Effective LoRA Adapter Routing using Task Representations

Anonymous ACL submission

Abstract

Low-rank adaptation (LoRA) enables parameter-efficient specialization of LLMs, leading to rapidly growing public adapter pools. Effectively leveraging such pools requires routing: identifying and composing the adapters most useful for a query. Existing routing approaches rely on adapter training data, parameter-space geometry, or learned routing modules, which often provide indirect signals of adapter utility or remain unsuitable for dynamic adapter pools. We introduce LoRAuter, a routing framework that organizes adapter pools through *task representations* derived from small validation sets. For each representative task, LoRAuter identifies useful adapters through lightweight empirical evaluation, requiring neither router training nor access to adapter training data. Given a query, LoRAuter retrieves relevant tasks and composes their associated adapters using an input-aware weighted fusion mechanism. Experiments across multiple tasks show that LoRAuter consistently outperforms baseline routing approaches, matching Oracle performance (101.2%) when task-aligned adapters exist and achieving state-of-the-art results on unseen tasks (+5.2 points). We further demonstrate the robustness of LoRAuter to very large, noisy adapter pools by scaling it to over 1500 adapters.

1 Introduction

Modern large language models (LLMs) such as GPT (OpenAI, 2025), Gemini (Gemini Team Google, 2025) and Llama (Touvron et al., 2023a) have driven remarkable progress in natural language processing (NLP) and beyond. However, specializing such massive monolithic models for each downstream task (*e.g.*, question answering or translation) remains computationally expensive and memory-intensive. To address these challenges, *parameter-efficient fine-tuning (PEFT)* methods (Lester et al., 2021; Liu et al., 2022; Hu et al., 2022) have emerged as effective alternatives, enabling adaptation by updating only a small subset of parameters or introducing lightweight, trainable modules. Among them, *low-rank adaptation (LoRA)* (Hu et al., 2022) has gained particular prominence due to its simplicity, modularity, and strong performance. LoRA injects small, low-rank matrices into frozen model weights, allowing efficient fine-tuning while preserving the general knowledge of the base model. Its modular design has led to the emergence of large, public adapter pools, *e.g.*, there currently are over 2300 adapters available just for the META-LLAMA/LLAMA-2-7B-HF model alone on HUGGINGFACE for a wide variety of tasks[1].

Mixed-task LoRA serving. Given these large and diverse adapter pools, a natural goal is to reuse existing LoRA adapters at inference time rather than train a new adapter for every downstream need. This is especially valuable in *mixed-task* serving, where a system faces a stream of heterogeneous requests (*e.g.*, translation, summarization, question answering) arriving without task labels (Zhao et al., 2024), and where a single query may require capabilities spread across several adapters. Effectively leveraging adapter pools therefore requires a routing mechanism: for each incoming query, the system must identify which adapter, or combination of adapters, is likely to be useful. Since different adapters may capture complementary capabilities, routing is naturally coupled with adapter composition, where multiple selected adapters are combined to produce the final response.

Prior routing approaches mainly fall into three families, distinguished by the signal they use to estimate adapter utility. *Spectral* methods such as ARROW (Ostapenko et al., 2024) and

[1] See https://huggingface.co/models?other=base_model:adapter:meta-llama/Llama-2-7b-hf

SPECTR (Fleshman and Durme, 2025) use the adapter parameters themselves, applying spectral decompositions to the low-rank matrices and routing each query toward adapters whose parameter subspaces align with it. *Training-based* methods learn the routing or composition from data: Mixture-of-Experts (MoE)-style approaches such as MOLE (Wu et al., 2024) train gating networks, while LORAHUB (Huang et al., 2024) learns coefficients to merge adapters. *Similarity-based* methods avoid training a router and depend on adapter training data: ADAPTERSOUP (Chronopoulou et al., 2023) and LORARETRIEVER (Zhao et al., 2024) derive semantic embeddings for each adapter from its training examples and route queries by matching them against these embeddings.

Challenges. Although these approaches provide useful mechanisms for selecting or composing adapters, they are not fully suited to handle large, heterogeneous public adapter pools. First, existing methods use routing signals that are indirect proxies for adapter utility. Training-data similarity and parameter-space geometry can suggest that an adapter is *related* to a query, but they do not directly measure whether it *performs* well on the target task. Second, training-based methods operate over fixed rigid pools, making them unsuitable for dynamic scenarios with ever-growing pools (Zhao et al., 2024). Furthermore, learning composition weights over large adapter pools remains challenging due to the combinatorial search space and suffers from high variance (Huang et al., 2024). Third, the dependency on training data restricts the pool of available adapters. For publicly available LoRA adapters, such as the ones on HUGGINGFACE, the original training datasets are often unavailable, poorly documented, or inaccessible.

We address the above challenges through two key insights. ***(i) Task representations are the interface to the wild.*** Existing approaches are largely adapter-centric: they reason about what training data adapters use, or how their parameters are structured. In contrast, we argue that routing should instead be grounded in the tasks a deployment system aims to support, independently of how individual public adapters were produced. Our key observation is that, although a deployer may not have full information about public adapters, they usually know the tasks their system needs to serve. For each task, the deployer can often obtain a small representative validation set drawn from public benchmarks (Wei et al., 2022; Wang et al., 2022), user-provided examples, production logs, or synthetic task suites (Wang et al., 2023). These examples define task representations that characterize the desired behavior of the serving system independently of any adapter's training information. Notably, this does not require explicit task labels: when only unlabeled queries are available, the tasks themselves can be discovered by clustering query representations, an approach we find to be effective (Section 4.4). ***(ii) Adapter evaluation is the most reliable routing signal.*** What routing ultimately requires is a functional signal: whether a given adapter actually works for the task. Therefore, the most reliable signal for routing is not training-data similarity or parameter-space structure, but empirical performance on representative task evaluations. We thus use the validation examples as a lightweight evaluation substrate for directly measuring adapter utility.

Our framework. This work introduces LORAUTER, which organizes an unstructured adapter pool into a task-indexed catalog, retrieves the most relevant tasks for a query, and composes the corresponding adapters in an input-aware manner. LORAUTER first constructs a database of representative tasks using small validation sets. For each task, it identifies the most suitable adapter from the pool through cheap performance evaluations on the corresponding validation sets. It therefore requires neither router training nor access to adapter training data. At inference time, incoming queries are embedded and matched against task representations, allowing LORAUTER to retrieve the top-K relevant tasks and their associated best adapters. These adapters are then composed using a weighted output-space fusion mechanism that reflects input–task similarity, enabling effective generalization to both in-domain and out-of-domain queries.

Strong performance and scalability. We implement LORAUTER and evaluate our framework on a mixed-task benchmark comprising diverse tasks, multiple base models, and 48 adapters (Zhao et al., 2024). In in-domain settings, LORAUTER achieves an average normalized performance of 101.2% relative to the Oracle task-aligned adapter *i.e.*, adapter trained on that task. This demonstrates that LORAUTER is effective in identifying and composing adapters which are collectively better than the parent adapter for a given task. In out-of-domain scenarios, where the model must generalize to unseen tasks, LORAUTER outperforms the strongest baseline, LORARETRIEVER (Zhao et al., 2024), by

5.2 percentage points. We then scale LoRAuter to a pool of 1500+ adapters fetched *from the wild*, and show that our framework achieves competitive performance to the original 48 adapters on the same mixed-task benchmark when excluding these 48 *well curated adapters* from the pool. Thus, LoRAuter can extract useful routing signals from large, noisy adapter collections, offering a scalable and robust solution for open-ended LoRA serving. To the best of our knowledge, LoRAuter is the first system to support routing at this scale.

2 Background and problem description

2.1 Low-rank adaptation (LoRA)

Low-rank adaptation (LoRA) (Hu et al., 2022) is a parameter-efficient fine-tuning (PEFT) method that adapts a frozen model by injecting trainable low-rank updates into selected weight matrices. For a weight matrix $W \in \mathbb{R}^{c\times d}$, LoRA learns two low-rank matrices $A \in \mathbb{R}^{r\times d}$ and $B \in \mathbb{R}^{c\times r}$, where $r \ll \min(c, d)$, such that the parameter update is $\Delta W = BA$. The resulting forward computation becomes $h' = (W + \Delta W)x = Wx + BAx$. This parameterization enables efficient adaptation with only a small number of trainable parameters.

2.2 Composing LoRA adapters

Each trained LoRA adapter (B_i, A_i) captures task-specific adaptations of the base model. Combining multiple adapters can therefore integrate complementary skills and knowledge across tasks. Early fusion strategies, *e.g.*, parameter-space interpolation (Zhang et al., 2023), merge adapters linearly:

$$h' = Wx + (\lambda \Delta W_1 + (1 - \lambda)\Delta W_2)x, \quad (1)$$

where $\lambda \in [0, 1]$ controls the relative contribution of each adapter. However, this approach requires manual tuning of λ for each task. More advanced approaches, such as LoRAHub (Huang et al., 2024), extend this idea by learning adaptive fusion weights rather than relying on manually tuned coefficients. Given a set of LoRA adapters, the fused representation is computed as:

$$h' = Wx + \left(\sum_i w_i B_i\right)\left(\sum_i w_i A_i\right)x, \quad (2)$$

where w_i are learnable weights optimized in a few-shot manner. However, learning stable fusion weights over large adapter pools remains challenging due to the combinatorial search space, resulting in high performance variance (Huang et al., 2024).

2.3 Problem formulation

We now formalize the problem of LoRA adapter routing. We consider a pool of N LoRA adapters, $\Phi = \{\phi_1, \phi_2, \ldots, \phi_N\}$, available for routing, where each adapter $\phi_i = (B_i, A_i)$. In practice, an adapter is not made of a single pair of matrices, but of a pair of matrices for every layer of the model that benefits from the adapter (Hu et al., 2022). To simplify notation, we consider these adapter layers to be implicit.

Let $\mathcal{T}^{tr} = \{t_1^{\text{tr}}, t_2^{tr}, \ldots, t_M^{tr}\}$ denote the collective training task set. Each adapter ϕ_i is associated with a task $t_j^{tr} \in \mathcal{T}^{tr}$. We allow $M \leq N$, since a single task may have multiple adapters trained for it (*e.g.*, with different ranks or training procedures)[2]. In the case when $M = N$, tasks and adapters have one-to-one correspondence. We assume that no data from the adapters' original training sets is available for routing. At test time, user queries span a wide variety of underlying tasks and arrive without explicit task labels. In particular, a query x can be both in-domain (non-OOD) *i.e.*, $\texttt{task}(x) \in \mathcal{T}^{tr}$ or out-of-domain (OOD) *i.e.*, $\texttt{task}(x) \notin \mathcal{T}^{tr}$ where $\texttt{task}(.)$ indicates the ground-truth task label of the query.

The goal of routing is to efficiently *select* and *compose* adapters from the pool to respond to a given user query, generalizing to unseen tasks or domains. Under this scenario, given a query x without its task label, the serving process can be expressed as: $y = F(\mathcal{R}_x, x, W)$ where $\mathcal{R}_x$ corresponds to the output of the retrieval mechanism, F is the composition function and W refers to the weights of base LLM. For instance, $\mathcal{R}_x$ can correspond to the top-K LoRA adapters retrieved for answering x and F can correspond to one of the aforementioned fusion strategies.

3 Design of LoRAuter

We now present the design of LoRAuter. The central idea of LoRAuter is to perform LoRA adapter selection at the level of tasks and derive routing signals from adapter evaluation. To this end, LoRAuter is organized into four core components: *(1)* a database of relevant tasks, *(2)* task-to-adapter pairing, *(3)* query-based task retrieval, and *(4)* adapter composition. Figure 1 illustrates the full workflow of LoRAuter.

[2] https://huggingface.co/tasks provides examples where multiple models are trained for the same task.

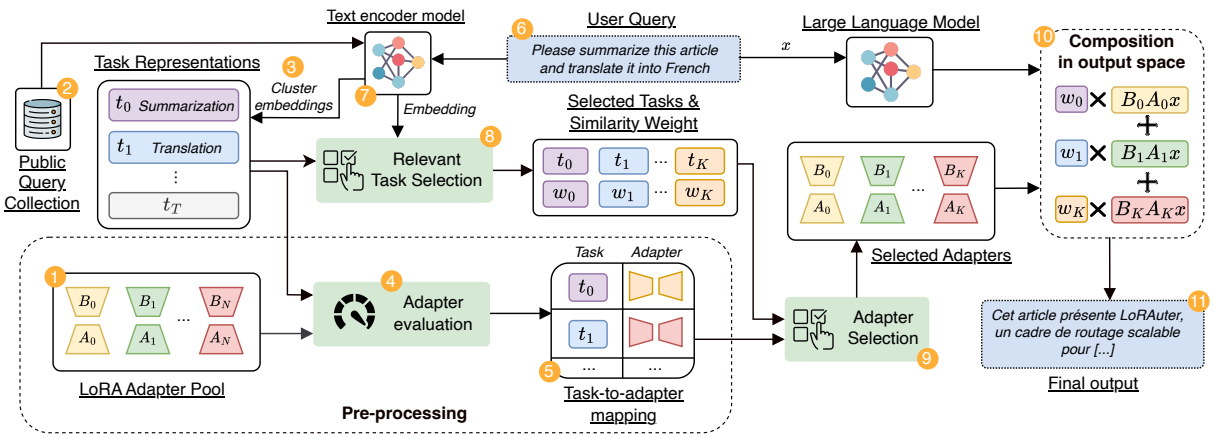

Figure 1: The LoRAuter framework.

3.1 Database of relevant tasks (❷)

LoRAuter begins by constructing a representative task database $\mathcal{T}^{rep} = \{t_1, t_2, \ldots, t_T\}$ from publicly available task queries. This task database serves as a repository of T diverse tasks representing the range of query types the system may encounter. Each task t_i is accompanied by a small validation set D_i containing labeled samples, which are lightweight and readily available in practice today. Note that T can be typically much lower than N, the total number of adapters in the pool. Importantly, LoRAuter does not require explicit task labels: as we show in Section 4.4, representative tasks can instead be automatically discovered by clustering query representations.

3.2 Task to adapter pairing (❸, ❹, ❺)

Once the task database has been constructed, the next step is to determine the most suitable LoRA adapter for each task. To build the mapping $\mathcal{M} : \mathcal{T}^{rep} \to \Phi$, we evaluate each adapter on the task's validation set and assign the highest-scoring adapter according to the task-specific evaluation metric (*e.g.*, ROUGE or BLEU). This procedure also allows flexible system updates, where new adapters or tasks can be added or removed, and the target evaluation metric for any task can be modified with minimal overhead.

Since adapters typically specialize to only a subset of tasks, LoRAuter can efficiently identify strong candidates without exhaustively evaluating the entire adapter pool. To further improve scalability, we employ tournament-based search strategies such as *Successive Halving* (Jamieson and Talwalkar, 2016) (SH), which progressively allocate more evaluations to promising adapters while quickly discarding weaker candidates. This enables efficient discovery of high-performing task–adapter pairings with substantially lower evaluation cost. Moreover, SH naturally scales to hundreds of adapters through parallel evaluation across GPUs with minimal coordination overhead.

3.3 Query-based task retrieval (❻, ❼, ❽, ❾)

LoRAuter identifies the most relevant task for a given query using task representations. These representations are constructed in a one-time offline manner by employing a sentence-embedding model E. Specifically, the model E encodes a subset of the tasks' validation queries to obtain the task representation as follows.

