Empirical Guidelines for Deploying LLMs onto Resource-constrained Edge Devices

RUIYANG QIN, ECE, Villanova University College of Engineering, Villanova, United States
DANCHENG LIU, University at Buffalo, Buffalo, United States
CHENHUI XU, University at Buffalo, Buffalo, United States
ZHEYU YAN, University of Notre Dame, Notre Dame, United States
ZHAOXUAN TAN, University of Notre Dame, Notre Dame, United States
ZHENGE JIA, University of Notre Dame, Notre Dame, United States
AMIR NASSERELDINE, University at Buffalo, Buffalo, United States
JIAJIE LI, University at Buffalo, Buffalo, United States
MENG JIANG, Computer Science and Engineering, University of Notre Dame, Notre Dame, United States
AHMED ABBASI, University of Notre Dame, Notre Dame, United States
JINJUN XIONG, University at Buffalo, Buffalo, United States
YIYU SHI, Electrical and Computer Engineering, University of Notre Dame, Notre Dame, United States

The scaling laws have become the de facto guidelines for designing large language models (LLMs), but they were studied under the assumption of unlimited computing resources for both training and inference. As LLMs are increasingly used as personalized intelligent assistants, their customization (i.e., learning through fine-tuning) and deployment onto resource-constrained edge devices will become more and more prevalent. An urgent but open question is how a resource-constrained computing environment would affect the design choices for a personalized LLM. We study this problem empirically in this work. In particular, we consider the tradeoffs among a number of key design factors and their intertwined impacts on learning efficiency and accuracy. The factors include the learning methods for LLM customization, the amount of personalized data used for learning customization, the types and sizes of LLMs, the compression methods of LLMs, the amount of time afforded to learn, and the difficulty levels of the target use cases. Through extensive experimentation and benchmarking, we draw a number of surprisingly insightful guidelines for deploying LLMs onto resource-constrained devices. For example, an optimal choice between parameter learning and RAG may

Ruiyang Qin and Dancheng Liu contributed equally to this research.

Authors' Contact Information: Ruiyang Qin, ECE, Villanova University College of Engineering, Villanova, Pennsylvania, United States; e-mail: ruiyangqin2021@gmail.com; Dancheng Liu, University at Buffalo, Buffalo, New York, United States; e-mail: dliu37@buffalo.edu; Chenhui Xu, University at Buffalo, Buffalo, New York, United States; e-mail: cxu26@buffalo.edu; Zheyu Yan, University of Notre Dame, Notre Dame, Indiana, United States; e-mail: yanzheyu@zju.edu.cn; Zhaoxuan Tan, University of Notre Dame, Notre Dame, Indiana, United States; e-mail: ztan3@nd.edu; Zhenge Jia, University of Notre Dame, Notre Dame, Indiana, United States; e-mail: zjia2@nd.edu; Amir Nassereldine, University at Buffalo, Buffalo, New York, United States; e-mail: amirnass@buffalo.edu; Jiajie Li, University at Buffalo, Buffalo, New York, United States; e-mail: jli433@buffalo.edu; Meng Jiang, Computer Science and Engineering, University of Notre Dame, Notre Dame, Indiana, United States; e-mail: mjiang2@nd.edu; Ahmed Abbasi, University of Notre Dame, Notre Dame, Indiana, United States; e-mail: aabbasi@nd.edu; jinjun xiong, University at Buffalo, Buffalo, New York, United States; e-mail: jinjun@buffalo.edu; Yiyu Shi, Electrical and Computer Engineering, University of Notre Dame, Notre Dame, Indiana, United States; e-mail: yshi4@nd.edu.

ACM 1084-4309/2025/08-ART70
https://doi.org/10.1145/3736721

vary depending on the difficulty of the downstream task, the longer fine-tuning time does not necessarily help the model, and a compressed LLM may be a better choice than an uncompressed LLM to learn from limited personalized data.

CCS Concepts: • **Human-centered computing** → **Empirical studies in ubiquitous and mobile computing;**

Additional Key Words and Phrases: on-device learning, large-language models, empirical study, AI personalization

ACM Reference Format:
Ruiyang Qin, Dancheng Liu, Chenhui Xu, Zheyu Yan, Zhaoxuan Tan, Zhenge Jia, Amir Nassereldine, Jiajie Li, Meng Jiang, Ahmed Abbasi, jinjun xiong, and Yiyu Shi. 2025. Empirical Guidelines for Deploying LLMs onto Resource-constrained Edge Devices. *ACM Trans. Des. Autom. Electron. Syst.* 30, 5, Article 70 (August 2025), 58 pages. https://doi.org/10.1145/3736721

1 Introduction

The world has witnessed the growing interest in applying **Large Language Models** (**LLMs**) as a potential solution to personalized AI assistants [48, 68]. This is particularly true when LLMs are deployed onto edge devices (edge LLMs) to meet the increasing needs of people's daily life assistance [5, 53], personalized companionship [19, 67], and real-time work assistance [18], where data privacy is of high priority [86] and constant internet connections may not be possible [46]. Some exemplar edge LLMs include *LLM on Nvidia IGX*, *Chat with RTX*, and *TinyChat*. Edge LLMs can keep locally generated data from users and learn from that data locally. Such a high assurance of privacy coupled with LLMs' strong reasoning capabilities can induce even more user interaction with the personal assistant at a deeper personal level than otherwise [25, 65].

