\boldsymbol{A}_{\mathcal{T}_k} \in \mathbb{R}^{r\times m}$, and $r \ll \min(d, m)$. The adapted forward pass for an input $\boldsymbol{h} \in \mathbb{R}^m$ is $(1 + \boldsymbol{B}_{\mathcal{T}_k}\boldsymbol{A}_{\mathcal{T}_k})\boldsymbol{W}\boldsymbol{h}$. We consider a cross-task generalization setting with upstream tasks $\mathbb{T}$. Each task k has a data distribution $\mathcal{D}_k$. The parameters are optimized by minimizing the following objective:

$$\mathcal{L}_{\mathcal{T}_k} = \mathbb{E}_{\boldsymbol{h}\sim\mathcal{D}_{\mathcal{T}_k}}\left[\|(W + B_{\mathcal{T}_k}A_{\mathcal{T}_k})\,h - y_{\mathcal{T}_k}\|_2^2\right],$$

where y_k is the target output. We aim to demonstrate that for a set of experts $\{\Delta W_1, ..., \Delta W_N\}$, where each expert i is trained on a specific task $\mathcal{T}_i$, the proposed static routing score will, in expectation, be maximized for samples drawn from its corresponding dataset. Given a hidden state $\boldsymbol{h}$ from task $\mathcal{T}_i$, we seek to prove that the routing logit for expert i, defined as:

$$\mathcal{R}_{\mathcal{T}_i}(\boldsymbol{h}) = \|\boldsymbol{B}_{\mathcal{T}_i}\boldsymbol{A}_{\mathcal{T}_i}\boldsymbol{h}\|_F, \quad \boldsymbol{h} \sim \mathcal{D}_k \tag{10}$$

is statistically greater than the logit for any other expert $j \neq i$. To better illustrate the proof, we first illustrate two key definitions.

Definition 1 (Task Subspace)**.** *For a task k, the task subspace U_k is defined as the column space of A_k^T, or equivalently, the row space of A_k: $U_k = span(\text{rows of } A_k) \subset \mathbb{R}^m$.*

Definition 2 (Subspace Similarity)**.** *The similarity between two task subspaces U_i and U_j is defined as the cosine of the first principal angle θ_1 between them:*

$$S(U_i, U_j) = \cos(\theta_1) = \max_{u\in U_i,\, v\in U_j} \frac{|u^T v|}{\|u\|_2\|v\|_2}.$$

Note that $0 \leq S(U_i, U_j) \leq 1$, where 0 implies orthogonality and 1 implies identical subspaces.

The proof relies on two foundational results from matrix analysis and subspace theory.

Lemma 1 (Eckart–Young–Mirsky Theorem, (Golub et al., 1987))**.** *Let $X \in \mathbb{R}^{d\times n}$ be a matrix of rank p, and let $r < p$. The optimal rank-r approximation X_r that minimizes $\|X - \widehat{X}\|_F$ is given by the truncated Singular Value Decomposition (SVD) of X, retaining the r largest*

singular values and their corresponding singular vectors. Furthermore, the approximation error is $\|X - X_r\|_F^2 = \sum_{i=r+1}^{p} \lambda \boldsymbol{A}_{\mathcal{T}_i}^2$.

Lemma 2 (Projection Bound via Principal Angles, (Golub and Van Loan, 2013))**.** *Let* U_i *and* U_j *be subspaces of* $\mathbb{R}^m$ *with first principal angle* θ_1*. Let* P_{U_j} *be the orthogonal projection matrix onto* U_j*. For any vector* $h \in U_i$*, the norm of its projection onto* U_j *is bounded by:*

$$\|P_{U_j}(h)\|_2 \leq \cos(\theta_1)\|h\|_2 = S(U_i, U_j)\|h\|_2.$$

For a general vector $h = h_{U_i} + h_\perp$ *where* $h_{U_i} \in U_i$ *and* $h_\perp \perp U_i$*, the bound holds approximately:*

$$\|P_{U_j}(h)\|_2 \lesssim S(U_i, U_j)\|h_{U_i}\|_2 + \delta,$$

where $\delta = \|P_{U_j}(h_\perp)\|_2$ *is a small term under the assumption that* h *is predominantly in* U_i*.*

Our proof is built upon the following three assumptions:

Assumption 1 (Bounded Parameters)**.** *The operator norms of the parameter matrices are bounded, i.e.,* $\exists C_B, C_A > 0$ *such that* $\|B_k\|_{op} \leq C_B$ *and* $\|A_k\|_{op} \leq C_A$ *for all tasks* k*.*

Assumption 2 (Optimization Efficiency)**.** *The task-specific residual* $r_k = y_k - Wh$ *is non-zero and correlates with the input, i.e.,* $\mathbb{E}[\|r_k\|_2] \geq K\mathbb{E}[\|h\|_2]$ *for some* $K > 0$ *and* $h \sim \mathcal{D}_k$*.*