For each task t_i, we construct a fixed embedding by encoding and aggregating a small subset of its validation queries. Let $\{v_j\}_{j=1}^{m} \subset D_i$ be a randomly sampled subset with $m \ll |D_i|$. Given an embedding instruction I and a validation query v, the encoder E produces an embedding by encoding the concatenation $I \oplus v$. The task representation $\mathbf{e}_i$ is then computed as $\mathbf{e}_i = \frac{1}{m}\sum_{j=1}^{m} E(I \oplus v_j)$. Similar to (Zhao et al., 2024), we set I to the following instruction: *"Represent the sentence for similar task retrieval"*, allowing the embedding to capture task similarity.

At inference time, we similarly encode an arriving query x to obtain an embedding $\mathbf{e}_x$. This embedding is then compared against the precomputed task embeddings $\{\mathbf{e}_i\}_{i=1}^{T}$ using the *cosine similarity* metric: $s_i = \frac{\mathbf{e}_x \cdot \mathbf{e}_i}{\|\mathbf{e}_x\| \|\mathbf{e}_i\|}$, where s_i denotes the similarity between the input and the task t_i. We then retrieve the set of top-K most similar tasks based on their scores s_i: $\mathcal{T}_x = \{t_i \mid s_i \text{ is in top-}K\}$. To obtain a probabilistic interpretation of the input-task affinities, we normalize these scores using the

softmax function with temperature parameter τ, which naturally aligns with the temperature-scaled contrastive objectives used to train the sentence encoders. The resulting probability p_i reflects the likelihood that the input belongs to task t_i.

3.4 Adapter composition (⑩)

After retrieving the top-K tasks $\mathcal{T}_x$ for a given query x along with corresponding probabilities $\{p_i\}_{i=1}^{K}$, we use the precomputed task-to-adapter mapping $\mathcal{M}$ to identify the adapter $\phi \in \Phi$ associated with each task t_i. We define $\mathcal{R}_x = \{(p_i, \mathcal{M}[t_i]) \mid t_i \in \mathcal{T}_x\}$ as the output of selection step, containing relevant adapters coupled with their matching likelihoods. The goal is now to compose these K adapters to produce the final output, leveraging their complementary strengths.

While some prior methods (Zhao et al., 2024; Chronopoulou et al., 2023) treat all retrieved tasks as equally relevant by using uniform averaging for composition, we use an input-aware fusion mechanism that dynamically adjusts the contribution of each adapter based on the input-task similarity. Specifically, the fused adapter outputs h' are obtained through a weighted combination of the top-K adapters in the *output space*:

$$h' = Wx + \sum_{(w_i,\phi_i)\in\mathcal{R}_x} w_i \, B_i A_i x, \qquad (3)$$

where $\sum_{i=1}^{K} w_i = 1$ since the weights w_i correspond to probabilities p_i. This formulation adaptively integrates information from task-relevant adapters, allowing the model to emphasize those most aligned with the input while minimizing interference from unrelated tasks. Furthermore, composition in the output space yields better performance than composition in the parameter space (Zhao et al., 2024), as LoRA adapters often exhibit limited alignment, which hinders effective parameter merging (Stoica et al., 2025).

4 Evaluation

We evaluate LoRAuter against representative benchmarks and several state-of-the-art (SOTA) baselines. We first present results under non-OOD, OOD, and semi-OOD settings, and study robustness when scaling to large adapter pools (§4.2). We then conduct ablation studies on retrieval and composition (§4.3), evaluate routing with clustering-based pseudo-tasks (§4.4), and measure the efficiency of adapter selection using SH (§4.5).

4.1 Experimental setup

Dataset. We adopt a curated subset of 48 individual tasks from FLANV2 (Wei et al., 2022), covering a broad range of Natural Language Understanding (NLU) and Natural Language Generation (NLG) settings. We note that the same set of tasks is used in Zhao et al. (2024), enabling a direct comparison between LoRAuter and LoRARetriever.

Adapters. We use Llama 2-7B, Llama 2-13B (Touvron et al., 2023b) and Llama 3.1-8B (Grattafiori et al., 2024) as the base model, following prior work and leveraging its large ecosystem of publicly available LoRA adapters. Each task from the above dataset is treated independently and used to train a dedicated LoRA adapter, resulting in a pool of 48 task-specific adapters. Each LoRA adapter is trained using the Alpaca instruction-following format (Taori et al., 2023). The rank r and scaling hyperparameter α are fixed to 6 and 12, respectively, matching the configuration in the LoRARetriever baseline. Following standard LoRA practice, the learned low-rank weight update is scaled by α/r. To further ensure comparability, we also adopt the same sentence encoder used in Zhao et al. (2024) for generating task embeddings and input representations. We include a sensitivity analysis to the choice of encoder in Section 4.3. Lastly, we also assess LoRAuter on a public adapter pool fetched from HuggingFace, comprising 1500+ adapters.

Representative tasks. In the non-OOD routing setting, $\mathcal{T}^{rep} = \mathcal{T}^{tr}$ as expected. In the OOD routing setting, we set $\mathcal{T}^{rep} = \mathcal{T}^{tr} \backslash \{\texttt{task}(x)\}$, *i.e.*, the adapter and validation data corresponding to the ground-truth task of query x both remain unavailable during routing. We further introduce a *semi-OOD* setting that lies between the non-OOD and OOD regimes and reflects realistic deployment scenarios. In this setting, the adapter trained on the target task is unavailable, but a validation set from the target task remains accessible for routing (see Figure 6, Appendix B). Across all settings, LoRAuter uses only validation data from tasks in $\mathcal{T}^{rep}$ and never accesses training data, with at most 200 validation samples per task. Despite this limited validation set, we show that it is sufficient to reliably identify the best-performing adapter from the pool. For the final evaluation, we use the same test set as Zhao et al. (2024), consisting of 50 held-out test samples for each of the 48 tasks.

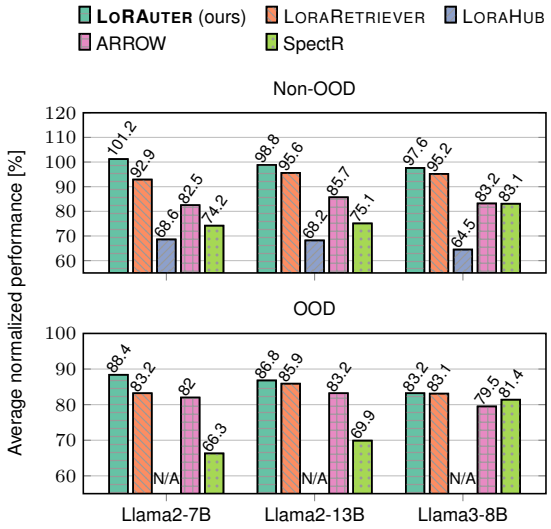

Figure 2: Performance for different routing methods on non-OOD and OOD tasks across all three models.

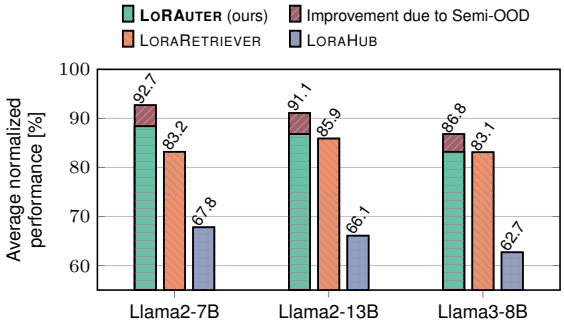

Figure 3: Results in semi-OOD setting where validation data is available but the task-specific adapter is removed.

Baselines. We compare LoRAuter against LoRAHub (Huang et al., 2024), which learns global adapter fusion weights; LoRARetriever (Zhao et al., 2024), which retrieves using training-data similarity and uniformly fuses adapters; ARROW (Ostapenko et al., 2024) and SpectR (Fleshman and Durme, 2025), two spectral routing methods based on adapter parameter decompositions. As an oracle, we report the performance achieved when directly using the adapter trained on the same task. Note that the oracle is fixed, and remains independent of the OOD or non-OOD setting.

Metrics. Following Wei et al. (2022) and Zhao et al. (2024), task performance is measured using task-specific metrics: Exact Match (EM) accuracy for NLU tasks, BLEU for translation tasks, and ROUGE-1/2/L for structure-to-text generation tasks. To aggregate performance across tasks with different metrics, we normalize each score by dividing it by the oracle performance for the corresponding task. We then average the normalized scores across tasks and report the resulting metric.

4.2 Main results

OOD and non-OOD settings. Figure 2 presents results across all models, with detailed numerical values reported in Table 8 (Appendix C.5). LoRAuter achieves the best performance in every configuration. Under the non-OOD setting with Llama2-7B, it attains an average normalized score of 101.2%, slightly surpassing the oracle (100%). This suggests that evaluation-driven routing and task-affinity-based composition of complementary adapters can sometimes outperform the adapter trained for the target task itself. Under the OOD setting, LoRAuter reaches 88.4%, outperforming the best baseline, LoRARetriever, by 5.2 points. The same trend holds for Llama2-13B and Llama3.1-8B, where LoRAuter achieves 98.8% and 97.6% respectively in the non-OOD setting, compared to 95.6% and 95.2% for LoRARetriever. Spectral routing methods (ARROW and SpectR) achieve relatively lower performance as they directly route based on parameter values, which may not carry sufficient routing signal. Lastly, LoRAHub faces challenges in learning good composition weights as discussed in Section 2.2 and is not applicable in OOD settings due to absence of validation data.

Semi-OOD setting. Figure 3 presents the results. With access to task validation data, LoRAuter can more accurately identify relevant adapters through task representations. The semi-OOD regime also allows evaluation of LoRAHub, while for LoRARetriever we continue to report its OOD performance since it uses only training data. Compared to the OOD setting, LoRAuter improves from 88.4% to 92.7% on Llama2-7B, from 86.8% to 91.1% on Llama2-13B, and from 83.2% to 86.8% on Llama3.1-8B. These gains consistently exceed 3–4 points and remain substantially higher than the two baselines.

Scaling the adapter pool. We study the robustness of LoRAuter on large heterogeneous adapter pools collected "*from the wild*". We collect all Llama2-7B adapters on HuggingFace with rank ≤ 64, resulting in a pool of 1567 adapters (HF). We evaluate LoRAuter with and without the 48 well-curated adapters (HF+48 and 48). Results are shown in Table 1, with per-task values in Table 10 (Appendix C.5). Despite having no access to

Table 1: Llama2-7B results across three pools: 48 well-curated adapters (48), adapters *from the wild* on HUGGINGFACE (HF), and the combined pool (HF+48).

Setting	Non-OOD	Semi-OOD	OOD
48	101.2	92.7	88.4
HF	N/A	89.6	85.7
HF + 48	100.2	91.4	89.6

Table 3: Embedding model sensitivity. Under Llama 3.1-8B, LORAUTER (LA) outperforms LORARETRIEVER (LR) even with the off-the-shelf `e5-large-v2` encoder.

Setup	**Off-the-shelf**		**Fine-tuned**	
	LA	**LR**	**LA**	**LR**
Non-OOD	**96.0**	94.3	**97.6**	95.2
Semi-OOD	**85.4**	82.7	**86.8**	83.1
OOD	**82.8**	82.7	**83.2**	83.1

adapter training data, LORAUTER achieves strong performance on the noisy HF pool alone, reaching 85.7% and 89.6% in the OOD and semi-OOD settings, compared to 88.4% and 92.7% using only the curated 48 adapters. Interestingly, combining the HF and curated pools further improves OOD performance from 88.4% to 89.6%, suggesting that LORAUTER can effectively identify and exploit useful adapters even within large noisy collections. To the best of our knowledge, this is the first work to evaluate LoRA routing on such large public adapter pools. Baseline methods cannot be meaningfully evaluated in this regime, as they either require adapter training data or become computationally infeasible at this scale.

4.3 Ablation studies

We conduct two ablation studies to assess: *(i)* the effects of different retrieval and composition strategies, and *(ii)* the sensitivity of LORAUTER (LA) to the embedding model. We recall that LORARETRIEVER (LR) uses train-data-based retrieval with uniform composition, while LA retrieves using task representations with task-affinity-aware composition. As shown in Table 2, both components consistently improve performance across multiple settings. For example, on Llama2-7B (non-OOD), replacing uniform composition with task-affinity-aware composition improves performance from 92.9% to 98.6%, while replacing train-data retrieval with task-representation retrieval improves it to 96.7%. Using both together achieves the best overall performance of 101.2%. A similar trend is observed for the OOD setup with Llama2-13B. We further evaluate sensitivity to the embedding model by replacing the fine-tuned `e5-large-v2` encoder used by LORARETRIEVER with the original off-the-shelf version. As shown in Table 3 for Llama 3.1-8B, LA consistently outperforms LR across all settings even without embedding-model fine-tuning. In the non-OOD setting, LA achieves 96.0% with the off-the-shelf encoder, outperforming not only off-the-shelf LR (94.3%) but also fine-tuned LR (95.2%). These results suggest that LA is substantially less sensitive to embedding-model fine-tuning while maintaining strong routing performance. Additional analysis of adapter composition is deferred to Appendix C.3.

Table 2: Retrieval and composition components of LORAUTER (task-rep + task affinity) individually bring performance improvements over the baseline LORARETRIEVER (train-data + uniform weights).

Model	**Retrieval based on**	**Composition using**	**Norm. Performance**	
			non-OOD	**OOD**
Llama2-7B	train-data	uniform weights	92.9	83.2
	train-data	task affinity	98.6	85.7
	task-rep	uniform weights	96.7	**89.9**
	task-rep	task affinity	**101.2**	88.4
Llama2-13B	train-data	uniform weights	95.6	85.9
	train-data	task affinity	**99.1**	86.2
	task-rep	uniform weights	96.8	86.3
	task-rep	task affinity	98.8	**86.8**

4.4 What if task labels are unavailable?

We study how LORAUTER behaves when ground-truth task labels are unavailable for constructing representative tasks. Instead of using the original 48 task labels, we cluster validation representations into K pseudo tasks using K-Means while keeping the total number of validation samples fixed. Routing must therefore rely purely on the structure of the data without access to true task annotations.

Figure 4 shows that the normalized performance remains within a stable range and peaks at an intermediate number of clusters. The optimal choice of K varies across regimes. For OOD inputs, fewer (larger) clusters perform better, encouraging routing toward robust adapters that generalize across broader data distributions. In contrast, non-OOD inputs benefit from finer-grained clusters that enable selection of more specialized adapters. Importantly, even without access to ground truth task labels, LORAUTER achieves higher performance than LORARETRIEVER for a significant range of K. At the best configuration, LORAUTER reaches 101.5% normalized performance for non-OOD inputs and 88.0% for OOD inputs.

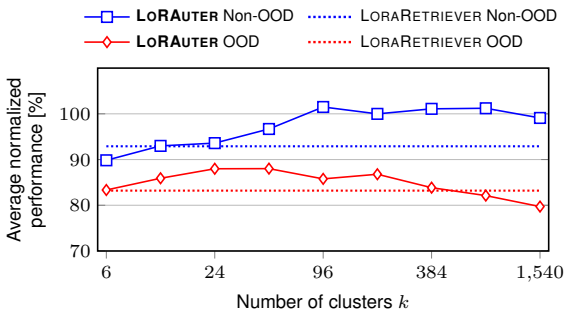

Figure 4: Routing via pseudo task labels for Llama2-7B. LoRAuter beats LoRARetriever across most K.