The scaling law [35] has emerged as the standard for creating LLMs, but a key assumption behind it is the reliance on unlimited computing power for both training and inference. However, the design of edge LLMs is no easy feat [47] because of the limited resources available on edge devices. It requires allocating limited resources to accommodate multiple needs [61, 84]. An ideal resource allocation strategy needs to balance many key, yet sometimes conflicting, factors, such as the types and sizes of pre-trained LLMs, the learning methods and hyper-parameters for LLM customization, the amount of historical data used for learning personalization, the compression methods of LLMs, the amount of time afforded to learn, and the difficulty levels of the target user cases. For example, models can be selected from a wide range of candidates with different model structures and sizes; while learning methods include **parameter-efficient fine-tuning** (**PEFT**) and **retrieval-augmented generation** (**RAG**). A poorly designed edge LLM may render a poor user experience as ~~it cannot learn well from user locally-generated content.~~ As such, a pressing yet unresolved issue is how those constraints of limited computing resources influence the design decisions for personalized LLMs on edge devices, and what principles should guide the deployment of these edge LLMs. In this work, we empirically investigate this problem, focusing on the tradeoffs between key design factors and their combined effects on learning efficiency and accuracy.

We formalize our empirical design guidelines for edge LLMs through extensive experimentation and benchmarking as shown in Figure 1, covering a wide range of possibilities for the key factors and their combinations. We draw a number of surprisingly insightful guidelines for deploying LLMs onto resource-constrained devices. These guidelines will be elaborated in six parts in Section 3. As a heads-up, a high-level summary of some interesting guidelines is as follows:

— The ~~difficulty of the downstream task is a main factor~~ for choosing the optimal edge LLM types and the learning method. Tasks that are either too easy or too hard tend to favor parameter learning, while medicore tasks tend to favor RAG.

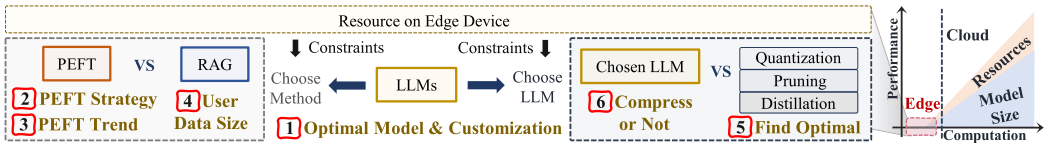


Fig. 1. Overview of our empirical study, filling the missing part of scaling law cycled by pink dash-line in right coordinate graph. Each numbered annotation in red corresponds to an empirical guideline, detailed in the respective subsections of Section 3.

— More historical user data for training is not always better. In most cases, the fine-tuning process does not necessarily need to use all the data and should stop early, especially when the model converges to some stable position (even at a very shallow local optimum).
— Among the three most popular model compression methods, distillation provides the most stable performance, quantization is less stable but has the highest peak performance, while pruning is not suitable for the edge LLMs.

To the best of our knowledge, this work is the first exploratory study that tries to address the full range of constraints on deploying LLMs on edge devices. By systematically examining the tradeoffs among various factors, we offer guidelines and insights for the community and future research in this field. We hope that this work will both increase awareness of the limitations that LLMs will encounter in future edge deployments and shed light on the opportunities for future LLM designs.

Paper Organization: The organization of this article is as follows. Following the Introduction, we provide background on our investigation, including device constraints, models, datasets, customization methods, and experimental settings. The **core** of this article is Section 3, which follows the diagram shown in Figure 1. In Section 3, each subsection corresponds to the numbered order in Figure 1. Notably, at the end of each subsection, we highlight a remark that emphasizes the findings and outlines potential directions for future research. In Section 4, we present our conclusions and summarize directions for future work, along with a high-level table of our empirical guidelines (Table 7). In the Appendix, we include additional experimental results to complement the main sections.

2 Preparation for Evaluation

In this section, we present a comprehensive evaluation framework for edge LLM customization. First, we examine the key constraints of edge devices that impact LLM deployment, followed by our methodology for selecting and quantifying specific resource constraints for our experiments. We then detail the datasets used to evaluate edge LLM customization for personalization. With the experimental environment established, we present our rationale for selecting a diverse range of LLMs suitable for edge deployment, including a detailed analysis of their characteristics. The section continues with an exploration of various compression methods essential for adapting LLMs to edge devices. We then discuss our customization approach for enhancing edge LLM personalization for individual users. The section concludes with our default experimental settings and additional background information about LLMs to provide the necessary context for our study.

2.1 Edge Device Constraints

Overview. Clusters and cloud servers that provide **computing-as-a-service** (**CaaS**) consist of millions of computation units like GPU with nearly unlimited RAM, storage, and energy [62, 63].

Table 1. Profile of Five Common Edge Devices Evaluated in This Article

Device	RAM (GB)	CPU Configuration	GPU Configuration
iPhone 12 (A14)	4	6-core	4-core
iPhone 15 (A16)	6	6-core	5-core
NVIDIA Jetson Orin Nano	8	6-core ARM Cortex-A78AE	1024-core NVIDIA Ampere
Samsung Galaxy S24 Ultra	10	Octa-core	Adreno GPU
NVIDIA Jetson Orin NX	16	6-core ARM Cortex-A78AE	1024-core NVIDIA Ampere

Common edge devices such as smartphones, on the other hand, are usually equipped with a multi-core CPU with limited resources. Emerging technologies are beginning to integrate GPUs into edge devices, exemplified by the Jetson Orin or phones powered by the Qualcomm Snapdragon 8 Gen, which offer computational powers of 3.5 TFlops—nearly half that of a cluster-level GPU like the Nvidia RTX 2070. As computational power in edge devices increases, the feasibility of running edge LLMs is significantly enhanced.

In this article, corresponding to the evaluation preparation described in Section 2, we selected five commonly used edge devices with varying RAM capacities, as shown in Table 1. Besides computational power, RAM is a critical resource for enabling on-device LLM learning. We categorize the edge devices by their RAM capacity because it offers a more concise classification than using computational power. In Table 1, note that while the CPU and GPU configurations do not constrain on-device LLM deployment as much as RAM does, the CPU and GPU configurations across the five devices are quite similar. Therefore, we can safely ignore potential differences arising from them and focus on the constraints imposed by RAM.