Assumption 3 (Distinct Subspace)**.** *The regularizer* $\mathcal{R}$ *ensures the similarity between any two distinct task subspaces is bounded above by a small constant* $\delta > 0$*, i.e.,* $S(U_i, U_j) \leq \delta$ *for all* $i \neq j$*.*

Proof of Theorem 1. For task i, the learned matrix $\boldsymbol{B}_{\mathcal{T}_i}\boldsymbol{A}_{\mathcal{T}_i}$ is a low-rank approximation of the residual mapping $r_i = y_i - \boldsymbol{W}\boldsymbol{h}_i$. By Lemma 1 (Eckart-Young), this approximation is optimal. Therefore, for an input $\boldsymbol{h}_i$, we have:

$$\boldsymbol{B}_{\mathcal{T}_i}\boldsymbol{A}_{\mathcal{T}_i}\boldsymbol{h}_i \approx r_i.$$

This implies that the output norm is approximately the norm of the residual:

$$\|\boldsymbol{B}_{\mathcal{T}_i}\boldsymbol{A}_{\mathcal{T}_i}\boldsymbol{h}_i\|_2 \approx \|r_i\|_2. \tag{11}$$

From Assumption 2, $\|r_i\|_2$ is significant relative to $\|\boldsymbol{h}_i\|_2$.

For a different task $j \neq i$, we analyze $\|\boldsymbol{B}_{\mathcal{T}_j}\boldsymbol{A}_{\mathcal{T}_j}\boldsymbol{h}_i\|_2$. First, we bound this using the operator norms:

$$\begin{aligned}\|\boldsymbol{B}_{\mathcal{T}_j}\boldsymbol{A}_{\mathcal{T}_j}\boldsymbol{h}_i\|_2 &= \|\boldsymbol{B}_{\mathcal{T}_j}(\boldsymbol{A}_{\mathcal{T}_j}\boldsymbol{h}_i)\|_2 \\ &\leq \|\boldsymbol{B}_{\mathcal{T}_j}\|_{\text{op}} \cdot \|\boldsymbol{A}_{\mathcal{T}_j}\boldsymbol{h}_i\|_2 \leq C_B \cdot \|\boldsymbol{A}_{\mathcal{T}_j}\boldsymbol{h}_i\|_2.\end{aligned} \tag{12}$$

The term $\|\boldsymbol{A}_{\mathcal{T}_j}\boldsymbol{h}_i\|_2$ measures the projection of $\boldsymbol{h}_i$ onto the row space of $\boldsymbol{A}_{\mathcal{T}_j}$, i.e., the task subspace U_j. Applying Lemma 2, and noting that $\boldsymbol{h}_i$ is predominantly in its own task subspace U_i, we get:

$$\begin{aligned}\|\boldsymbol{A}_{\mathcal{T}_j}\boldsymbol{h}_i\|_2 &\lesssim \|\boldsymbol{A}_{\mathcal{T}_j}\|_{\text{op}} \cdot S(U_i, U_j) \cdot \|\boldsymbol{h}_i\|_2. \\ &\leq C_A \cdot \delta \cdot \|\boldsymbol{h}_i\|_2.\end{aligned} \tag{13}$$

Substituting Eqn. (13) into Eqn. (12) yields the final bound for the cross-task output:

$$\|\boldsymbol{B}_{\mathcal{T}_j}\boldsymbol{A}_{\mathcal{T}_j}\boldsymbol{h}_i\|_2 \leq C \cdot \delta \cdot \|\boldsymbol{h}_i\|_2, \tag{14}$$

where $C = C_B \cdot C_A$.

We now compare the task-specific output (Eqn. (11)) and the cross-task output (Eqn. (14)):

$$\|\boldsymbol{B}_{\mathcal{T}_i}\boldsymbol{A}_{\mathcal{T}_i}\boldsymbol{h}_i\|_2 - \|\boldsymbol{B}_{\mathcal{T}_j}\boldsymbol{A}_{\mathcal{T}_j}\boldsymbol{h}_i\|_2 \gtrsim \|r_i\|_2 - C\delta\|\boldsymbol{h}_i\|_2.$$

From Assumption 2 ($\|r_i\|_2 \geq K\|\boldsymbol{h}_i\|_2$), it follows that:

$$\|\boldsymbol{B}_{\mathcal{T}_i}\boldsymbol{A}_{\mathcal{T}_i}\boldsymbol{h}_i\|_2 - \|\boldsymbol{B}_{\mathcal{T}_j}\boldsymbol{A}_{\mathcal{T}_j}\boldsymbol{h}_i\|_2 \gtrsim (K - C\delta)\|\boldsymbol{h}_i\|_2.$$