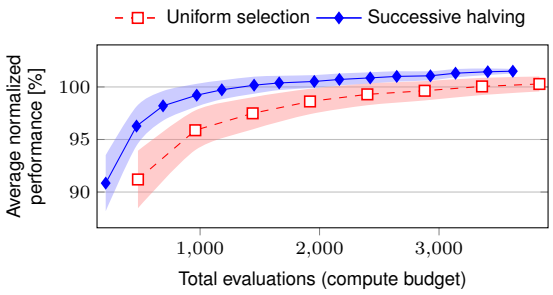

Figure 5: Results for Llama2-7B. Successive halving (SH) efficiently pairs tasks to adapters.

4.5 Efficiently pairing tasks with adapters

We now compare two strategies for identifying the best adapter for a given task under a fixed compute budget (number of inference calls). We consider: *(i) Uniform Selection:* the total evaluation budget is evenly divided across all adapters, and each adapter is evaluated on the same number of validation samples before selecting the highest-scoring one; *(ii) Successive Halving (SH):* adapters are progressively filtered based on performance estimates, allocating more budget to promising candidates while respecting the same total budget. For each budget, we conduct 100 independent runs, where each run randomly samples a subset from the validation datasets. In Figure 5, SH reaches near-peak performance using over 2× less evaluation budget than Uniform Selection. Thus, LoRAuter scales efficiently to larger adapter pools by avoiding evaluation of all adapters on all samples.

5 Related work

Model routing. Model-routing methods dynamically select models at inference time to balance quality and efficiency (Wang et al., 2025b). Prior work includes both non-predictive approaches that rely on intermediate outputs or early exiting (Chen et al., 2024; Aggarwal et al., 2024), and predictive approaches that estimate model utility before inference (Ding et al., 2024; Wang et al., 2025b). More recent systems extend routing to dynamic model pools using learned capability embeddings (Wang et al., 2025a). However, these methods treat models as indivisible units by selecting a single model per query, unlike modern PEFT settings where composing multiple LoRA adapters can be beneficial.

LoRA adapter routing and composition. Existing approaches to LoRA routing primarily differ in how they represent and select adapters. Spectral methods route using parameter-space structure (Ostapenko et al., 2024; Fleshman and Durme, 2025), similarity-based methods derive adapter representations from training data (Chronopoulou et al., 2023; Zhao et al., 2024), and training-based approaches learn routing or fusion mechanisms over adapter pools (Huang et al., 2024; Wu et al., 2024; Muqeeth et al., 2024). While effective, these methods either depend on adapter training data, require learned routing components, or assume fixed adapter pools. In contrast, LoRAuter performs evaluation-driven routing using lightweight task evaluations and naturally supports dynamic pools.

Task representations. Prior work has explored representing tasks as compact embeddings for measuring task similarity and predicting transferability (Achille et al., 2019; Zamir et al., 2018). More recently, task-related signals have been incorporated into routing architectures to guide expert selection (Liang et al., 2025). In contrast, existing LoRA routing methods remain largely adapter-centric, deriving representations from adapter parameters or training data. Our work instead treats tasks as first-class routing primitives independent of how adapters were trained, enabling generalization across heterogeneous public adapter pools.

6 Conclusion

We introduced LoRAuter, a framework for LoRA routing based on task representations and evaluation-driven signals. Unlike prior approaches, LoRAuter requires neither router training nor access to adapter training data, enabling robust routing over large adapter pools. Across extensive benchmarks, LoRAuter consistently outperforms prior routing methods in both in-domain and OOD settings while scaling to pool sizes over 1500 adapters. Overall, LoRAuter provides a scalable foundation for open-ended LoRA routing.

Limitations

One limitation of LoRAuter is that it requires a small validation set for each task in the catalog. While this distinguishes it from fully data-free methods such as Arrow and SpectR, we argue that this requirement is reasonable to satisfy in practice: a deployer typically knows the tasks their system must serve, and representative examples can readily be obtained from public benchmarks, production logs, user-provided queries, or synthetic task suites. Furthermore, LoRAuter does not strictly require task labels, since tasks can alternatively be discovered by clustering query representations (see Section 4.4).

References

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless Fowlkes, Stefano Soatto, and Pietro Perona. 2019. Task2vec: Task embedding for meta-learning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*.

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. 2024. Automix: automatically mixing language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*.

Lingjiao Chen, Matei Zaharia, and James Zou. 2024. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*.

Alexandra Chronopoulou, Matthew Peters, Alexander Fraser, and Jesse Dodge. 2023. AdapterSoup: Weight averaging to improve generalization of pre-trained language models. In *Findings of the Association for Computational Linguistics: EACL 2023*.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. 2024. Hybrid LLM: Cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations*.

William Fleshman and Benjamin Van Durme. 2025. Spectr: Dynamically composing LM experts with spectral routing. In *Second Conference on Language Modeling*.

Gemini Team Google. 2025. Gemini: A family of highly capable multimodal models.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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*.

Kevin Jamieson and Ameet Talwalkar. 2016. Non-stochastic best arm identification and hyperparameter optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Yunlong Liang, Fandong Meng, and Jie Zhou. 2025. THOR-MoE: Hierarchical task-guided and context-responsive routing for neural machine translation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*.

Mohammed Muqeeth, Haokun Liu, Yufan Liu, and Colin Raffel. 2024. Learning to route among specialized experts for zero-shot generalization. In *Proceedings of the 41st International Conference on Machine Learning*.

OpenAI. 2025. Introducing gpt-5. Accessed: 2026-01-21.

Oleksiy Ostapenko, Zhan Su, Edoardo Maria Ponti, Laurent Charlin, Nicolas Le Roux, Lucas Caccia, and Alessandro Sordoni. 2024. Towards modular llms by building and reusing a library of loras. In *Proceedings of the 41st International Conference on Machine Learning*.

George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. 2025. Model merging with SVD to tie the knots. In *The Thirteenth International Conference on Learning Representations*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023b. Llama 2: Open foundation and fine-tuned chat models.

Chenxu Wang, Hao Li, Yiqun Zhang, Linyao Chen, Jianhao Chen, Ping Jian, Peng Ye, Qiaosheng Zhang, and Shuyue Hu. 2025a. Icl-router: In-context learned model representations for llm routing. Accepted by AAAI 2026.

Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. 2025b. MixLLM: Dynamic routing in mixed large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, and 16 others. 2022. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

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

Amir R. Zamir, Alexander Sax, William Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. 2018. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jinghan Zhang, Shiqi Chen, Junteng Liu, and Junxian He. 2023. Composing parameter-efficient modules with arithmetic operations. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*.

Ziyu Zhao, Leilei Gan, Guoyin Wang, Wangchunshu Zhou, Hongxia Yang, Kun Kuang, and Fei Wu. 2024. LoraRetriever: Input-aware LoRA retrieval and composition for mixed tasks in the wild. In *Findings of the Association for Computational Linguistics: ACL 2024*.

A Reproducibility statement

All datasets and LoRA adapters used in our experiments are publicly available. To ensure full reproducibility, we release our implementation together with all hyperparameters, evaluation scripts, and model outputs at https://anonymous.4open.science/r/LoRAuter-5B2C/README.md. AI assistants (such as ChatGPT) were used for copy-editing, proofreading, and limited coding assistance. All scientific content and claims are the authors' own.

B Evaluation details

This section provides additional evaluation details for all methods considered in our study. All baselines are evaluated under an identical experimental setup: each method operates over the same pool of 48 LoRA adapters with fixed rank $r = 6$ and scaling $\alpha = 12$, and is assessed on the same mixed-task evaluation set. All experiments are conducted using both **Llama2-7B** and **Llama2-13B** backbone models, with inference performed using `bfloat16` precision. To verify that our findings generalize beyond the Llama2 family, we additionally run experiments with a **Llama3.1-8B** backbone. For **LoRAuter Fusion**, we fuse the $k = 3$ adapters and use a softmax temperature of $\tau = 0.2$ in all evaluations. The sentence transformer used is https://huggingface.co/Styxxxx/lora_retriever, which is the same as the one found in the LoRARetriever paper. All research was conducted on an internal GPU cluster equipped with A100 GPUs, using roughly 2,000 A100 GPU-hours in total (including preliminary and exploratory experiments).

We evaluate our methods under three different routing regimes, illustrated in Figure 6. In the non-OOD (in-domain) setting, both the validation samples and the LoRA adapter corresponding to the input's ground-truth task are available during routing. In the *Semi-OOD* setting, validation samples for the target task remain available, but the adapter trained explicitly on that task is removed from the adapter pool, requiring methods to generalize by selecting adapters trained on related tasks. Finally, in the OOD (out-of-domain) setting, both the task-specific validation samples and the corresponding adapter are excluded, representing the most challenging scenario in which routing must rely entirely on cross-task generalization.

B.1 Details of baseline methods

For completeness, we evaluate LoRAuter against several representative training-free or lightweight adapter–routing approaches. All baselines are implemented following the algorithmic descriptions in their respective papers, and where applicable, we reuse the same text encoder and LoRA configurations to ensure a fair comparison. The baselines correspond exactly to those summarized in Section 4.1 of this paper.

LoRAHub (Huang et al., 2024). LoRAHub is a task-specific adapter–merging method that learns task-specific fusion weights over a set of LoRA parameters via black-box optimization. The method outputs a single fused adapter per task and does not support input-adaptive routing. Following the recommendations in the original paper, we randomly sample 20 adapters from the full pool to constrain the optimization search space. For evaluation, each input is answered using the fused adapter optimized for its corresponding task (i.e., we select the task-specific fused adapter based on the ground-truth task label). LoRAHub was trained using the same validation set as in our method, and we fix the number of optimization iterations to 48, matching the total number of adapter evaluations performed by our approach.

LoRARetriever (Zhao et al., 2024). A retrieve-then-compose method that ranks LoRA adapters using similarity between input and adapter representations, then fuses the top-k adapters uniformly. This method assumes the availability of supervised training data for each task and requires homogeneous adapter training conditions. Our method was evaluated using the exact same set of adapters and the same text encoder as employed in the original LoRARetriever paper.

Arrow (Ostapenko et al., 2024). A spectral routing method that applies SVD to adapter representations and selects experts based on spectral affinity with the input. It is training-free but introduces additional per-token inference overhead proportional to the number of adapters. Following the recommendations of the original paper, we use a temperature of 1 for converting spectral affinities into routing weights, and we retrieve $k = 3$ adapters during routing for consistency with our other baselines.

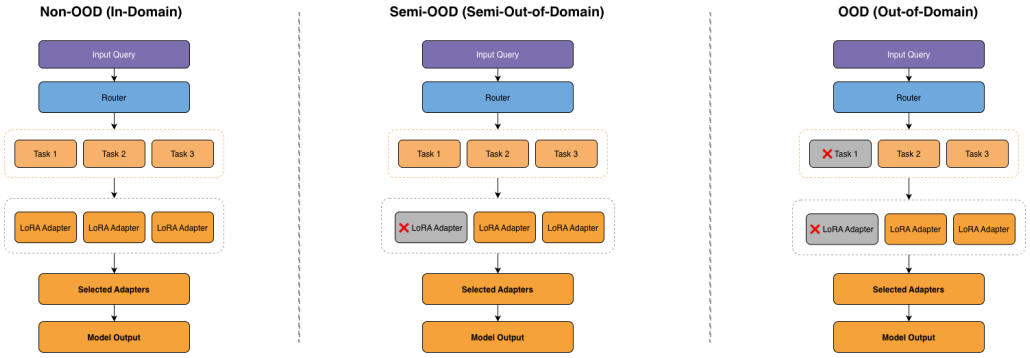

Figure 6: Illustration of the evaluation regimes used in our experiments. non-OOD (left) corresponds to the in-domain setting where both task-specific validation samples and the task-aligned LoRA adapter are available during routing. *Semi-OOD* (center) removes the aligned adapter while retaining validation samples. OOD (right) excludes both the task-specific validation samples and the corresponding adapter.

SpectR (Fleshman and Durme, 2025). A recently proposed spectral routing variant that aligns input and adapter representations through eigenspace projections. It avoids training but generally underperforms methods that incorporate task-level context. While the method supports softmax-based weighting of the selected adapters, in our evaluation we follow the configuration used in the original paper and apply uniform weights instead. Furthermore, we retrieve $k = 4$ adapters during routing, as suggested in the original paper.

Oracle task-aligned adapter. For reference, we report an *oracle task-aligned* baseline, which corresponds to selecting the LoRA adapter that was explicitly trained on the ground-truth task of each input.

Normalized average computation. To obtain a single aggregate performance metric across heterogeneous tasks and evaluation scales, we compute a normalized average. For structure-to-text generation, we first average ROUGE-1, ROUGE-2, and ROUGE-L to obtain a single score per method. Then, for each task (i.e., each row in the results table), we normalize all method scores using the corresponding value in the *Oracle Task-Aligned* column as the normalization constant. Finally, the normalized averages are computed by averaging these per-task normalized scores across all tasks.

B.2 Dataset details

We adopt the mixed-task evaluation benchmark introduced by Zhao et al. (2024), which is designed to reflect realistic multi-task inference scenarios where user prompts span heterogeneous task types and task labels are not available at inference time. The benchmark comprises a pool of 48 LoRA adapters, each trained independently on a distinct downstream task drawn from a diverse set of natural language understanding (NLU) and natural language generation (NLG) datasets. This setup mirrors practical serving environments in which LoRA adapters are specialized for different tasks.

Task taxonomy. The 48 tasks covered by the benchmark follow the high-level categorization popularized by the *FLAN* instruction-tuning framework (Wei et al., 2022). For evaluation and reporting, as illustrated in Figure 7 and reported in Zhao et al. (2024), these 48 tasks are grouped into 10 semantically coherent benchmark tasks. Concretely, the 48 adapters span task categories including *natural language inference*, *reading comprehension*, *closed-book question answering*, *commonsense reasoning*, *sentiment analysis*, *coreference resolution*, *paraphrase detection*, *structure-to-text generation*, and *machine translation*. This taxonomy ensures coverage of both classification-style tasks with discrete label spaces and open-ended generation tasks with free-form outputs.

Instruction formatting. Following the instruction-based paradigm of Wei et al. (2022), each dataset is verbalized using natural language instructions that describe the task objective (e.g., entailment determination, sentiment classification, or translation). While multiple instruction templates are used during adapter training in the original datasets, our evaluation treats adapters as

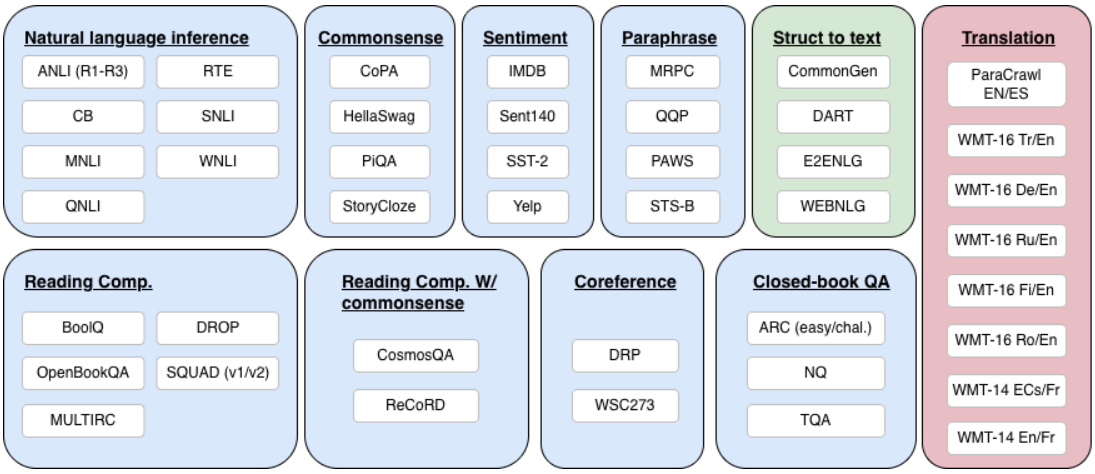

Figure 7: Overview of the task taxonomy and dataset coverage used in the mixed-task benchmark. Each LoRA adapter is trained on a dataset belonging to a specific task cluster (e.g., natural language inference, reading comprehension, translation), following the taxonomy introduced in FLAN (Wei et al., 2022). At inference time, inputs are provided without task labels, requiring routing methods to infer the underlying task and select or compose suitable adapters. Figure reproduced from LoRARetriever (Zhao et al., 2024).

black-box modules and does not assume access to their training prompts or data. At inference time, inputs are provided as plain user queries without explicit task identifiers, reflecting the mixed-task setting.