2.2 Running LLMs on Edge Devices

We use data processing capability to quantify the computational power of each edge device [64, 66]. By deploying LLMs to each edge device and determining the optimal processing time per data sample, we can calculate how many data samples each edge device can process per hour for all selected LLMs. The evaluation results are shown in Table 2.

In addition to the two primary resource constraints—computational power and RAM, some constraints such as temperature and energy consumption may also be of concern. However, these are considered secondary issues. Without addressing computational power and RAM requirements, an LLM cannot be deployed on edge devices at all. Once an LLM is deployed, temperature and energy consumption are correlated with computational power and RAM usage. In our work, we focus on the primary constraints. Additionally, we assume the edge devices are charging, connected to a power source, and training the deployed LLMs while users are not using these devices. Hence, we can minimize the impact of temperature and power constraints.

To efficiently conduct comprehensive evaluations, we use a high-performance NVIDIA A10 GPU to simulate the training time on edge devices. We set the corresponding number of data samples processed by the A10 GPU to represent the amount of time required for training on an edge device. For example, the first entry in Table 2 indicates that 825 data samples can be processed when the Pythia-70m model is deployed on an edge device with 4GB RAM.

We calculate the number of data samples processed per hour using the corresponding throughput from training on both the A10 GPU and on edge devices. Since this work spans a wide range of LLMs, directly measuring each model's throughput on every edge device would be prohibitively time-consuming. Instead, we adopt a fixed conversion factor for one particular edge device, which is then applied to all models to approximate their training throughput on that device. Specifically,

Table 2. Selected Models with their Initial Model Weights Size and the Training Peak RAM

Total RAM	Pretrained LLM (ID from Hugging Face)	Training Peak memory (GB)	Weights Size (GB)	Data Samples per Hour
4G	EleutherAI/pythia-70m	1.15	0.17	825
	EleutherAI/pythia-160m	1.61	0.38	415
	EleutherAI/pythia-410m	2.69	0.91	215
	facebook/opt-125m	1.78	0.25	550
	facebook/opt-350m	2.25	0.66	320
	TheBloke/TinyLlama-1.1B-Chat-v0.3-GPTQ	1.72	0.77	110
	TheBloke/open-llama-3b-v2-GPTQ	3.39	2.09	95
6G	facebook/opt-1.3b	5.68	2.63	245
	TheBloke/stablelm-zephyr-3b-GPTQ	3.17	1.84	85
	TechxGenus/gemma-2b-GPTQ	5.31	2.08	145
	EleutherAI/pythia-1b	4.98	2.09	250
	TinyLlama/TinyLlama-1.1B-step-50K-105b	5.32	2.20	165
8G	microsoft/phi-1_5	6.76	2.84	153
	EleutherAI/pythia-1.4b	6.68	2.93	182
	TheBloke/Llama-2-7B-Chat-GPTQ	6.07	3.90	82
	TheBloke/Mistral-7B-v0.1-GPTQ	6.61	4.16	75
	TheBloke/Synthia-7B-v1.3-GPTQ	6.61	4.16	75
	TheBloke/openchat_3.5-GPTQ	6.16	4.16	76
10G	princeton-nlp/Sheared-LLaMA-1.3B-Pruned	6.89	2.69	222
	stabilityai/stablelm-2-1_6b	8.12	3.29	156
	TechxGenus/Meta-Llama-3-8B-GPTQ	8.59	5.74	85
	princeton-nlp/Sheared-LLaMA-1.3B	6.89	5.38	222
	facebook/opt-2.7b	6.89	5.30	120
	TechxGenus/gemma-7b-GPTQ	9.45	7.18	71
	TheBloke/Llama-2-13B-GPTQ	9.95	7.26	50
16G	microsoft/phi-2	11.97	5.00	94
	microsoft/Phi-3-mini-4k-instruct	13.57	7.64	84
	EleutherAI/pythia-2.8b	12.97	5.68	106
	google/gemma-2b	13.45	4.95	132
	princeton-nlp/Sheared-LLaMA-2.7B-Pruned	12.26	5.40	89
	stabilityai/stablelm-3b-4e1t	12.61	5.59	100
	openlm-research/open_llama_3b_v2	15.11	6.85	89
	princeton-nlp/Sheared-LLaMA-2.7B	12.26	9.95	89
	meta-llama/Meta-Llama-3-8B*	23.00	16.07	50

* for comparison with quantized Llama-3-8B

For the data size per hour, it indicates the estimated amount of data each model can learn or train from given one hour. NOTE: The peak memory usage for some models may approach the total available RAM. Although our measurements indicate that these models can run on devices with the specified RAM (for example, open-llama-3b-v2-GPTQ on a 4G RAM device), it is recommended to use devices with more RAM (such as 6G) to avoid potential unexpected "out of memory" errors.

we first measure the time required to train 100 to 500 samples on an A10 GPU for each LLM and use the average to obtain the GPU throughput (in tokens per second). Next, we determine a coefficient that converts from the A10 GPU to the target edge device by measuring the throughput for one model on the edge device. Finally, the number of data samples per hour is computed as: $\frac{3600 * \text{A10 GPU throughput}}{\text{edge device coefficient}}$.

This approach allows us to run experiments to assess how Pythia-70m performs given different training times by simply assigning different numbers of data samples to the model under various settings, such as different LoRA hyperparameters. It also closely simulates practical scenarios where a user employs an edge device with a specific LLM and wants the edge LLM to learn from available data.

2.3 Datasets

2.3.1 Background. To prepare for evaluation in Section 2, it is crucial to utilize appropriate datasets. The edge LLM is primarily employed for handling prompts made by a single user. Consequently, user-specific datasets are necessary. Such datasets should feature data that is correlated and includes user-specific information.