From Assumption 3, the similarity δ is small. By ensuring that $\delta < K/C$, the term $(K - C\delta)$ is positive. Therefore,

$$\|\boldsymbol{B}_{\mathcal{T}_i}\boldsymbol{A}_{\mathcal{T}_i}\boldsymbol{h}_i\|_2 > \|\boldsymbol{B}_{\mathcal{T}_j}\boldsymbol{A}_{\mathcal{T}_j}\boldsymbol{h}_i\|_2 \quad \text{for all} \quad j \neq i,$$

which completes the proof. □

A.2 Empirical Analysis

In this subsection, we validate Theorem 1 through empirical investigation. Specifically, we estimate the theoretical expectation by means of sampling. We assert that for $\mathbb{E}_{\boldsymbol{h} \sim P_{\mathcal{T}_i}}(\mathcal{R}_{\mathcal{T}_i}(\boldsymbol{h})) >$

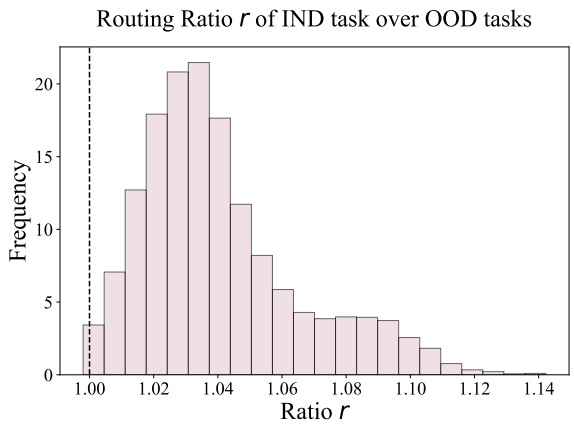

Figure 5: The frequency distributions of route ratio calculated in Eqn. (15).

$\mathbb{E}_{\boldsymbol{h}\sim P_{\mathcal{T}_i}}\left(\mathcal{R}_{\mathcal{T}_j}(\boldsymbol{h})\right), i \neq j$. To verify this, we randomly sample 1000 instances from multiple tasks, and for each instance, we select 5 pivot tokens. For each token, we compute the logit $\|\boldsymbol{B}_{\mathcal{T}_i}\boldsymbol{A}_{\mathcal{T}_i}\boldsymbol{h}\|_2$ in the sampled task $\mathcal{T}i$, as well as $\left\|\boldsymbol{B}_{\mathcal{T}_j}\boldsymbol{A}_{\mathcal{T}_j}\boldsymbol{h}\right\|_2$ in an additional randomly sampled task $\mathcal{T}_j$. The routing ratio r is then averaged across each layer of the LLM as follows:

$$r = \sum_{l}^{L} \frac{1}{L} \frac{\left\|\boldsymbol{B}_{\mathcal{T}_i}^{(l)}\boldsymbol{A}_{\mathcal{T}_i}^{(l)}\boldsymbol{h}^{(l)}\right\|_2}{\left\|\boldsymbol{B}_{\mathcal{T}_j}^{(l)}\boldsymbol{A}_{\mathcal{T}_j}^{(l)}\boldsymbol{h}^{(l)}\right\|_2}. \tag{15}$$

The distributions of routing ratio r are visualized in Figure 5. As visualized in Figure 4, most ratio r is larger than 1.0, which empirically demonstrates that our routing logit can efficiently estimate the intervention corresponding to the most related task.

A.3 Additional Experiment Details

Other Experiment Setting. The maximum duration required for training all upstream task interventions was 48 GPU hours, with each testing task completed within 1 GPU hour.

Routing Strategy Baseline Description. Due to the page limitations, we extensively illustrate the evaluated routing strategies in Figure 4, including: 1) μ-routing, which assigns uniform weights to all experts; 2) TP-routing, which learns a classifier based on categorical cross-entropy loss that depends on the entire input prompt rather than individual token representations; 3) CM-routing, which preserves a prototype for each expert by averaging the hidden representations from the LLM forward pass on each upstream task and applies softmax aggregation based on the cosine similarity between these expert prototypes and the token representations; and 4) Arrow, which ignores the effect of singular values and utilizes only the first right singular vector.