Validation and test data. For each task, a small validation subset is used exclusively for routing or adapter selection, while a disjoint test set is used for final evaluation. Validation sets are intentionally lightweight (typically a few hundred examples per task) to reflect realistic constraints where large-scale labeled data may be unavailable. Test metrics follow standard task-specific evaluation measures (e.g., accuracy, exact match, BLEU, or ROUGE), as reported in the main results.

Overall, this benchmark provides a controlled yet challenging testbed for evaluating scalable LoRA routing methods under both in-domain and distribution-shifted conditions.

C Additional results

In addition to the primary findings discussed in the main body, this section provides a comprehensive breakdown of the performance of alternative adapter routing strategies. Beyond these comparisons, we also include supplementary adapter evaluation results on the validation set. Finally, we present supplementary results for the Successive Halving (SH) algorithm to further illustrate its efficiency.

C.1 Selection-based methods

In this section, we report results for several selection-based strategies for adapter routing. While these methods achieve strong performance in the non-OOD setting, they underperform relative to weighted fusion-based routing. This gap arises because selection-based approaches are more sensitive to input–task misalignment introduced by the imperfect text encoder used for input-to-task routing.

Moreover, selection-based methods collapse in the *Semi-OOD* and OOD settings, as they lack a mechanism for aggregating knowledge across multiple related tasks and must rely on a single selected adapter. Table 4 summarizes the performance of the evaluated selection strategies across the non-OOD, *Semi-OOD*, and OOD settings.

The quantitative results in Table 4 highlight a widening performance gap as the evaluation shifts toward out-of-distribution (OOD) scenarios. In the non-OOD setting, both LoRARetriever and LoRAuter selections maintain high normalized averages (e.g., 99.0% for LoRAuter on Llama2-7B), indicating that the text encoders are generally capable of identifying the correct specialized adapter when the task is known. However, this proficiency diminishes significantly in *Semi-OOD* and OOD regimes. For instance, the normalized average for LoRAuter Selection drops from 99.0% to 81.8% on Llama2-7B when moving from non-OOD to OOD. Furthermore, while the larger Llama2-13B model exhibits higher absolute scores across most metrics, it follows a similar degradation pattern to

the 7B variant, suggesting that increased parameter scale does not inherently resolve the fundamental limitations of hard-selection routing in zero-shot or OOD contexts.

C.2 LoRaRetriever weighted fusion

This subsection evaluates the effect of applying our weighted fusion mechanism to the LoRaRetriever routing strategy. Table 5 reports results across non-OOD and OOD regimes for both Llama2-7B and Llama2-13B backbones. Incorporating weighted fusion substantially improves the previous LoRaRetriever Mixture method, and achieves performance that is nearly on par with **LoRAuter** in both the non-OOD and OOD settings. However, unlike LoRAuter, this approach does not exploit additional task samples when they are available. As a result, in the *Semi-OOD* setting, where limited validation data from the target task can be leveraged for routing, weighted fusion over LoRaRetriever cannot adapt its mixture to the new task and therefore significantly underperforms the LoRAuter method.

Comparing normalized averages across the three fusion strategies further clarifies this trend. In non-OOD, LoRaRetriever's uniform mixture reaches 92.9% (Llama2-7B) and 95.6% (Llama2-13B), while weighted fusion improves to 98.6% and 99.1%, approaching LoRAuter Fusion at 101.2% and 98.8%, respectively. In OOD, the same ordering holds: the original mixture scores 83.2% (7B) and 85.9% (13B), weighted fusion rises to 85.7% and 86.2%, and LoRAuter Fusion remains strongest at 88.4% and 86.8%. Overall, weighting reduces errors from imperfect retrieval by upweighting the most relevant retrieved adapters, yielding consistent gains over uniform mixing.

C.3 Effect of Adapter Composition

We further analyze the effect of composing multiple adapters compared to selecting only the single best retrieved adapter. Table 6 reports the performance of using only the top retrieved adapter ($K = 1$) versus composing the top-3 retrieved adapters ($K = 3$) for both LoRaRetriever (LR) and LoRAuter (LA). Across both Llama2-7B and Llama2-13B, composition consistently improves performance in the OOD setting. For example, on Llama2-7B, LA improves from 81.8% with $K = 1$ to 88.4% with $K = 3$, while LR improves from 78.4% to 83.2%. A similar trend is observed for Llama2-13B. These results suggest that OOD queries often benefit from combining complementary capabilities from multiple related adapters rather than relying on a single retrieved adapter.

C.4 Additional results on embedding model sensitivity

We further evaluate the sensitivity of LoRAuter to the embedding model by replacing the fine-tuned `e5-large-v2` encoder with the original off-the-shelf version for the Llama2 experiments. As shown in Table 7, LoRAuter (LA) consistently outperforms LR across all settings even without embedding-model fine-tuning. For example, in the OOD setting, LA improves from 83.2% to 90.0% on Llama2-7B and from 85.9% to 88.4% on Llama2-13B when using the off-the-shelf encoder. While using the original encoder slightly reduces performance compared to the fine-tuned variant in the non-OOD and semi-OOD settings, the degradation remains modest. These results suggest that LoRAuter is substantially less sensitive to embedding-model fine-tuning while still achieving strong routing performance.

C.5 Main results per benchmark task

OOD, non-OOD, semi-OOD settings. We complement Figure 2 by reporting results broken down by benchmark task. Table 8 and Table 9 present per-benchmark normalized scores, showing that LoRAuter achieves the strongest performance on the majority of tasks. This indicates that the gains observed in Figure 2 are not driven by a single task category, but are broadly consistent across diverse benchmarks.

We further observe a similar trend in the semi-OOD setting, where the task-specific adapter is unavailable but validation data for the target task remains accessible. In this regime, LoRAuter continues to outperform baselines, demonstrating its ability to leverage task-level validation signals. Compared to the fully OOD setting, semi-OOD performance improves consistently across benchmarks. Overall, these results confirm that LoRAuter 's advantages hold at a fine-grained, per-task level and extend robustly across non-OOD, semi-OOD, and OOD scenarios.

Scaling the adapter pool. The scaling results in Section 4.2 show that LoRAuter maintains performance comparable to baseline routing methods

Table 4: Performance of selection-based adapter routing methods across evaluation regimes. *Task-Aligned* reports the oracle adapter trained on the ground-truth task, while *Perf-Aligned* selects the single adapter with highest validation performance on the ground-truth task. LORARETRIEVER *Selection* and LORAUTER *Selection* denote input-aware single-adapter selection ($k = 1$) using the corresponding routing method.

LLAMA2-7B							
	ORACLE		NON-OOD		SEMI-OOD	OOD	
DOMAIN-METRIC	TASK-ALIGNED	PERF-ALIGNED	LORA-RETRIEVER SELECTION	LORAUTER SELECTION	LORAUTER SELECTION	LORA-RETRIEVER SELECTION	LORAUTER SELECTION
STRUCT TO TEXT-ROUGE-1	63.5	63.5	61.3	61.0	49.1	50.1	43.4
STRUCT TO TEXT-ROUGE-2	39.0	39.0	37.0	36.9	25.1	26.6	20.8
STRUCT TO TEXT-ROUGE-L	56.6	56.6	54.5	54.6	43.2	43.9	38.5
COMMONSENSE-EM	63.0	68.5	55.5	68.5	65.5	46.0	62.0
SENTIMENT-EM	90.0	90.5	89.5	90.0	87.5	89.0	90.0
READING COMP-EM	67.7	67.7	51.7	63.7	53.7	40.3	48.7
CLOSED_BOOK QA-EM	44.0	50.5	40.0	45.0	40.0	43.0	41.0
COREFERENCE-EM	51.0	59.0	50.0	59.0	59.0	46.0	52.0
READ.COMP.W:COM	70.0	70.0	69.0	59.0	36.0	30.0	19.0
PARAPHRASE-EM	63.5	69.0	58.0	60.5	56.0	45.5	53.0
NLI-EM	73.1	72.5	70.0	71.7	67.6	60.6	64.0
TRANSLATION-BLEU	13.1	12.6	12.8	12.6	11.9	12.0	11.7
NORMALIZED AVERAGE (%)	100.0	104.4	93.2	99.0	88.4	78.4	81.8
LLAMA2-13B							
	ORACLE		NON-OOD		SEMI-OOD	OOD	
DOMAIN-METRIC	TASK-ALIGNED	PERF-ALIGNED	LORA-RETRIEVER SELECTION	LORAUTER SELECTION	LORAUTER SELECTION	LORA-RETRIEVER SELECTION	LORAUTER SELECTION
STRUCT TO TEXT-ROUGE-1	65.0	65.0	62.6	62.6	50.4	49.4	44.1
STRUCT TO TEXT-ROUGE-2	40.4	40.4	38.2	38.2	26.8	25.8	21.9
STRUCT TO TEXT-ROUGE-L	58.3	58.3	56.0	56.0	44.0	42.9	38.9
COMMONSENSE-EM	69.0	68.5	59.0	66.0	64.0	47.5	65.5
SENTIMENT-EM	90.0	90.0	90.5	90.0	90.0	91.0	92.0
READING COMP-EM	76.3	76.3	60.3	72.3	60.7	48.0	46.0
CLOSED_BOOK QA-EM	65.0	64.0	60.0	61.0	58.0	53.0	56.5
COREFERENCE-EM	75.0	76.0	75.0	76.0	69.0	65.0	55.0
READ.COMP.W:COM	81.0	81.0	80.0	67.0	54.0	33.0	34.0
PARAPHRASE-EM	76.5	76.5	68.0	67.0	59.5	52.5	53.5
NLI-EM	82.0	82.6	78.9	79.6	69.5	70.2	73.5
TRANSLATION-BLEU	12.8	13.2	12.9	13.2	12.9	12.7	12.8
NORMALIZED AVERAGE (%)	100.0	100.3	93.8	95.2	85.8	76.7	78.3

evaluated on the 48-adapter pool, even when operating with adapters collected exclusively "from the wild," over substantially larger and noisier adapter collections, and without access to well-curated task-specific adapters.

In addition to the HF-only setting discussed in Section 4.2, we also evaluate LORAUTER on a combined adapter pool consisting of the 1,567 publicly available HUGGINGFACE adapters augmented with the original 48 curated task-specific adapters. This setting allows us to assess whether LORAUTER can effectively leverage high-quality task-aligned adapters when they are present, while remaining robust to a large number of adapters. We find that performance in this combined pool closely matches the results obtained with the curated adapters alone, indicating that the presence of a large, noisy adapter pool does not hinder LORAUTER 's ability to select and compose strong adapters. Detailed per-task results are provided in Table 10 and Table 1.

C.6 Adapter performance on the validation set

In this subsection, we analyze the normalized performance of each adapter evaluated on the validation set. Figure 8 visualizes this performance matrix for both **Llama2-7B** and **Llama2-13B** backbone models. Each row corresponds to a single evaluation task, while each column represents a single LoRA adapter trained for a specific task.

To facilitate comparison across tasks with heterogeneous metrics and score ranges, we apply row-wise normalization. Specifically, for each task (row), we subtract the minimum observed score across all adapters and divide by the range between the maximum and minimum scores for that task, resulting in values normalized to the interval $[0, 1]$. This normalization emphasizes relative adapter performance within each task while removing scale effects induced by differing evaluation metrics.

The figure reveals a pronounced diagonal trend, indicating that adapters trained on a given task generally achieve strong performance when evaluated on that same task. However, this diagonal dominance is not absolute: in several cases, adapters

Table 5: Results of applying weighted adapter fusion to LORARETRIEVER on Non-OOD and OOD tasks for Llama2-7B and Llama2-13B

	LLAMA2-7B		LLAMA2-13B	
DOMAIN-METRIC	NON-OOD	OOD	NON-OOD	OOD
STRUCT TO TEXT-ROUGE-1	59.5	48.0	60.7	50.3
STRUCT TO TEXT-ROUGE-2	33.6	24.7	36.2	27.1
STRUCT TO TEXT-ROUGE-L	52.7	42.5	53.6	43.9
COMMONSENSE-EM	60.0	58.0	70.0	67.5
SENTIMENT-EM	90.5	91.0	92.0	92.5
READING COMP-EM	59.3	47.7	70.3	53.0
CLOSED_BOOK QA-EM	47.0	46.5	66.0	59.0
COREFERENCE-EM	58.0	57.0	81.0	62.0
READ.COMP.W:COM	66.0	36.0	79.0	56.0
PARAPHRASE-EM	62.5	45.5	71.5	58.0
NLI-EM	71.0	63.9	80.0	74.9
TRANSLATION-BLEU	13.2	12.2	13.4	13.9
NORMALIZED AVERAGE (%)	98.6	85.7	99.1	86.2

Table 6: Effect of composition for LORAUTER (LA) and LORARETRIEVER (LR) on OOD tasks.

Model	Method	w/o Composition (K=1)	with Composition (K=3)
Llama2-7B	LR	78.4	83.2
	LA	81.8	88.4
Llama2-13B	LR	76.7	85.9
	LA	78.3	86.8

Table 7: Sensitivity to embedding model choice. Average normalized score on the 48-task benchmark using either the original off-the-shelf or fine-tuned `e5-large-v2` embedding model.

Setup	LA (original)	LA (finetuned)	LR (finetuned)
Llama2-7B			
Non-OOD	97.4	101.2	92.9
Semi-OOD	92.6	92.7	83.2
OOD	90.0	88.4	83.2
Llama2-13B			
Non-OOD	96.7	98.8	95.6
Semi-OOD	89.5	91.1	85.9
OOD	88.4	86.8	85.9

trained on related tasks outperform the task-specific adapter.

In addition to task-specific peaks, we observe consistent blocks of elevated performance among adapters belonging to the same task group. This effect is especially visible for the *structure-to-text* and *machine translation* tasks in the bottom-right of the figure, where adapters form well-defined blocks with strong intra-group generalization, reflecting both their shared structure and their use of different evaluation metrics compared to classification-style tasks.

C.7 Successive halving

To efficiently evaluate the effectiveness of *Successive Halving* (SH) for adapter selection, we reuse pre-computed adapter outputs obtained from up to 200 validation samples per task. We exclude tasks with fewer than 200 validation examples to ensure that every run can draw sufficiently large subsets while keeping the evaluation protocol consistent across tasks. To compare performance across heterogeneous metrics and scales, we normalize each task by the same Oracle Task-Aligned domain normalization constants as before, and report the resulting normalized average across tasks.