We have selected the LaMP datasets [72] to evaluate our edge LLM. In LaMP, each user has history data as shown in Table 4. The history data for each user consists of numerous utterances that include a label and textual content. The edge LLM will primarily learn from this historical data. The user prompts in the LaMP datasets are presented in Table 5. A well-trained edge LLM should be capable of accurately responding to the prompts using the history data.

The LaMP datasets consist of **seven datasets**. The first three datasets involve text classification, including LaMP-1 for citation identification, LaMP-2 for movie tagging, and LaMP-3 for product rating. The remaining four datasets pertain to text generation, including LaMP-4 for news headline generation, LaMP-5 for scholarly title generation, LaMP-6 for email subject generation, and LaMP-7 for tweet paraphrasing.

2.3.2 Data Preparation. As we mentioned in Section 2, we prepare the data for our edge LLM evaluation from two perspectives: user-level and task-level.

User-level: Each dataset contains many users, and each user has many samples of user history data and user query data. One sample of user history data and user query data are shown in Table 4 and in Table 5. In terms of generalization, we randomly select 100 users and calculate the average performance of edge LLMs over the 100 users in our evaluation.

Task-level: When evaluating various edge LLMs across multiple datasets (tasks), it is important to account for the latent factor of task difficulty, which can influence the evaluation results [21]. Within a single dataset, we assume that utterances exhibit similar levels of difficulty [3]. However, difficulty can vary significantly across different datasets, making it a critical consideration for accurate assessment. Previous research emphasizes the importance of considering task difficulty in evaluating machine learning models, as this factor can substantially impact performance comparisons across tasks and models. In particular, zero-shot learning has been used as an effective tool to estimate the inherent difficulty of datasets on strong pre-trained models, allowing for more refined assessments [70]. To this end, we evaluate the performance of models such as GPT-4 [1], Claude 3 Opus [36], Gemini 1.0 Pro [77], and Llama 3 70B [55] on selected test data from each dataset, with the results summarized in Table 3.

For classification tasks, where the number of choices varies across tasks, we adopt a more equitable assessment method inspired by Refs. [10, 14], which normalizes accuracy relative to human expert performance. Since obtaining human performance benchmarks on these datasets was not feasible, we instead normalized accuracy by dividing the LLMs' achieved accuracy by the accuracy expected from random guessing, a metric we refer to as "normalized accuracy." This is mathematically expressed as $\frac{\text{accuracy}}{1/\text{No. choices}}$. This approach allows us to address task difficulty disparities across datasets, as normalizing the model's accuracy against random guessing helps isolate performance differences related to the models themselves rather than the intrinsic difficulty of the task. Similar methods for task difficulty normalization have been discussed in literature [90], highlighting

Table 3. Difficulty (Weighted Average) Comparisons of Different Datasets

Dataset	GPT-4	Claude 3 Opus	Gemini 1.0 Pro	Llama 3 70B	Average	**Normalized Accuracy**	Task Type
LaMP-1 (Accuracy)	0.539	0.539	0.490	0.422	0.4975	0.995	Classification
LaMP-2 (Accuracy)	0.355	0.320	0.300	0.400	0.3438	5.157	Classification
LaMP-3 (Accuracy)	0.667	0.657	0.510	0.755	0.6472	3.236	Classification
LaMP-4 (ROUGE-1)	0.143	0.171	0.139	0.131	0.1460	0.1460	Generation
LaMP-5 (ROUGE-1)	0.386	0.374	0.405	0.084	0.3123	0.3123	Generation
LaMP-6 (ROUGE-1)	0.351	0.356	0.405	0.278	0.3475	0.3475	Generation
LaMP-7 (ROUGE-1)	0.326	0.136	0.237	0.255	0.2385	0.2385	Generation

The mean of four cloud-based LLMs' zero-shot performance can be used as an evaluator of task difficulty.

the importance of using a baseline such as random guessing or human experts when comparing performance across datasets of varying complexities.

For summarization tasks, normalized accuracy is equivalent to the raw accuracy, since random guessing does not apply in such contexts. This metric allows us to gauge task difficulty, where a higher normalized accuracy reflects a less challenging task.

Our analysis yields normalized accuracy scores for the classification datasets as follows: LaMP-1 (0.995), LaMP-2 (5.157), and LaMP-3 (3.236). For the summarization datasets, the normalized accuracies are: LaMP-4 (0.1460), LaMP-5 (0.3123), LaMP-6 (0.3475), and LaMP-7 (0.2385). Detailed results for each dataset across the various cloud-based LLMs are presented in Table 3.

2.4 Selected LLMs

In Section 2.1, we have explained the edge device constraints. As we mentioned in Section 2, we elaborate on selected LLMs in this section. In Table 1, we categorize five RAM sizes based on the common edge devices. To run pre-trained LLMs under such RAM constraints, we first need to ensure the size of these LLMs can fit into their corresponding RAM. To satisfy 4G RAM, we pick five models from the family of Pythia [6] and Llama [79]. To satisfy 6G RAM, we pick four models from the family of StableLM [4], Pythia [6], and Gemma [78]. To satisfy 8G RAM, we pick six models from the family of Phi [23], Pythia, Llama [79], Mistral [34], Synthia, and OpenChat. To satisfy 10G RAM, we pick five models from the family of Llama, StableLM, and Gemma. To satisfy 16G RAM, we pick seven models from the family of Phi, Pythia, Gemma, Llama, and StableLM. The detailed model descriptions can be found in Table 2. For each model we also make profiling to find its peak RAM usage, showing as "Training Peak RAM" on Table 2. [1] The model with smaller training peak RAM can also run on larger RAMs.

2.5 Model Compression

As we mentioned in Section 2, model compression is another component in our empirical study. The LLMs were originally designed and trained for cluster computing [63]. Their size can easily go beyond the size of edge device RAM size. Furthermore, the larger the model, the longer time it takes to train and make inferences [27]. Model compression can help such LLMs deploy on edge devices (edge LLMs) and improve their training and inference efficiency. There are three common implementation methods for model compression: quantization, pruning, and distillation.