Hyperparameter Analysis. Figure 6 The figure illustrates the impact of varying the hyperparameters K_θ and k_θ on model performance, specifically measured by Accuracy (%) on WG dataset, and on model efficiency, specifically measured by HBM usage (in GB) and TTFT (in seconds). The hyperparameter analysis of K_θ demonstrates that the asynchronous orchestrator effectively filters out irrelevant expert parameters by adjusting K_θ, which significantly reduces HBM usage without causing performance degradation. This indicates that only

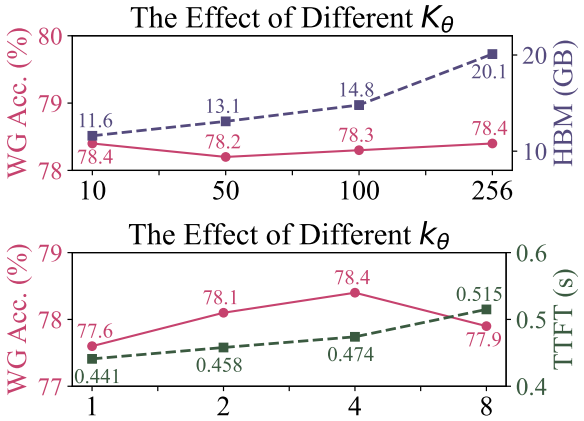

Figure 6: The hyperparameter effects of different K_θ for the asynchronous orchestrator and k_θ for the modular interventions.

a limited number of key experts are necessary for prediction per sample. Meanwhile, the results for k_θ reveal that a modular parameter fine-tuning approach enhances the CTG performance of LLMs. The optimal performance is achieved at $k_\theta = 4$, suggesting that appropriate integration of expert parameters across different tasks improves performance on unseen tasks without introducing significant TTFT latency.

B LLM Usage

The core intellectual contributions of this research, including the conceptualization of ideas, the development of the research methodology, and the execution of all experimental work, were solely conducted by the authors. The LLM's role was strictly limited to that of a writing aid and did not extend to any aspect of the research's conception or analytical processes.

Algorithm 1 The algorithm procedure of RaMod for LLM cross-task generalization

Input: Upstream tasks $\mathbb{T} = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_N\}$ available for LLM fine-tuning, each task $\mathcal{T}_i$ is associated with a set of samples $\mathcal{D}_i = \{(x_1, y_1), ..., (x_n, y_n)\}$, an unseen task $\mathcal{T}'$ and an LLM with $\mathbb{L}$ layers.

Output: Optimized expert interventions $\{\boldsymbol{A}_{\mathcal{T}_i}^{(l)}, \boldsymbol{B}_{\mathcal{T}_i}^{(l)}\}_{\mathcal{T}_i \in \mathbb{T}}^{l \in \mathbb{L}}$ for each upstream task, and the predictions for the unseen task $\mathcal{T}'$

▷ Stage 1: training expert interventions for each upstream task

for task $\mathcal{T}_i \in \mathbb{T}$ **do**

  ▷ Obtain the training samples $\mathcal{D}_i = \{(x_1, y_1), ..., (x_n, y_n)\}$ from task $\mathcal{T}_i$

  ▷ Add the trainable intervention parameters $\{\boldsymbol{A}_{\mathcal{T}_i}^{(l)}, \boldsymbol{B}_{\mathcal{T}_i}^{(l)}\}^{l \in \mathbb{L}}$ for the LLM

  ▷ Employ one of two pivot token selection strategies to fine-tune the token representation:

    (1) RaMod-Fix: a fixed number of prefix and suffix tokens;

    (2) RaMod-Pivot: the tokens with the top-k_{AG} aggregated attention values derived by Eqn. (1).

  ▷ Fine-tune the pivot token representations through DiReFT transformation with the language modeling objective.

  ▷ Save the optimized expert intervention $\{\boldsymbol{A}_{\mathcal{T}_i}^{(l)}, \boldsymbol{B}_{\mathcal{T}_i}^{(l)}\}^{l \in \mathbb{L}}$ for task $\mathcal{T}_i$

end for

▷ Stage 2: Efficient Cross-Task Generalization

for task $\mathcal{T}_i \in \mathbb{T}$ **do**

  ▷ Save the first singular vectors $\boldsymbol{u}_{1,\mathcal{T}_i}$ and $\boldsymbol{v}_{1,\mathcal{T}_i}$ from the singular value decomposition of $(\boldsymbol{B}_{\mathcal{T}_i}\boldsymbol{A}_{\mathcal{T}_i})$

end for

for unseen sample $s' \in \mathcal{T}'$ **do**

  for LLM layer $l = 1$ to $|\mathbb{L}|$ **do**

    ▷ Select the pivot token representations $\{\boldsymbol{h}_*^{(l)}\}$ via either RaMod-Fix or RaMod-Pivot strategy

    ▷ Calculate the FluidRoute routing logits from Eqn. (6) for the interventions filtered by Asynchronous Orchestrator in Sec. 3.3

    ▷ Route each pivot token representation $\boldsymbol{h}_*^{(l)}$ to top-k_θ interventions and use softmax composition to derive target interventions $\{\boldsymbol{A}_*^{(l)}, \boldsymbol{B}_*^{(l)}\}^{l \in \mathbb{L}}$ through Eqn. (7) and (8).

    ▷ Calculating the modified token representations through DiReFT transformation

  end for

end for