We estimate the SH and uniform evaluation curves using 100 independent runs per evaluation budget. In each run, we randomly sample subsets of varying sizes from the original pool of 200 validation samples, perform adapter selection under the corresponding budget, and then evaluate the

Algorithm 1 Successive Halving

Input: adapters $\mathcal{A}$, validation set $\mathcal{V}$, base samples m, keep ratio $\eta \in (0,1)$, budget growth $\gamma > 1$, rounds R, warmup rounds k
Output: selected adapter $a^\star$
$\mathcal{S} \leftarrow \mathcal{A}$
for $r = 0$ **to** $R-1$ **do**
  if $r < k$ **then**
    $m_r \leftarrow m$ // fixed budget during warmup
  else
    $m_r \leftarrow \lceil m \cdot \gamma^{r-k+1} \rceil$ // grow budget after warmup
  end if
  $m_r \leftarrow \min(m_r, |\mathcal{V}|)$
  Sample $S_r \subseteq \mathcal{V}$ with $|S_r| = m_r$
  for all $a \in \mathcal{S}$ **do**
    $s[a] \leftarrow \mathrm{Score}(a, S_r)$
  end for
  $q \leftarrow \max(1, \lceil \eta \cdot |\mathcal{S}| \rceil)$
  $\mathcal{S} \leftarrow$ top-q adapters in $\mathcal{S}$ by $s[\cdot]$
  if $|\mathcal{S}| = 1$ **then**
    break
  end if
end for
$a^\star \leftarrow$ the single remaining adapter in $\mathcal{S}$

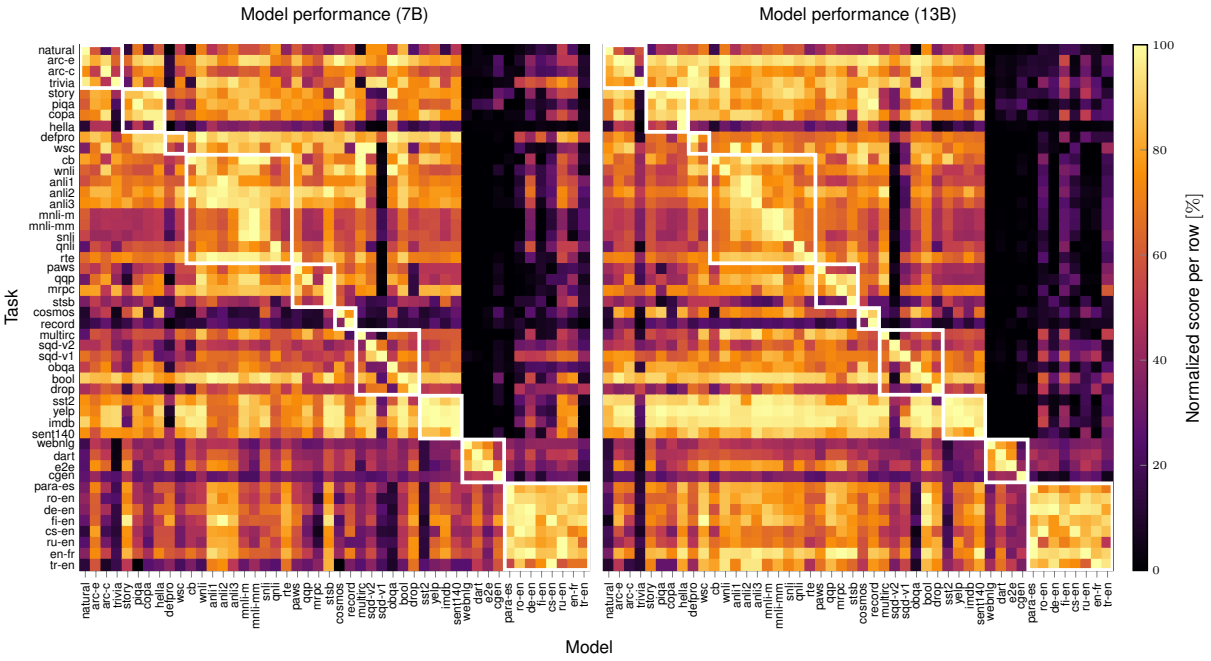

Figure 8: Performance matrix of adapters across tasks evaluated on the validation data. Each row corresponds to one task, while each column represents a single LoRA adapter trained for a specific task.

selected adapter on the task's held-out test set.

Pseudo-code. Algorithm 1 summarizes the Successive Halving (SH) procedure used for adapter selection under a fixed evaluation budget (measured in adapter-sample evaluations). Compared to the standard formulation, we (i) use a fixed per-adapter evaluation budget for the first k rounds (warmup), and (ii) decouple the elimination rate from the budget schedule by using separate constants for retention (η) and per-round budget growth (γ). Starting from an initial candidate set, SH evaluates each adapter on a validation subset, retains the top fraction, and repeats with an increased budget allocation until a single adapter remains.

Figure 9 confirms that the Llama2-7B trend reported in the main text (Figure 5) carries over to Llama2-13B.

C.8 Choosing K for Input-Aware Adapter Fusion

In Table 11, we evaluate how the number of tasks selected for each input (K) affects LoRAuter performance with Llama2-7B in the in-distribution setting. We vary K from 1 (selecting a single task, i.e., using one adapter without any fusion) to 5 (fusing the top-5 task-specific adapters per input). The best performance is achieved at $K = 2$, with $K = 3$ performing essentially identically. We therefore select $K = 3$ throughout the rest of the paper to remain consistent with prior fusion-based baselines.

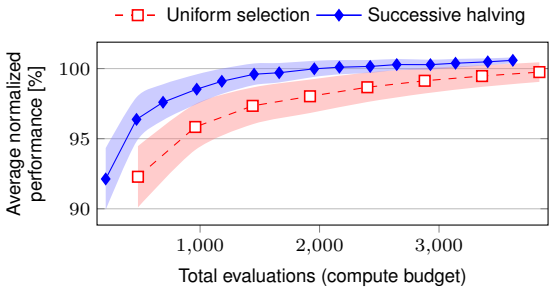

Figure 9: Normalized performance of Successive Halving over uniform selection with respect to their allocated compute budget for the 13B model. Standard deviation is reported over the randomness of the validation set ordering with 100 samples.

D Full results

In Tables 12 and 13, we report detailed per-task results for LoRAuter across each of the 48 tasks in the evaluation dataset. Results are reported for both Llama2-7B and Llama2-13B backbone models and cover all evaluation regimes considered in the study.

In Tables 14 and 15, we report per-task results for the LoRAHub, ARROW, and SpectR baselines on both Llama2-7B and Llama2-13B, evaluated under the Non-OOD, Semi-OOD, and OOD regimes.

In Table 16, we report per-task results for LoRAuter on Llama2-7B using 1567 publicly available HuggingFace adapters with rank ≤ 64, evaluated under Non-OOD, Semi-OOD, and OOD regimes.

In Tables 17 and 18, we present detailed per-task results for selection-based routing methods on both Llama2-7B and Llama2-13B across the same set of 48 evaluation tasks. These tables include results for oracle task-aligned selection, performance-aligned selection, and input-aware selection strategies based on both LoRARetriever and LoRAuter.

In Table 19, we present detailed per-task results for the LoRARetriever weighted-mixture routing method on both Llama2-7B and Llama2-13B. All results follow the same evaluation protocol, enabling direct comparison across methods and evaluation settings.

Table 8: Adapter performance across multiple domains for LLaMA2-7B, LLaMA2-13B and Llama3.1-8B. *Non-OOD* denotes in-domain tasks, while *OOD* and *Semi-OOD* correspond to different out-of-domain evaluation settings. The LoRAuter column represents the proposed input-aware adapter routing method. The *Oracle Task-Aligned* baseline (shaded in gray) reports performance of the adapter trained on the ground-truth task of each input. Results for the LoRARetriever *Mixture* are taken directly from the original LoRARetriever paper. All average scores are normalized by the Task-Aligned performance per task.

Llama2-7B							
		Non-OOD		OOD		Semi-OOD	
Domain-Metric	Oracle Task-Aligned	LoRA-Retriever Mixture	LoRAuter	LoRA-Retriever Mixture	LoRAuter	LoRAuter	LoRAHub
struct to text-rouge-1	63.5	55.9	**58.8**	**50.4**	48.5	**51.4**	44.8
struct to text-rouge-2	39.0	30.0	**33.6**	**26.9**	25.3	**27.3**	23.1
struct to text-rouge-l	56.6	49.5	**51.8**	**44.0**	43.1	**45.3**	39.8
commonsense-em	63.0	61.5	**68.0**	50.0	**63.0**	**65.5**	42.5
sentiment-em	90.0	89.5	**90.5**	90.5	**91.0**	**91.0**	63.5
reading comp-em	67.7	51.3	**62.3**	**47.3**	**47.3**	**51.7**	45.0
closed_book qa-em	44.0	45.0	**50.5**	**48.5**	47.5	**46.0**	34.0
coreference-em	51.0	63.0	**63.0**	49.0	**54.0**	**65.0**	46.0
read.comp.w:com	70.0	46.0	**67.0**	**40.0**	39.0	**42.0**	14.0
paraphrase-em	63.5	56.5	**60.0**	46.0	**53.5**	**57.5**	38.0
nli-em	73.1	67.9	**70.3**	56.5	**64.0**	**69.3**	50.6
translation-bleu	13.1	**12.8**	12.6	12.2	**12.9**	**11.8**	11.7
Normalized Average (%)	100.0	92.9	**101.2**	83.2	**88.4**	**92.7**	67.8

Llama2-13B							
		Non-OOD		OOD		Semi-OOD	
Domain-Metric	Oracle Task-Aligned	LoRA-Retriever Mixture	LoRAuter	LoRA-Retriever Mixture	LoRAuter	LoRAuter	LoRAHub
struct to text-rouge-1	65.0	57.7	**61.1**	**52.1**	50.7	**53.3**	44.4
struct to text-rouge-2	40.4	32.6	**36.5**	**28.1**	27.2	**29.3**	22.9
struct to text-rouge-l	58.3	50.8	**53.9**	**45.4**	43.8	**47.0**	39.0
commonsense-em	69.0	64.0	**70.5**	60.5	**67.0**	**69.5**	56.0
sentiment-em	90.0	**91.5**	**91.5**	91.5	**92.5**	**91.0**	41.5
reading comp-em	76.3	60.3	**71.3**	51.3	**52.3**	**60.7**	48.0
closed_book qa-em	65.0	**63.0**	**63.0**	**61.0**	59.5	**60.0**	59.5
coreference-em	75.0	76.0	**77.0**	64.0	**66.0**	**68.0**	48.0
read.comp.w:com	81.0	78.0	**79.0**	**58.0**	56.0	**63.0**	30.0
paraphrase-em	76.5	71.0	**71.0**	55.5	**59.0**	**66.0**	36.5
nli-em	82.0	78.1	**79.5**	**75.7**	74.7	**75.7**	49.6
translation-bleu	12.8	**14.6**	14.3	**14.1**	13.9	**13.9**	13.5
Normalized Average (%)	100.0	95.6	**98.8**	85.9	**86.8**	**91.0**	66.1

Llama3.1-8B							
		Non-OOD		OOD		Semi-OOD	
Domain-Metric	Oracle Task-Aligned	LoRA-Retriever Mixture	LoRAuter	LoRA-Retriever Mixture	LoRAuter	LoRAuter	LoRAHub
struct to text-rouge-1	66.8	58.8	**62.2**	50.7	**52.4**	**55.5**	43.6
struct to text-rouge-2	42.5	32.8	**37.4**	27.0	**28.0**	**30.4**	21.7
struct to text-rouge-l	59.6	51.7	**55.2**	43.8	**45.2**	**48.9**	38.4
commonsense-em	74.5	**71.0**	70.5	**66.5**	66.0	**69.0**	63.5
sentiment-em	90.5	**92.0**	91.0	91.0	**91.5**	**91.5**	70.0
reading comp-em	79.0	71.0	**74.3**	53.7	**54.0**	**64.0**	35.7
closed_book qa-em	67.5	65.5	**67.0**	61.5	**64.5**	**63.5**	43.5
coreference-em	75.0	71.0	**72.0**	**70.0**	63.0	**61.0**	56.0
read.comp.w:com	55.0	**54.0**	**54.0**	**33.0**	**33.0**	**38.0**	25.0
paraphrase-em	81.0	71.0	**75.0**	49.5	**52.0**	**58.0**	42.5
nli-em	79.3	**82.0**	81.8	**77.3**	74.1	**76.5**	38.8
translation-bleu	15.4	15.3	**16.3**	15.2	**15.7**	**15.5**	11.0
Normalized Average (%)	100.0	95.2	**97.6**	83.1	**83.2**	**86.8**	62.7

Table 9: Baseline adapter performance across multiple domains for LLaMA2-7B, LLaMA2-13B and Llama3.1-8B. *Non-OOD* denotes in-domain tasks, while *OOD* corresponds to the out-of-domain evaluation setting.

LLAMA2-7B						
	NON-OOD			OOD		
DOMAIN-METRIC	LORAHUB	ARROW	SPECTR	LORAHUB	ARROW	SPECTR
STRUCT TO TEXT-ROUGE-1	55.0	39.5	44.2	N/A	40.1	44.2
STRUCT TO TEXT-ROUGE-2	31.9	19.6	22.5	N/A	20.1	22.5
STRUCT TO TEXT-ROUGE-L	48.0	35.5	39.5	N/A	35.6	39.5
COMMONSENSE-EM	44.5	59.0	47.0	N/A	54.0	47.0
SENTIMENT-EM	61.0	82.5	69.5	N/A	86.0	69.5
READING COMP-EM	44.0	51.7	43.0	N/A	50.7	43.0
CLOSED_BOOK QA-EM	34.5	46.5	37.0	N/A	47.0	37.0
COREFERENCE-EM	43.0	60.0	57.0	N/A	59.0	57.0
READ.COMP.W:COM	15.0	31.0	25.0	N/A	27.0	25.0
PARAPHRASE-EM	35.0	45.0	42.5	N/A	49.5	42.5
NLI-EM	49.2	56.2	53.5	N/A	59.9	53.5
TRANSLATION-BLEU	11.9	11.6	11.6	N/A	10.8	11.6
NORMALIZED AVERAGE (%)	68.6	82.5	74.2	N/A	82.0	66.3
LLAMA2-13B						
	NON-OOD			OOD		
DOMAIN-METRIC	LORAHUB	ARROW	SPECTR	LORAHUB	ARROW	SPECTR
STRUCT TO TEXT-ROUGE-1	57.1	45.3	48.0	N/A	46.1	48.0
STRUCT TO TEXT-ROUGE-2	34.8	23.6	25.0	N/A	23.8	25.0
STRUCT TO TEXT-ROUGE-L	50.9	40.1	42.0	N/A	39.9	42.0
COMMONSENSE-EM	55.5	68.5	59.5	N/A	64.0	59.5
SENTIMENT-EM	43.0	90.5	75.0	N/A	91.0	75.0
READING COMP-EM	47.3	55.7	51.7	N/A	53.0	51.7
CLOSED_BOOK QA-EM	59.0	64.5	54.5	N/A	62.5	54.5
COREFERENCE-EM	45.0	61.0	53.0	N/A	61.0	53.0
READ.COMP.W:COM	30.0	59.0	39.0	N/A	52.0	39.0
PARAPHRASE-EM	38.5	52.5	44.5	N/A	54.5	44.5
NLI-EM	48.5	70.2	59.8	N/A	70.7	59.8
TRANSLATION-BLEU	13.7	14.1	14.0	N/A	13.1	14.0
NORMALIZED AVERAGE (%)	68.2	85.7	75.1	N/A	83.2	69.9
LLAMA3.1-8B						
	NON-OOD			OOD		
DOMAIN-METRIC	LORAHUB	ARROW	SPECTR	LORAHUB	ARROW	SPECTR
STRUCT TO TEXT-ROUGE-1	51.5	33.0	44.8	N/A	27.0	44.2
STRUCT TO TEXT-ROUGE-2	29.4	16.2	23.4	N/A	12.5	23.3
STRUCT TO TEXT-ROUGE-L	45.2	30.6	39.6	N/A	24.9	39.1
COMMONSENSE-EM	68.0	70.5	69.5	N/A	70.5	71.0
SENTIMENT-EM	69.5	89.0	90.0	N/A	89.0	91.0
READING COMP-EM	35.3	62.3	60.3	N/A	61.3	58.7
CLOSED_BOOK QA-EM	43.0	64.5	61.5	N/A	65.0	60.5
COREFERENCE-EM	57.0	67.0	62.0	N/A	65.0	62.0
READ.COMP.W:COM	25.0	37.0	30.0	N/A	29.0	26.0
PARAPHRASE-EM	42.0	61.5	63.5	N/A	57.5	58.5
NLI-EM	38.2	74.8	73.3	N/A	73.3	71.7
TRANSLATION-BLEU	11.1	13.9	15.2	N/A	13.4	15.1
NORMALIZED AVERAGE (%)	64.5	83.2	83.1	N/A	79.5	81.4