[1]Phi-3 theoretically could fit in a 16G RAM during training, but typical optimizations such as HuggingFace's trainer require some extra RAM, which will cause overflow issues. Thus, we do not include Phi-3 in Table 2 but we still use the model for some of the experiments.

Quantization: The default precision for model weights in LLMs is FP32 or FP16. By reducing the weight precision from 32 bits down to 3 or 4 bits (quantization), we can significantly decrease the model size and inference time due to simpler gradient computations. However, reducing the number of bits in weights can compromise their precision and, consequently, model performance. The primary goal of quantization is to reduce weight bits without degrading performance. The main strategy for LLM quantization involves quantizing and then calibrating the pre-trained model, a technique known as **post-training quantization** (**PTQ**) [88]. Several implementations of PTQ for LLMs exist, including SmoothQuant [83], AWQ [45], LLM-QAT [51], and GPTQ [20].

For experimental consistency, in our experiments, we selected the widely-used GPTQ (W4A16), which has been applied to almost all LLMs and has proven effective. Furthermore, GPTQ has been specifically adapted for edge LLMs [74, 89].

Pruning: Different from quantization, pruning lightens the model by trimming off certain weights without lowering the model performance. Compared with quantization, pruning receives less attention in the beginning. One reason can be the high weight complexity of LLMs which makes it difficult to prune them while maintaining their performance. Furthermore, since the core modules of LLMs, the transformer blocks, make the structures of LLMs complex, pruning the LLMs can be more challenging than pruning on pure neural networks with simple structures. Within the existing pruning works including Wanda [75], LLM-Pruner [54], and Sheared-LLaMA [82], Sheared-LLaMA has demonstrated decent performance with published LLM weights.

To simplify the experiments and avoid potential issues with implementing Wanda or LLM-Pruner to prune LLMs, we use Sheared-LLaMA in our experiments.

Distillation: Other than quantization and pruning, another method involves using a smaller model to emulate and learn from a larger model, a process known as distillation. Microsoft introduced a **small language model** (**SLM**) called Phi, which leverages synthetic datasets specifically created by GPT-3.5 to learn domains such as common sense reasoning, general knowledge, science, daily activities, and theory of mind [23]. This approach represents a novel departure from traditional LLMs, allowing SLMs to acquire dense knowledge from LLMs while maintaining a smaller size. Although other works such as Distilling Step-by-Step [28], MetaIE [58], and MLFS [37] have proposed distillation methods, they have not yielded a well-trained model comparable to Phi, which benefited from Microsoft's extensive resources and data. Therefore, we have chosen to use Phi in our experiments.

2.6 Methods for edge LLM customization

In Section 2, we also mention the customization. There are two methods for edge LLM customization: PEFT [31] and RAG [9]. PEFT is an approach that improves edge LLM performance by tuning a small portion of the model's parameters. RAG, on the other hand, focuses on storing user history data and retrieving appropriate information to provide more context to user prompts. While both methods can generally enhance cloud-based LLM performance [38], their effectiveness can vary significantly depending on different settings and datasets. It is crucial to investigate whether similar variations in performance exist for edge LLMs.

PEFT: Among the various PEFT implementations, low-rank adaptation (LoRA) has demonstrated promising performance [29] in fine-tuning a wide range of LLMs by updating only a small fraction of the parameters. Notably, the hyperparameters *rank* and *alpha* have the most significant impact on the performance of LoRA-tuned models [39]. Therefore, we keep all other hyperparameters at their default settings (see Section 2.7) and systematically explore a wide range of rank and alpha combinations, along with their corresponding numbers of trainable parameters, to evaluate the performance of the edge LLM. In addition to LoRA, we also include three other widely used

PEFT methods: **Prefix Tuning** (**PfixT**), **Prompt Tuning** (**PmptT**), and IA3. Other methods, such as LoHA [33], are excluded due to their limited usage and lack of software optimization.

RAG: RAG consists of a retriever that is commonly based on **max inner product search** (**MIPS**) and a generator, which usually is an LLM. The retriever gets the appropriate query-relevant information from user history data to formalize the final prompt [41]. While this method does not consume resources to fine-tune the model, it requires saving all user history data embeddings for information retrieval. As user history data accumulates, the data embeddings can become a significant burden on the edge device [68]. In our experiments, we focus on the size of user history data and set all other hyperparameters to default, as shown in Section 2.7.

Our experiments investigate how different settings in each method can impact model performance and study which method can outperform the other under certain circumstances.

2.7 Experimental Settings

In response to the mentioned experimental setting in Section 2, we provide detailed experimental settings in this section. For each subsec:Datasets]LaMP dataset, we analyze 100 users, each containing up to 1,000 documents in their user history data. Each document encapsulates a single piece of user information, as illustrated in Table 4. Unless specifically stated otherwise, we set the temperature to 0.1, top_p to 0.9, max_new_tokens to 100, and top_k to 10 for content generation by the LLM.

Retrieval-Augmented Inference (RAG). RAG operates with two main parts: a retriever that obtains the user-specific documents from his historical data using MIPS, and a generator backed by an LLM. This generator takes both the user query and the retrieved document as inputs to create an informative prompt and generate the corresponding content. For MIPS to function effectively, all history documents must be converted into embeddings via a sentence embedding model. In our experiments, we use *all-MiniLM-L6-v2* [32]. We select the highest-ranked data as the output in MIPS, setting the top_k parameter to 3.