Table 10: Per-domain results for LoRAuter on Llama2-7B across Non-OOD, Semi-OOD and OOD settings for two different adapter pools: only 48 well curated adapters (48), only adapters from the wild (HF), both well curated and from the wild adapters (HF+48)

	HF+48			HF	
Domain-Metric	Non-OOD	Semi-OOD	OOD	Semi-OOD	OOD
struct to text-rouge-1	58.8	51.2	49.6	48.7	36.2
struct to text-rouge-2	33.6	27.1	25.6	24.2	18.1
struct to text-rouge-L	52.0	44.7	43.9	43.5	32.0
commonsense-em	68.0	65.5	64.0	65.5	63.5
sentiment-em	90.5	90.5	91.5	89.0	88.5
reading comp-em	61.7	53.3	48.3	51.0	49.7
closed_book QA-em	48.5	46.0	47.0	41.5	37.0
coreference-em	60.0	59.0	54.0	62.0	64.0
read.comp.w:com	68.0	44.0	46.0	34.0	37.0
paraphrase-em	61.0	57.0	52.5	38.5	52.5
nli-em	69.9	68.2	68.0	69.6	70.3
translation-bleu	12.4	10.9	11.8	11.5	5.2

Table 11: Ablation with different values of K. Based on these results, we select $K = 3$, which achieves performance close to the best result ($K = 2$) while remaining consistent with baselines which also use $K = 3$.

Value of K	1	2	3	4	5
struct to text-rouge-1	**60.9**	59.5	58.4	58.5	58.9
struct to text-rouge-2	**36.6**	34.2	33.1	33.2	33.6
struct to text-rouge-l	**54.4**	52.9	51.3	51.8	51.7
commonsense-em	68.0	**69.5**	67.5	67.0	68.5
sentiment-em	**91.0**	90.5	90.5	90.5	90.5
reading comp-em	**64.0**	63.7	62.7	61.0	61.3
closed_book QA-em	45.0	48.5	49.5	49.0	**51.5**
coreference-em	58.0	62.0	**66.0**	63.0	65.0
read.comp.w:commonsense-em	59.0	**69.0**	67.0	62.0	61.0
paraphrase-em	61.5	**62.0**	60.5	60.5	61.5
nli-em	**70.9**	69.5	70.1	70.5	70.5
translation-bleu	12.5	**12.8**	12.7	12.7	12.7
Normalized Average (%)	98.9	**101.8**	101.6	99.9	101.2

Table 12: Per-task results for fusion-based adapter routing methods on Llama2-7B across Non-OOD, Semi-OOD, and OOD evaluation regimes. Results for the LORARETRIEVER Mixture baseline are directly copied from the original LORARETRIEVER paper.

		Non-OOD		Semi-OOD	OOD	
Domain-Metric	Oracle Task-Aligned	LoRA-Retriever Mixture	LoRAuter Fusion	LoRAuter Fusion	LoRA-Retriever Mixture	LoRAuter Fusion
Struct to Text						
web_nlg_en-rouge-1	69.6	57.8	63.4	52.3	53.9	49.9
web_nlg_en-rouge-2	48.0	33.5	41.9	29.0	29.4	28.0
web_nlg_en-rouge-l	62.8	52.3	56.3	48.1	49.6	45.4
dart-rouge-1	71.7	63.2	60.1	60.3	60.0	57.4
dart-rouge-2	49.4	36.6	34.4	35.7	35.4	33.6
dart-rouge-l	64.9	56.3	53.1	54.1	52.4	51.8
e2e_nlg-rouge-1	66.1	66.0	64.1	63.1	58.7	55.6
e2e_nlg-rouge-2	39.6	38.8	37.6	36.0	32.1	30.4
e2e_nlg-rouge-l	56.4	56.9	54.0	52.9	49.0	48.6
common_gen-rouge-1	46.5	36.5	47.8	30.1	29.0	31.0
common_gen-rouge-2	18.9	11.1	20.6	8.4	8.6	9.2
common_gen-rouge-l	42.2	32.7	43.6	26.0	24.8	26.7
Translation						
para_crawl_enes-bleu	24.0	22.8	23.9	19.5	22.1	23.6
wmt16_translate_tren-bleu	3.7	3.7	2.0	2.0	2.6	2.3
wmt16_translate_deen-bleu	17.8	11.0	19.5	18.5	10.8	20.4
wmt16_translate_ruen-bleu	11.5	18.8	11.9	11.3	18.7	12.0
wmt16_translate_fien-bleu	6.8	7.3	7.0	5.9	7.8	6.6
wmt16_translate_roen-bleu	14.6	13.1	12.4	13.2	12.2	13.8
wmt14_enfr-bleu	16.0	17.8	16.8	16.8	18.0	17.9
wmt16_translate_csen-bleu	10.5	8.3	7.3	7.2	5.8	6.5
COMMONSENSE						
story_cloze-em	72.0	84.0	96.0	96.0	58.0	88.0
piqa-em	44.0	38.0	44.0	38.0	34.0	42.0
copa-em	86.0	80.0	82.0	80.0	68.0	78.0
hellaswag-em	50.0	44.0	50.0	48.0	40.0	44.0
SENTIMENT						
sst2-em	98.0	96.0	98.0	98.0	94.0	94.0
yelp_polarity_reviews-em	98.0	98.0	98.0	98.0	98.0	98.0
imdb_reviews-em	96.0	96.0	98.0	98.0	96.0	98.0
sentiment140-em	68.0	68.0	68.0	70.0	74.0	74.0
READING Comp.						
multirc-em	68.0	48.0	64.0	56.0	44.0	44.0
squad_v2-em	62.0	22.0	54.0	20.0	16.0	14.0
squad_v1-em	68.0	62.0	72.0	68.0	68.0	64.0
openbookqa-em	84.0	78.0	80.0	76.0	66.0	74.0
bool_q-em	84.0	80.0	82.0	74.0	76.0	74.0
drop-em	40.0	18.0	22.0	16.0	14.0	14.0
CLOSE-BOOK QA						
natural_questions-em	18.0	16.0	14.0	12.0	10.0	12.0
arc_easy-em	48.0	66.0	78.0	76.0	82.0	82.0
arc_challenge-em	46.0	50.0	56.0	50.0	46.0	48.0
trivia_qa-em	64.0	48.0	54.0	46.0	56.0	48.0
COREFERENCE						
definite_pronoun_resolution-em	52.0	68.0	70.0	72.0	56.0	58.0
wsc-em	50.0	58.0	56.0	58.0	42.0	50.0
READ. COMP. W/ COM						
cosmos_qa-em	70.0	50.0	72.0	52.0	46.0	48.0
record-em	70.0	42.0	62.0	32.0	34.0	30.0
PARAPHRASE						
paws_wiki-em	90.0	56.0	60.0	66.0	46.0	54.0
glue_qqp-em	70.0	80.0	78.0	78.0	58.0	74.0
glue_mrpc-em	60.0	60.0	66.0	66.0	58.0	62.0
stsb-em	34.0	30.0	36.0	20.0	20.0	24.0
NLI						
cb-em	88.9	86.7	88.9	86.7	66.7	57.8
wnli-em	72.0	60.0	56.0	58.0	54.0	50.0
anli_r1-em	52.0	40.0	54.0	50.0	42.0	48.0
anli_r2-em	46.0	46.0	46.0	44.0	46.0	48.0
anli_r3-em	46.0	44.0	46.0	46.0	50.0	48.0
mnli_matched-em	88.0	80.0	86.0	86.0	88.0	86.0
mnli_mismatched-em	92.0	88.0	94.0	92.0	90.0	94.0
snli-em	96.0	90.0	84.0	84.0	92.0	82.0
qnli-em	94.0	74.0	70.0	68.0	38.0	66.0
rte-em	56.0	70.0	78.0	78.0	76.0	60.0

Table 13: Per-task results for fusion-based adapter routing methods on Llama2-13B across Non-OOD, Semi-OOD, and OOD evaluation regimes. Results for the LORARETRIEVER Mixture baseline are directly copied from the original LORARETRIEVER paper.

		Non-OOD		Semi-OOD	OOD	
Domain-Metric	Oracle Task-Aligned	LoRA-Retriever Mixture	LoRAuter Fusion	LoRAuter Fusion	LoRA-Retriever Mixture	LoRAuter Fusion
Struct to Text						
web_nlg_en-rouge-1	72.4	59.7	68.9	57.4	53.7	52.4
web_nlg_en-rouge-2	52.1	35.6	46.9	31.6	29.1	28.9
web_nlg_en-rouge-l	66.1	55.1	61.0	50.8	49.0	46.6
dart-rouge-1	72.9	62.6	64.3	59.0	60.6	59.2
dart-rouge-2	52.9	38.9	40.4	35.4	37.3	35.6
dart-rouge-l	66.7	55.0	56.4	53.1	53.4	51.7
e2e_nlg-rouge-1	66.5	66.7	65.1	64.4	63.9	61.6
e2e_nlg-rouge-2	38.8	39.0	37.4	36.7	36.4	33.9
e2e_nlg-rouge-l	56.4	56.2	55.5	53.4	53.7	51.6
common_gen-rouge-1	48.4	41.7	46.3	32.5	30.0	29.7
common_gen-rouge-2	17.9	17.0	21.2	13.6	9.5	10.6
common_gen-rouge-l	44.1	36.7	42.8	30.5	25.3	25.4
Translation						
para_crawl_enes-bleu	25.0	27.4	25.9	24.3	25.8	23.6
wmt16_translate_tren-bleu	2.0	3.4	3.7	3.6	2.7	2.8
wmt16_translate_deen-bleu	19.8	10.5	21.3	21.7	12.0	20.9
wmt16_translate_ruen-bleu	11.3	20.2	12.4	11.9	20.2	11.9
wmt16_translate_fien-bleu	6.8	8.7	9.3	8.2	7.8	8.6
wmt16_translate_roen-bleu	14.2	17.4	11.7	11.2	13.3	13.7
wmt14_enfr-bleu	16.5	18.7	20.5	20.5	20.9	20.9
wmt16_translate_csen-bleu	6.9	10.3	9.5	9.7	10.1	9.0
COMMONSENSE						
story_cloze-em	94.0	80.0	98.0	96.0	76.0	96.0
piqa-em	48.0	46.0	50.0	48.0	46.0	48.0
copa-em	76.0	78.0	80.0	82.0	76.0	80.0
hellaswag-em	58.0	52.0	54.0	52.0	44.0	44.0
SENTIMENT						
sst2-em	98.0	98.0	98.0	96.0	100.0	100.0
yelp_polarity_reviews-em	98.0	98.0	98.0	98.0	98.0	100.0
imdb_reviews-em	96.0	98.0	98.0	98.0	98.0	98.0
sentiment140-em	68.0	72.0	72.0	72.0	70.0	72.0
READING Comp.						
multirc-em	88.0	66.0	78.0	66.0	44.0	40.0
squad_v2-em	68.0	60.0	68.0	38.0	60.0	18.0
squad_v1-em	74.0	34.0	70.0	70.0	24.0	64.0
openbookqa-em	86.0	88.0	88.0	78.0	82.0	78.0
bool_q-em	84.0	78.0	84.0	86.0	78.0	86.0
drop-em	58.0	36.0	40.0	26.0	20.0	28.0
CLOSE-BOOK QA						
natural_questions-em	30.0	24.0	26.0	18.0	16.0	18.0
arc_easy-em	94.0	94.0	94.0	90.0	94.0	94.0
arc_challenge-em	68.0	68.0	70.0	70.0	70.0	70.0
trivia_qa-em	68.0	66.0	62.0	62.0	64.0	56.0
COREFERENCE						
definite_pronoun_resolution-em	88.0	88.0	90.0	76.0	72.0	74.0
wsc-em	62.0	64.0	64.0	60.0	56.0	58.0
READ. COMP. W/ COM						
cosmos_qa-em	84.0	82.0	82.0	80.0	70.0	60.0
record-em	78.0	74.0	76.0	46.0	46.0	18.0
PARAPHRASE						
paws_wiki-em	92.0	88.0	88.0	78.0	74.0	64.0
glue_qqp-em	86.0	84.0	82.0	82.0	62.0	82.0
glue_mrpc-em	84.0	78.0	72.0	68.0	64.0	64.0
stsb-em	44.0	34.0	42.0	36.0	22.0	26.0
NLI						
cb-em	95.6	93.3	93.3	91.1	88.9	86.7
wnli-em	72.0	76.0	74.0	68.0	66.0	64.0
anli_r1-em	70.0	64.0	66.0	64.0	68.0	68.0
anli_r2-em	66.0	60.0	62.0	54.0	60.0	56.0
anli_r3-em	68.0	62.0	66.0	62.0	60.0	60.0
mnli_matched-em	86.0	86.0	86.0	86.0	88.0	84.0
mnli_mismatched-em	90.0	94.0	92.0	92.0	100.0	92.0
snli-em	90.0	90.0	92.0	90.0	92.0	92.0
qnli-em	94.0	74.0	84.0	76.0	58.0	74.0
rte-em	88.0	82.0	80.0	74.0	76.0	70.0

Table 14: Performance of baseline routing methods on Llama2-7B .

	Non-OOD			Semi-OOD	OOD	
Domain-Metric	LoRAHub	ARROW	SpectR	LoRAHub	ARROW	SpectR
Struct to Text						
web_nlg_en-rouge-1	67.2	30.8	41.6	46.6	30.6	33.6
web_nlg_en-rouge-2	44.4	15.8	22.0	24.2	15.5	16.9
web_nlg_en-rouge-l	59.1	29.4	39.0	42.6	28.2	31.6
dart-rouge-1	61.2	49.4	53.6	60.4	50.8	44.0
dart-rouge-2	36.7	26.3	30.3	36.0	29.4	24.1
dart-rouge-l	54.7	44.4	48.3	53.7	46.2	39.5
e2e_nlg-rouge-1	65.1	49.7	53.5	47.2	52.1	49.2
e2e_nlg-rouge-2	38.6	27.2	29.3	25.3	27.7	26.7
e2e_nlg-rouge-l	55.1	42.4	45.4	40.8	43.5	42.0
common_gen-rouge-1	26.3	28.2	28.1	24.9	26.9	25.7
common_gen-rouge-2	7.8	9.0	8.3	6.8	7.8	6.2
common_gen-rouge-l	23.0	25.7	25.4	22.2	24.7	24.0
Translation						
para_crawl_enes-bleu	23.6	21.6	23.4	22.6	19.5	19.6
wmt16_translate_tren-bleu	3.5	2.7	2.4	2.4	3.8	2.9
wmt16_translate_deen-bleu	16.9	17.6	16.6	16.4	17.5	12.6
wmt16_translate_ruen-bleu	10.1	9.4	9.1	9.7	6.8	6.9
wmt16_translate_fien-bleu	7.3	7.2	6.6	8.4	5.9	5.9
wmt16_translate_roen-bleu	10.6	10.3	11.3	10.9	9.3	8.4
wmt14_enfr-bleu	16.6	17.1	17.0	16.9	16.8	15.0
wmt16_translate_csen-bleu	6.4	7.0	6.4	6.5	6.9	6.8
COMMONSENSE						
story_cloze-em	62.0	84.0	70.0	64.0	70.0	74.0
piqa-em	30.0	38.0	34.0	28.0	40.0	32.0
copa-em	62.0	74.0	66.0	66.0	70.0	70.0
hellaswag-em	24.0	40.0	18.0	12.0	36.0	12.0
SENTIMENT						
sst2-em	66.0	82.0	70.0	72.0	90.0	50.0
yelp_polarity_reviews-em	72.0	94.0	82.0	70.0	96.0	76.0
imdb_reviews-em	50.0	92.0	70.0	52.0	96.0	62.0
sentiment140-em	56.0	62.0	56.0	60.0	62.0	50.0
READING Comp.						
multirc-em	42.0	46.0	38.0	44.0	50.0	42.0
squad_v2-em	24.0	24.0	22.0	26.0	22.0	28.0
squad_v1-em	66.0	74.0	62.0	66.0	72.0	48.0
openbookqa-em	52.0	66.0	56.0	54.0	66.0	58.0
bool_q-em	72.0	82.0	70.0	70.0	80.0	66.0
drop-em	8.0	18.0	10.0	10.0	14.0	4.0
CLOSE-BOOK QA						
natural_questions-em	14.0	20.0	12.0	12.0	20.0	10.0
arc_easy-em	48.0	64.0	56.0	48.0	66.0	52.0
arc_challenge-em	30.0	44.0	34.0	30.0	44.0	32.0
trivia_qa-em	46.0	58.0	46.0	46.0	58.0	36.0
COREFERENCE						
definite_pronoun_resolution-em	50.0	68.0	62.0	56.0	66.0	58.0
wsc-em	36.0	52.0	52.0	36.0	52.0	52.0
READ. COMP. W/ COM						
cosmos_qa-em	24.0	32.0	38.0	22.0	34.0	10.0
record-em	6.0	30.0	12.0	6.0	20.0	6.0
PARAPHRASE						
paws_wiki-em	46.0	48.0	42.0	44.0	60.0	48.0
glue_qqp-em	20.0	52.0	52.0	32.0	70.0	58.0
glue_mrpc-em	62.0	60.0	62.0	62.0	52.0	60.0
stsb-em	12.0	20.0	14.0	14.0	16.0	8.0
NLI						
cb-em	44.4	75.6	71.1	57.8	73.3	80.0
wnli-em	44.0	52.0	54.0	46.0	48.0	46.0
anli_r1-em	36.0	32.0	36.0	34.0	38.0	34.0
anli_r2-em	34.0	32.0	36.0	34.0	40.0	28.0
anli_r3-em	44.0	38.0	40.0	44.0	40.0	36.0
mnli_matched-em	56.0	60.0	54.0	56.0	70.0	44.0
mnli_mismatched-em	76.0	88.0	72.0	74.0	86.0	60.0
snli-em	64.0	70.0	62.0	64.0	74.0	58.0
qnli-em	46.0	58.0	52.0	46.0	68.0	56.0
rte-em	48.0	56.0	58.0	50.0	62.0	58.0

Table 15: Performance of baseline routing methods on Llama2-13B.

	Non-OOD			Semi-OOD	OOD	
Domain-Metric	LoRAHub	ARROW	SpectR	LoRAHub	ARROW	SpectR
Struct to Text						
web_nlg_en-rouge-1	71.9	39.8	47.3	42.5	37.7	41.4
web_nlg_en-rouge-2	51.9	20.1	23.5	21.9	17.6	20.1
web_nlg_en-rouge-l	66.1	36.3	42.3	38.9	34.0	38.2
dart-rouge-1	63.0	56.5	56.5	63.0	57.3	45.6
dart-rouge-2	37.8	34.1	33.6	38.0	33.2	27.2
dart-rouge-l	55.9	51.5	51.1	55.7	49.6	40.2
e2e_nlg-rouge-1	66.9	55.9	57.6	49.3	56.6	57.1
e2e_nlg-rouge-2	39.1	29.5	31.0	23.8	30.6	30.3
e2e_nlg-rouge-l	56.6	45.4	46.5	40.0	45.2	45.7
common_gen-rouge-1	26.7	29.0	30.4	22.9	32.8	29.3
common_gen-rouge-2	10.3	10.6	12.0	8.1	13.7	11.4
common_gen-rouge-l	24.9	27.4	28.2	21.4	30.9	27.3
Translation						
para_crawl_enes-bleu	22.7	24.5	26.9	22.8	23.2	23.4
wmt16_translate_tren-bleu	4.0	3.9	3.3	3.3	3.6	3.3
wmt16_translate_deen-bleu	18.7	18.9	19.8	18.5	16.2	18.9
wmt16_translate_ruen-bleu	12.2	11.9	11.2	13.0	10.0	10.5
wmt16_translate_fien-bleu	9.9	8.9	8.2	9.9	8.2	6.9
wmt16_translate_roen-bleu	12.1	13.5	12.5	11.8	13.3	10.5
wmt14_enfr-bleu	19.3	19.1	19.6	18.3	18.8	18.9
wmt16_translate_csen-bleu	11.1	11.8	10.7	10.7	11.5	11.5
COMMONSENSE						
story_cloze-em	78.0	96.0	92.0	78.0	94.0	90.0
piqa-em	42.0	48.0	44.0	42.0	42.0	44.0
copa-em	68.0	86.0	72.0	68.0	78.0	70.0
hellaswag-em	34.0	44.0	30.0	36.0	42.0	22.0
SENTIMENT						
sst2-em	32.0	98.0	78.0	34.0	100.0	70.0
yelp_polarity_reviews-em	46.0	98.0	88.0	46.0	98.0	86.0
imdb_reviews-em	48.0	96.0	68.0	42.0	98.0	62.0
sentiment140-em	46.0	70.0	66.0	44.0	68.0	72.0
READING Comp.						
multirc-em	38.0	48.0	54.0	34.0	42.0	46.0
squad_v2-em	34.0	26.0	26.0	34.0	24.0	30.0
squad_v1-em	60.0	68.0	62.0	60.0	68.0	48.0
openbookqa-em	68.0	76.0	68.0	68.0	72.0	70.0
bool_q-em	64.0	88.0	82.0	72.0	84.0	78.0
drop-em	20.0	28.0	18.0	20.0	28.0	8.0
CLOSE-BOOK QA						
natural_questions-em	16.0	24.0	14.0	14.0	22.0	16.0
arc_easy-em	94.0	94.0	88.0	94.0	94.0	80.0
arc_challenge-em	58.0	70.0	54.0	62.0	64.0	52.0
trivia_qa-em	68.0	70.0	62.0	68.0	70.0	58.0
COREFERENCE						
definite_pronoun_resolution-em	54.0	76.0	64.0	62.0	70.0	70.0
wsc-em	36.0	46.0	42.0	34.0	52.0	42.0
READ. COMP. W/ COM						
cosmos_qa-em	42.0	82.0	64.0	42.0	70.0	52.0
record-em	18.0	36.0	14.0	18.0	34.0	14.0
PARAPHRASE						
paws_wiki-em	42.0	62.0	52.0	42.0	68.0	44.0
glue_qqp-em	42.0	58.0	56.0	34.0	68.0	42.0
glue_mrpc-em	56.0	64.0	58.0	56.0	62.0	60.0
stsb-em	14.0	26.0	12.0	14.0	20.0	4.0
NLI						
cb-em	48.9	82.2	75.6	44.4	86.7	57.8
wnli-em	44.0	54.0	68.0	44.0	60.0	44.0
anli_r1-em	46.0	48.0	36.0	46.0	48.0	34.0
anli_r2-em	30.0	50.0	34.0	32.0	42.0	34.0
anli_r3-em	30.0	50.0	32.0	30.0	50.0	30.0
mnli_matched-em	60.0	92.0	86.0	64.0	92.0	86.0
mnli_mismatched-em	64.0	94.0	74.0	68.0	94.0	78.0
snli-em	52.0	94.0	64.0	52.0	94.0	58.0
qnli-em	50.0	70.0	58.0	54.0	72.0	58.0
rte-em	60.0	68.0	70.0	62.0	68.0	72.0

Table 16: Per-task results of LoRAuter for Llama2-7B with two large adapter pools.

	HF+48			HF	
Domain-Metric	Non-OOD	Semi-OOD	OOD	Semi-OOD	OOD
Struct to Text					
web_nlg_en-rouge-1	63.6	52.7	50.4	32.3	46.8
web_nlg_en-rouge-2	42.4	29.4	27.6	17.4	23.7
web_nlg_en-rouge-l	57.5	48.2	45.6	29.7	43.3
dart-rouge-1	61.0	61.0	58.9	41.3	57.2
dart-rouge-2	35.7	35.6	33.7	20.9	31.2
dart-rouge-l	53.7	54.0	52.7	38.2	52.2
e2e_nlg-rouge-1	63.7	63.2	58.0	48.4	58.2
e2e_nlg-rouge-2	37.2	36.5	33.0	26.7	31.6
e2e_nlg-rouge-l	53.6	52.3	50.6	39.4	50.8
common_gen-rouge-1	47.0	28.1	31.0	22.9	32.5
common_gen-rouge-2	19.1	7.1	8.1	7.4	10.5
common_gen-rouge-l	43.0	24.3	26.6	20.6	27.6
Translation					
para_crawl_enes-bleu	23.5	16.1	22.3	8.3	23.4
wmt16_translate_tren-bleu	2.0	2.0	3.2	1.0	2.3
wmt16_translate_deen-bleu	19.4	17.8	19.9	9.1	16.0
wmt16_translate_ruen-bleu	11.6	8.1	8.7	2.0	10.5
wmt16_translate_fien-bleu	6.2	7.2	6.9	2.5	5.0
wmt16_translate_roen-bleu	13.3	14.1	11.0	4.7	10.9
wmt14_enfr-bleu	16.0	16.2	17.9	9.8	17.1
wmt16_translate_csen-bleu	6.9	5.6	4.3	4.5	6.7
COMMONSENSE					
story_cloze-em	96.0	90.0	88.0	94.0	92.0
piqa-em	44.0	40.0	40.0	38.0	38.0
copa-em	82.0	82.0	82.0	80.0	80.0
hellaswag-em	50.0	50.0	46.0	42.0	52.0
SENTIMENT					
sst2-em	98.0	98.0	98.0	94.0	92.0
yelp_polarity_reviews-em	98.0	98.0	98.0	98.0	96.0
imdb_reviews-em	98.0	98.0	100.0	98.0	100.0
sentiment140-em	68.0	68.0	70.0	64.0	68.0
READING Comp.					
multirc-em	62.0	56.0	42.0	54.0	50.0
squad_v2-em	54.0	20.0	18.0	32.0	32.0
squad_v1-em	74.0	72.0	66.0	54.0	66.0
openbookqa-em	80.0	78.0	76.0	66.0	66.0
bool_q-em	78.0	80.0	72.0	82.0	76.0
drop-em	22.0	14.0	16.0	10.0	16.0
CLOSE-BOOK QA					
natural_questions-em	16.0	12.0	12.0	4.0	10.0
arc_easy-em	78.0	78.0	84.0	70.0	70.0
arc_challenge-em	50.0	48.0	46.0	38.0	38.0
trivia_qa-em	50.0	46.0	46.0	36.0	48.0
COREFERENCE					
definite_pronoun_resolution-em	62.0	64.0	62.0	70.0	66.0
wsc-em	58.0	54.0	46.0	58.0	58.0
READ. COMP. W/ COM					
cosmos_qa-em	72.0	54.0	62.0	52.0	46.0
record-em	64.0	34.0	30.0	22.0	22.0
PARAPHRASE					
paws_wiki-em	60.0	66.0	52.0	66.0	54.0
glue_qqp-em	80.0	78.0	76.0	82.0	52.0
glue_mrpc-em	66.0	66.0	62.0	50.0	34.0
stsb-em	38.0	18.0	20.0	12.0	14.0
NLI					
cb-em	88.9	84.4	84.4	86.7	84.4
wnli-em	52.0	52.0	54.0	62.0	64.0
anli_r1-em	54.0	50.0	46.0	52.0	52.0
anli_r2-em	44.0	40.0	40.0	40.0	38.0
anli_r3-em	44.0	54.0	56.0	54.0	48.0
mnli_matched-em	88.0	86.0	86.0	82.0	90.0
mnli_mismatched-em	94.0	94.0	92.0	98.0	98.0
snli-em	84.0	84.0	84.0	90.0	88.0
qnli-em	66.0	72.0	72.0	74.0	70.0
rte-em	84.0	66.0	66.0	64.0	64.0

Table 17: Per-task results for selection-based adapter routing methods on Llama2-7B, including oracle, performance-aligned, and input-aware selection. Results for LORARETRIEVER-based selection are directly copied from the original LORARETRIEVER paper.