Parameter-Efficient Fine-Tuning (PEFT). We selected LoRA [29], prefix tuning [43], prompt tuning [40], and IA3 [49] as the PEFT implementations in our study. For prompt tuning and prefix tuning, we set the virtual token size to 20. In the case of LoRA and IA3, we enabled fine-tuning of parameters for the *query*, *key*, and *value* layers. We configured the dropout rate for LoRA at *0.1*. The initial learning rate was set to *5e-4*, and we employed a linear learning rate scheduler to optimize it. For LoRA, we utilized the *AdamW* optimizer. Furthermore, we designated the fine-tuning task type as *CAUSAL_LM*. To explore a range of trainable parameter sizes, we established wide ranges for both rank and alpha, as these directly influence the number of trainable parameters. The rank was set to increment from *8* to *256*, while alpha ranged from *8* to *32*. This approach allowed us to investigate various trainable parameter configurations.

3 Results Analysis and Empirical Guidelines

In this section, we present analyses of our experimental results and explain how these findings inform our empirical guidelines for edge LLMs. Additionally, we provide a perspective **remark** at the end of each subsection to highlight the significant finding and future potential research question. Due to space limitations, some of the detailed experimental results are placed in Appendix A and Appendix B.

For the datasets we used, their difficulties can be ranked as LaMP-2 < LaMP-3 < LaMP-1 in classification task type, and LaMP-6 < LaMP-5 < LaMP-7 < LaMP-4 in generation task type. The details are included in Appendix 2.3. In addition, we find that models often exceed human performance on classification tasks [22], whereas summarization tasks remain an open problem [87]. Thus, we argue that classification tasks are generally easier than summarization tasks. In the fol-

Table 4. One Sample of User History Data in Each Dataset

Dataset		User History Data
LaMP-1	[label] title abstract	"DSP architectures: past, present and futures" "As far as the future of communication is concerned, we have seen that there is great demand for audio and video data to complement text. Digital signal processing (DSP) is the science that enables traditionally analog audio and video signals to be processed digitally for transmission, storage, reproduction and manipulation. In this paper, we will explain the various DSP architectures and its silicon implementation. We will also discuss the state-of-the art and examine the issues pertaining to performance."
LaMP-2	[label] tag description	"classic" "Young Dorothy finds herself in a magical world where she makes friends with a lion, a scarecrow and a tin man as they make their way along the yellow brick road to talk with the Wizard and ask for the things they miss most in their lives. The Wicked Witch of the West is the only thing that could stop them."
LaMP-3	[label] score text	"4" "Amazing story of love that overcomes many obstacles. This book demonstrates that many of us are imprisoned by our views that have developed due to our upbringing, our culture, our environment and that it is possible to work through these problems of prejudice."
LaMP-4	[label] title text	"Five Things Women Can Do Today to Move Past Divorce" "I know it sounds trite, but now you have the freedom to be you and to let it reflect in your home, surroundings and daily activities. Embrace it, run with it and most of all enjoy it."
LaMP-5	[label] title abstract	"Performability Studies of Hypercube Architectures" "The authors propose a novel technique to study composite reliability and performance (performability) measures of hypercube systems using generalized stochastic Petri nets (GSPNs). This technique essentially consists of the following: (i) a GSPN reliability model; (ii) a GSPN performance model; and (iii) a way of combining the results from these two models. Models and performability results for an iPSC/2 hypercube system under the workload of concurrent matrix multiplication algorithm are presented."
LaMP-6	[label] title abstract	"Get paid real $$$$$ to drive your own car!" "You are receiving this exclusive promotion because you agreed to receive special offers from an emailYOUlike marketing partner. If you have received this email in error or would like to no longer receive these special offers, please follow the instructions at the end of the message. This message is brought to you by emailYOUlike. To find out more about emailYOUlike, visit http:\/\/www.emailyoulike.com\/ or write us at 212 Technology Dr., Suite P, Irvine, CA 92602. If you would prefer not to receive future marketing messages from us, click here or visit http://www.emailyoulike.com/remove.asp, enter your email address, and click on the unsubscribe button. Only unsubscribe requests submitted to this page can be fulfilled. emailYOUlike cannot fulfill unsubscribe requests submitted elsewhere or to the email boxes of individuals."
LaMP-7	tweet	"Atleast Now, All fake political drama by parties (to get votes) in TN in the name of suffering Eelam Tamils will end, Thats it, Game over HT Channel News: 25000 SL Tamils lie injured in NFZ w/o medical care and food. Thousands died in the final assault by SL army"

lowing analyses, we will use "difficulties" defined here loosely in multiple guidelines in the context of better choices.

3.1 Optimal Model and Customization for LLMs on the Edge

Study target. As illustrated in Figure 1, two primary questions need to be addressed within the constraints of edge device resources: (1) Which LLM is best suited for the edge device? and (2) What appropriate customization method should be applied to the chosen LLM? The performance of LLMs of various sizes, combined with different customization methods, can vary significantly depending on user tasks. These questions are particularly crucial for edge LLMs, as resource constraints necessitate inevitable tradeoffs between model size and customization methods.

Table 5. One Sample of Query Data in Each Dataset

Dataset	Prompt
LaMP-1	"For an author who has written the paper with the title "An application-specific protocol architecture for wireless microsensor networks", which reference is related? Just answer with [1] or [2] without explanation. [1]: "End-to-end Internet packet dynamics" [2]: "Energy-Neutral Source-Channel Coding with Battery and Memory Size Constraints"
LaMP-2	"Which tag does this movie relate to among the following tags? Just answer with the tag name without further explanation. tags: [sci-fi, based on a book, comedy, action, twist ending, dystopia, dark comedy, classic, psychology, fantasy, romance, thought-provoking, social commentary, violence, true story] description: A snobbish phonetics professor agrees to a wager that he can take a flower girl and make her presentable in high society."
LaMP-3	"What is the score of the following review on a scale of 1 to 5? just answer with 1, 2, 3, 4, or 5 without further explanation. review: If You Were Me and Lived In...Germany: A Child's Introduction to Culture Around the World (If You Were Me and Lived) by Carole P. Roman, Kelsea Wienrenga (Illustrations) is a wonderful addition to the series. It's a delight to read and look at the illustrations! Plus this book provides so many facts about the culture and customs in Germany but not in a dry, boring way. This series is a terrific way to spark interest in the world for your child and maybe for you! With thanks to the author for my copy."
LaMP-4	"Generate a headline for the following article: Being an ex wife was very unexpected. I went into it kicking and screaming. And drunk texting. Oh-and a little bit of stalking. To those going through it now I can tell you, you will survive."
LaMP-5	"Generate a title for the following abstract of a paper: Web caching is the process in which web objects are temporarily stored to reduce bandwidth consumption, server load and latency. Web prefetching is the process of fetching web objects from the server before they are actually requested by the client. Integration of caching and prefetching can be very beneficial as the two techniques can support each other. By implementing this integrated scheme in a client-side proxy, the perceived latency can be reduced for not one but many users. In this paper, we propose a new integrated caching and prefetching policy called the WCP-CMA which makes use of a profit-driven caching policy that takes into account the periodicity and cyclic behaviour of the web access sequences for deriving prefetching rules. Our experimental results have shown a 10%-15% increase in the hit ratios of the cached objects and 5%-10% decrease in delay compared to the existing scheme."
LaMP-6	"Generate a subject for the following email: You are receiving this exclusive promotion because you agreed to receive special offers from an emailYOUlike marketing partner. If you have received this email in error or would like to no longer receive these special offers, please follow the instructions at the end of the message. This message is brought to you by emailYOUlike. To find out more about emailYOUlike, visit http:\/\/www.emailyoulike.com\/ or write us at 212 Technology Dr., Suite P, Irvine, CA 92618. If you would prefer not to receive future marketing messages from us, click here or visit http://www.emailyoulike.com/remove.asp, enter your email address, and click on the unsubscribe button. Only unsubscribe requests submitted to this page can be fulfilled. emailYOUlike cannot fulfill unsubscribe requests submitted elsewhere or to the email boxes of individuals."
LaMP-7	"Paraphrase the following tweet without any explanation before or after it: I concur with @peyarili that there is animosity, and I believe that the Indian media is exacerbating the situation for the students. It seems as though the foolish media desires conflict with Australia."

Experiments and guidelines. To investigate these questions, we first compiled a collection of over 30 LLMs with diverse architectures and sizes, as described in Section 2.2. We then selected five widely used customization methods: LoRA [29], Prompt Tuning [40], Prefix Tuning [43], IA3 [49], and RAG [41]. We evaluated these LLMs on datasets designed for personalization tasks. It's worth noting that LoRA, Prompt Tuning, Prefix Tuning, and IA3 are categorized as PEFT methods[17].

Table 6 presents results for 10 representative LLMs across the five customization methods on four tasks of increasing difficulty. The featured models include Pythia(Py)-2.8B, OPT-

Table 6. Performance Comparisons between Parameter Learning and RAG, Across Ten Relatively Big Models on Four Datasets of Increasing Difficulty

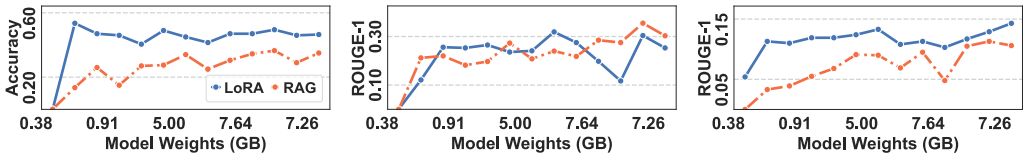
	Models	Py-2.8b	OPT-2.7b	Ll2-3b	Sta-3b	Ge-2b	Phi-2	Mis-7b-G	OpCt-3.5-G	Ge-7b-G	S-Ll-2.7b-P
LaMP-2	PfixT	0.223	0.105	0.035	0.145	0.130	0.122	0.205	0.145	0.185	0.033
	PmptT	0.055	0.025	0.045	0.087	0.070	0.204	0.085	0.215	0.170	0.050
	IA3	0.055	0.080	0.085	0.090	0.108	0.115	0.189	0.105	0.155	0.055
	LoRA	**0.475**	**0.480**	**0.430**	**0.480**	**0.430**	**0.450**	**0.485**	**0.460**	**0.420**	**0.455**
	RAG	0.320	0.110	0.295	0.245	0.205	0.340	0.375	0.290	0.365	0.035
LaMP-3	PfixT	0.490	0.392	0.520	0.412	0.500	0.451	0.206	0.627	0.275	0.265
	PmptT	0.373	0.304	0.461	0.471	0.451	0.324	0.353	0.647	0.324	0.451
	IA3	0.480	0.314	0.451	0.343	0.539	0.567	0.425	0.605	0.314	0.382
	LoRA	**0.716**	0.627	**0.765**	**0.784**	**0.784**	**0.745**	**0.814**	0.647	**0.775**	**0.725**
	RAG	0.716	**0.696**	0.461	0.667	0.765	0.627	0.755	**0.814**	0.480	0.578
LaMP-6	PfixT	0.043	0.017	0.000	0.064	0.026	0.075	0.039	0.119	0.074	0.015
	PmptT	0.062	0.039	0.056	0.058	0.053	0.093	0.084	0.072	0.038	0.065
	IA3	0.063	0.049	0.070	0.084	0.057	0.081	0.102	0.098	0.079	0.046
	LoRA	**0.285**	**0.155**	**0.264**	**0.280**	0.295	**0.241**	0.296	0.304	0.198	**0.319**
	RAG	0.186	0.134	0.253	0.256	**0.299**	0.208	**0.327**	**0.355**	**0.223**	0.240
LaMP-7	PfixT	0.103	0.105	0.008	0.104	0.083	0.115	0.106	0.140	0.093	0.117
	PmptT	0.118	0.092	0.041	0.089	0.084	0.102	0.080	0.075	0.034	0.145
	IA3	0.118	0.140	0.011	0.099	0.103	0.135	0.117	0.103	0.081	0.137
	LoRA	0.154	**0.168**	**0.204**	0.108	0.201	**0.220**	**0.199**	**0.290**	0.030	0.124
	RAG	**0.170**	0.156	0.149	**0.166**	**0.270**	0.197	0.181	0.231	**0.126**	**0.188**

PfixT stands for prefix tuning, and PmptT stands for prompt tuning.