LLAMA2-7B							
	ORACLE		NON-OOD		SEMI-OOD	OOD	
DOMAIN-METRIC	TASK-ALIGNED	PERF-ALIGNED	LORA-RETRIEVER SELECTION	LORAUTER SELECTION	LORAUTER SELECTION	LORA-RETRIEVER SELECTION	LORAUTER SELECTION
STRUCT TO TEXT							
WEB_NLG_EN-ROUGE-1	69.6	69.6	67.0	66.2	54.5	53.9	51.4
WEB_NLG_EN-ROUGE-2	48.0	48.0	44.5	44.8	30.0	30.0	27.8
WEB_NLG_EN-ROUGE-L	62.8	62.8	60.9	59.9	49.2	49.1	46.7
DART-ROUGE-1	71.7	71.7	67.9	67.6	60.2	58.4	51.3
DART-ROUGE-2	49.4	49.4	45.8	46.3	35.2	34.9	27.9
DART-ROUGE-L	64.9	64.9	61.1	61.8	53.5	52.4	46.3
E2E_NLG-ROUGE-1	66.1	66.1	65.8	63.9	52.2	59.3	41.5
E2E_NLG-ROUGE-2	39.6	39.6	39.4	37.4	27.6	34.1	19.6
E2E_NLG-ROUGE-L	56.4	56.4	55.7	54.2	46.2	50.2	36.5
COMMON_GEN-ROUGE-1	46.5	46.5	44.7	46.2	29.3	29.0	29.5
COMMON_GEN-ROUGE-2	18.9	18.9	18.3	18.9	7.5	7.3	7.8
COMMON_GEN-ROUGE-L	42.2	42.2	40.5	42.4	24.0	24.0	24.4
TRANSLATION							
PARA_CRAWL_ENES-BLEU	24.0	24.0	24.2	23.8	19.3	20.3	19.6
WMT16_TRANSLATE_TREN-BLEU	3.7	2.0	3.1	2.0	2.0	2.6	1.9
WMT16_TRANSLATE_DEEN-BLEU	17.8	17.8	10.4	17.8	16.9	9.8	16.3
WMT16_TRANSLATE_RUEN-BLEU	11.5	11.5	18.7	11.5	11.9	20.3	11.3
WMT16_TRANSLATE_FIEN-BLEU	6.8	8.0	6.5	8.0	8.0	7.0	7.4
WMT16_TRANSLATE_ROEN-BLEU	14.6	13.5	14.0	13.5	13.5	12.3	11.9
WMT14_ENFR-BLEU	16.0	17.0	16.1	17.0	17.0	16.9	18.0
WMT16_TRANSLATE_CSEN-BLEU	10.5	6.9	9.4	6.9	6.9	7.0	7.0
COMMONSENSE							
STORY_CLOZE-EM	72.0	96.0	62.0	96.0	96.0	42.0	84.0
PIQA-EM	44.0	44.0	46.0	44.0	34.0	32.0	36.0
COPA-EM	86.0	84.0	74.0	84.0	84.0	68.0	78.0
HELLASWAG-EM	50.0	50.0	40.0	50.0	48.0	42.0	50.0
SENTIMENT							
SST2-EM	98.0	98.0	98.0	98.0	96.0	96.0	96.0
YELP_POLARITY_REVIEWS-EM	98.0	98.0	94.0	98.0	98.0	94.0	100.0
IMDB_REVIEWS-EM	96.0	98.0	96.0	96.0	94.0	96.0	96.0
SENTIMENT140-EM	68.0	68.0	70.0	68.0	62.0	70.0	68.0
READING COMP.							
MULTIRC-EM	68.0	68.0	52.0	68.0	66.0	38.0	62.0
SQUAD_V2-EM	62.0	62.0	56.0	62.0	20.0	12.0	14.0
SQUAD_V1-EM	68.0	68.0	66.0	68.0	72.0	68.0	62.0
OPENBOOKQA-EM	84.0	84.0	68.0	76.0	66.0	58.0	76.0
BOOL_Q-EM	84.0	84.0	60.0	80.0	78.0	60.0	72.0
DROP-EM	40.0	40.0	8.0	28.0	20.0	6.0	6.0
CLOSE-BOOK QA							
NATURAL_QUESTIONS-EM	18.0	18.0	16.0	16.0	10.0	10.0	10.0
ARC_EASY-EM	48.0	74.0	56.0	74.0	66.0	70.0	74.0
ARC_CHALLENGE-EM	46.0	46.0	42.0	46.0	40.0	46.0	46.0
TRIVIA_QA-EM	64.0	64.0	46.0	44.0	44.0	46.0	34.0
COREFERENCE							
DEFINITE_PRONOUN_RESOLUTION-EM	52.0	64.0	50.0	64.0	64.0	50.0	50.0
WSC-EM	50.0	54.0	50.0	54.0	54.0	42.0	54.0
READ. COMP. W/ COM							
COSMOS_QA-EM	70.0	70.0	68.0	64.0	50.0	34.0	18.0
RECORD-EM	70.0	70.0	70.0	54.0	22.0	26.0	20.0
PARAPHRASE							
PAWS_WIKI-EM	90.0	90.0	64.0	60.0	52.0	40.0	50.0
GLUE_QQP-EM	70.0	82.0	74.0	82.0	82.0	68.0	78.0
GLUE_MRPC-EM	60.0	70.0	58.0	66.0	72.0	58.0	58.0
STSB-EM	34.0	34.0	36.0	34.0	18.0	16.0	26.0
NLI							
CB-EM	88.9	88.9	80.0	86.7	75.6	62.2	57.8
WNLI-EM	72.0	64.0	68.0	64.0	64.0	46.0	46.0
ANLI_R1-EM	52.0	52.0	50.0	48.0	46.0	50.0	44.0
ANLI_R2-EM	46.0	38.0	46.0	42.0	38.0	46.0	46.0
ANLI_R3-EM	46.0	38.0	42.0	44.0	44.0	38.0	54.0
MNLI_MATCHED-EM	88.0	88.0	84.0	88.0	88.0	88.0	88.0
MNLI_MISMATCHED-EM	92.0	92.0	90.0	92.0	84.0	94.0	94.0
SNLI-EM	96.0	84.0	84.0	84.0	82.0	84.0	82.0
QNLI-EM	94.0	94.0	94.0	88.0	68.0	26.0	64.0
RTE-EM	56.0	86.0	62.0	80.0	86.0	72.0	64.0

Table 18: Per-task results for selection-based adapter routing methods on Llama2-13B, including oracle, performance-aligned, and input-aware selection. Results for LORARETRIEVER-based selection are directly copied from the original LORARETRIEVER paper.

LLAMA2-7B							
	ORACLE		NON-OOD		SEMI-OOD	OOD	
DOMAIN-METRIC	TASK-ALIGNED	PERF-ALIGNED	LORA-RETRIEVER SELECTION	LORAUTER SELECTION	LORAUTER SELECTION	LORA-RETRIEVER SELECTION	LORAUTER SELECTION
STRUCT TO TEXT							
WEB_NLG_EN-ROUGE-1	72.4	72.4	68.5	68.1	56.7	51.9	51.1
WEB_NLG_EN-ROUGE-2	52.1	52.1	47.5	48.8	33.0	28.6	29.4
WEB_NLG_EN-ROUGE-L	66.1	66.1	62.4	62.2	50.8	48.3	46.4
DART-ROUGE-1	72.9	72.9	67.0	69.4	59.7	57.0	51.3
DART-ROUGE-2	52.9	52.9	45.9	49.5	35.3	33.6	29.2
DART-ROUGE-L	66.7	66.7	61.2	62.9	53.5	50.0	45.6
E2E_NLG-ROUGE-1	66.5	66.5	66.1	64.3	57.8	59.2	44.4
E2E_NLG-ROUGE-2	38.8	38.8	39.3	36.6	31.0	32.8	20.5
E2E_NLG-ROUGE-L	56.4	56.4	56.4	54.3	45.9	48.9	37.9
COMMON_GEN-ROUGE-1	48.4	48.4	48.9	48.6	27.6	29.3	29.5
COMMON_GEN-ROUGE-2	17.9	17.9	20.3	17.9	7.8	8.2	8.6
COMMON_GEN-ROUGE-L	44.1	44.1	44.1	44.5	25.7	24.4	25.5
TRANSLATION							
PARA_CRAWL_ENES-BLEU	25.0	25.0	25.4	24.9	22.1	23.1	23.1
WMT16_TRANSLATE_TREN-BLEU	2.0	3.2	2.4	3.2	3.2	1.2	2.5
WMT16_TRANSLATE_DEEN-BLEU	19.8	19.8	11.5	19.8	20.7	10.3	18.0
WMT16_TRANSLATE_RUEN-BLEU	11.3	10.9	19.9	10.9	10.9	20.7	11.0
WMT16_TRANSLATE_FIEN-BLEU	6.8	7.3	6.8	7.3	7.3	5.1	6.6
WMT16_TRANSLATE_ROEN-BLEU	14.2	10.2	13.9	10.2	10.2	10.9	12.9
WMT14_ENFR-BLEU	16.5	20.4	17.1	20.4	20.4	18.4	18.8
WMT16_TRANSLATE_CSEN-BLEU	6.9	8.5	6.2	8.5	8.5	11.6	9.2
COMMONSENSE							
STORY_CLOZE-EM	94.0	94.0	80.0	90.0	90.0	56.0	96.0
PIQA-EM	48.0	44.0	52.0	44.0	44.0	30.0	40.0
COPA-EM	76.0	78.0	74.0	78.0	78.0	68.0	78.0
HELLASWAG-EM	58.0	58.0	30.0	52.0	44.0	36.0	48.0
SENTIMENT							
SST2-EM	98.0	98.0	98.0	98.0	98.0	98.0	100.0
YELP_POLARITY_REVIEWS-EM	98.0	98.0	98.0	98.0	98.0	100.0	100.0
IMDB_REVIEWS-EM	96.0	96.0	98.0	98.0	98.0	98.0	98.0
SENTIMENT140-EM	68.0	68.0	68.0	66.0	66.0	68.0	70.0
READING COMP.							
MULTIRC-EM	88.0	88.0	72.0	86.0	74.0	36.0	40.0
SQUAD_V2-EM	68.0	68.0	62.0	68.0	40.0	62.0	14.0
SQUAD_V1-EM	74.0	74.0	58.0	68.0	70.0	34.0	52.0
OPENBOOKQA-EM	86.0	86.0	80.0	78.0	74.0	72.0	70.0
BOOL_Q-EM	84.0	84.0	68.0	82.0	76.0	66.0	80.0
DROP-EM	58.0	58.0	22.0	52.0	30.0	18.0	20.0
CLOSE-BOOK QA							
NATURAL_QUESTIONS-EM	30.0	30.0	28.0	24.0	18.0	12.0	14.0
ARC_EASY-EM	94.0	94.0	90.0	92.0	84.0	88.0	86.0
ARC_CHALLENGE-EM	68.0	62.0	66.0	70.0	66.0	58.0	68.0
TRIVIA_QA-EM	68.0	70.0	56.0	58.0	64.0	54.0	58.0
COREFERENCE							
DEFINITE_PRONOUN_RESOLUTION-EM	88.0	88.0	90.0	88.0	74.0	72.0	62.0
WSC-EM	62.0	64.0	60.0	64.0	64.0	58.0	48.0
READ. COMP. W/ COM							
COSMOS_QA-EM	84.0	84.0	82.0	74.0	74.0	38.0	30.0
RECORD-EM	78.0	78.0	78.0	60.0	34.0	28.0	38.0
PARAPHRASE							
PAWS_WIKI-EM	92.0	92.0	70.0	62.0	56.0	52.0	50.0
GLUE_QQP-EM	86.0	86.0	88.0	82.0	84.0	76.0	80.0
GLUE_MRPC-EM	84.0	84.0	70.0	80.0	70.0	62.0	62.0
STSB-EM	44.0	44.0	44.0	44.0	28.0	20.0	22.0
NLI							
CB-EM	95.6	95.6	91.1	95.6	91.1	84.4	91.1
WNLI-EM	72.0	72.0	72.0	72.0	58.0	62.0	52.0
ANLI_R1-EM	70.0	74.0	70.0	70.0	54.0	70.0	70.0
ANLI_R2-EM	66.0	66.0	56.0	66.0	38.0	56.0	64.0
ANLI_R3-EM	68.0	68.0	56.0	60.0	64.0	56.0	60.0
MNLI_MATCHED-EM	86.0	88.0	90.0	88.0	88.0	88.0	88.0
MNLI_MISMATCHED-EM	90.0	90.0	90.0	90.0	88.0	94.0	90.0
SNLI-EM	90.0	90.0	88.0	90.0	82.0	88.0	88.0
QNLI-EM	94.0	94.0	94.0	84.0	66.0	30.0	58.0
RTE-EM	88.0	88.0	82.0	80.0	66.0	74.0	74.0

Table 19: Per-task results for the LORARETRIEVER weighted mixture routing method on Llama2-7B and Llama2-13B under Non-OOD and OOD evaluation.

	LLAMA2-7B		LLAMA2-13B	
DOMAIN-METRIC	NON-OOD	OOD	NON-OOD	OOD
STRUCT TO TEXT				
WEB_NLG_EN-ROUGE-1	60.9	48.5	68.6	52.2
WEB_NLG_EN-ROUGE-2	36.7	26.5	47.2	28.5
WEB_NLG_EN-ROUGE-L	54.0	43.8	61.6	46.7
DART-ROUGE-1	62.7	58.0	64.1	59.2
DART-ROUGE-2	36.7	33.4	39.9	36.1
DART-ROUGE-L	55.9	52.3	55.7	52.2
E2E_NLG-ROUGE-1	64.3	55.5	64.2	61.8
E2E_NLG-ROUGE-2	37.8	29.8	36.6	34.4
E2E_NLG-ROUGE-L	54.4	47.9	54.4	52.1
COMMON_GEN-ROUGE-1	50.1	29.8	46.1	28.2
COMMON_GEN-ROUGE-2	23.3	9.1	21.2	9.6
COMMON_GEN-ROUGE-L	46.5	26.2	42.6	24.4
TRANSLATION				
PARA_CRAWL_ENES-BLEU	23.4	23.0	25.7	23.6
WMT16_TRANSLATE_TREN-BLEU	3.2	2.6	2.5	2.8
WMT16_TRANSLATE_DEEN-BLEU	20.0	19.5	20.8	20.8
WMT16_TRANSLATE_RUEN-BLEU	10.8	8.5	10.3	11.0
WMT16_TRANSLATE_FIEN-BLEU	6.5	7.1	6.8	8.3
WMT16_TRANSLATE_ROEN-BLEU	14.6	13.3	15.8	16.2
WMT14_ENFR-BLEU	17.3	18.2	15.7	18.8
WMT16_TRANSLATE_CSEN-BLEU	10.0	5.8	9.4	9.9
COMMONSENSE				
STORY_CLOZE-EM	80.0	88.0	96.0	94.0
PIQA-EM	42.0	36.0	52.0	48.0
COPA-EM	84.0	78.0	80.0	80.0
HELLASWAG-EM	34.0	30.0	52.0	48.0
SENTIMENT				
SST2-EM	98.0	94.0	98.0	100.0
YELP_POLARITY_REVIEWS-EM	98.0	98.0	100.0	100.0
IMDB_REVIEWS-EM	98.0	100.0	98.0	98.0
SENTIMENT140-EM	68.0	72.0	72.0	72.0
READING COMP.				
MULTIRC-EM	62.0	46.0	78.0	40.0
SQUAD_V2-EM	56.0	14.0	66.0	20.0
SQUAD_V1-EM	64.0	64.0	68.0	62.0
OPENBOOKQA-EM	74.0	70.0	88.0	80.0
BOOL_Q-EM	80.0	78.0	84.0	86.0
DROP-EM	20.0	14.0	38.0	30.0
CLOSE-BOOK QA				
NATURAL_QUESTIONS-EM	12.0	10.0	28.0	18.0
ARC_EASY-EM	74.0	80.0	94.0	90.0
ARC_CHALLENGE-EM	46.0	46.0	80.0	72.0
TRIVIA_QA-EM	56.0	50.0	62.0	56.0
COREFERENCE				
DEFINITE_PRONOUN_RESOLUTION-EM	60.0	64.0	92.0	68.0
WSC-EM	56.0	50.0	70.0	56.0
READ. COMP. W/ COM				
COSMOS_QA-EM	74.0	48.0	82.0	62.0
RECORD-EM	58.0	24.0	76.0	50.0
PARAPHRASE				
PAWS_WIKI-EM	52.0	46.0	88.0	62.0
GLUE_QQP-EM	86.0	60.0	82.0	82.0
GLUE_MRPC-EM	62.0	64.0	72.0	64.0
STSB-EM	50.0	12.0	44.0	24.0
NLI				
CB-EM	82.2	66.7	97.8	86.7
WNLI-EM	66.0	44.0	74.0	64.0
ANLI_R1-EM	52.0	44.0	72.0	68.0
ANLI_R2-EM	48.0	48.0	60.0	64.0
ANLI_R3-EM	48.0	46.0	64.0	60.0
MNLI_MATCHED-EM	84.0	80.0	86.0	84.0
MNLI_MISMATCHED-EM	90.0	84.0	94.0	92.0
SNLI-EM	86.0	84.0	92.0	88.0
QNLI-EM	80.0	74.0	80.0	72.0
RTE-EM	74.0	68.0	80.0	70.0