2.7B, Llama(Ll)-2-3B, StableLM(Sta)-3B, Gemma(Ge)-2B, Phi-2, Mistral(Mis)-7B quantized by GPTQ(G) [20], OpenChat(OpCt)-G, Gemma-7B-G, and Sheared(S)-Llama-2.7B pruned. Our experiments reveal that LoRA consistently outperforms the other three PEFT methods, although it occasionally underperforms compared with RAG. Consequently, we focus our subsequent investigations on LoRA and RAG. Additional experimental results can be found in Appendix A.1.1.

Figure 2 showcases 13 models of varying sizes, ranging from 160M to 13B parameters, and their performance on three tasks of increasing difficulty. The models, arranged on the x-axis based on their weight sizes, include Pythia-160m, OPT-125m, OPT-350m, Pythia-410m, Phi-1.5, Gemma-2b-GPTQ, Phi-2, Sheared-Llama-2.7b, Llama2-3b-GPTQ, Phi-3, Llama2-7b-GPTQ, OpenChat-3.5-GPTQ, and Llama-13b-GPTQ. In our analysis, we selected LaMP-2 as the easy classification task, LaMP-6 as the easy summarization task, and LaMP-4 as the hard summarization task. Through our extensive experiments, we conclude that the optimal choice of model and customization method depends on the complexity of the downstream task. We provide the general guidelines below:

— For edge LLM customizations across different tasks, LoRA and RAG should be primarily considered. Other PEFT methods, including Prefix Tuning, IA3, and Prompt Tuning, can be resource-inefficient and less effective compared with LoRA and RAG.
— For simple classification tasks, the optimal choice should be a small LLM with LoRA. However, when an LLM is excessively small, like Pythia-70m or Pythia-160m shown in Appendix A.1.1, the model may be incapable of handling even simple classification tasks. It is noteworthy to avoid selecting such overly small models.
— As task difficulty increases, such as with complex classification tasks and simple summarization tasks, the choice should gradually shift to RAG with the strongest model. Here, the



(a) Easy classification task (b) Easy summarization task (c) Hard summarization task

Fig. 2. Performance comparisons on multiple models on tasks with different difficulties. The model size in the figures increases from left to right. The easy classification task refers to LaMP-2, the easy summarization task refers to LaMP-6, and the hard summarization task refers to LaMP-4.

strongest models are (quantized) LLMs that excel at general benchmarks and fit within the RAM constraint.

— As the difficulty of downstream tasks further increases to challenging summarization tasks, RAG will not be sufficient, and LoRA with strong and quantized models will stand out as the optimal choice.

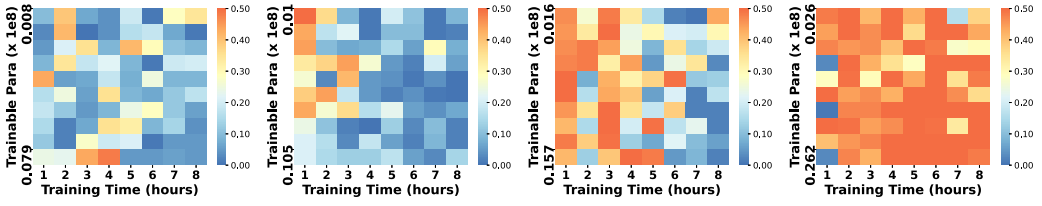
— When the difficulty of a task cannot be readily assessed, the Phi family combined with LoRA provides a safe option, offering decent performance across various scenarios.

Remark 1: Why does RAG work best with moderately complex tasks? We hypothesize that this behavior stems from RAG's reliance on the base LLM's semantic understanding, which limits the potential for performance improvement. Given the emergent abilities of LLMs [81], where certain skills (particularly those required for solving more complex problems) only appear in larger models, it is expected that smaller LLMs perform well on simpler tasks but struggle with more complex ones. Furthermore, LLMs suited for edge deployment tend to be smaller in size, meaning that even 7B-parameter models may lack the capacity to address highly complex tasks effectively using RAG. In these cases, fine-tuning is more advantageous, as it teaches the models the semantic structure of the expected answers, making parameter learning more effective than RAG for such difficult tasks.

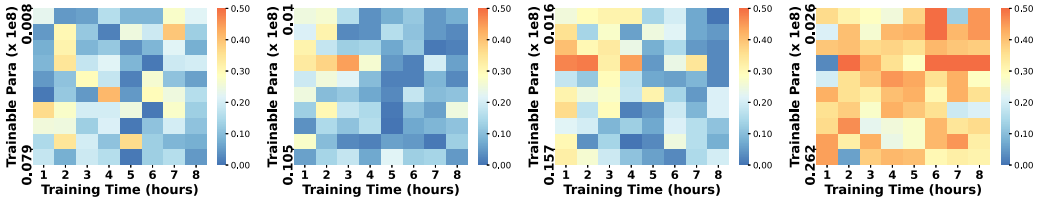
3.2 Choice of LoRA Strategies

Study target. In Section 3.1, we identified the generally superior performance of LoRA and RAG, and determined the situations where LoRA is more suitable than RAG. This section delves deeper into LoRA settings, aiming to establish efficient strategies for optimizing LoRA configurations across various edge LLM use cases. This investigation is particularly crucial for edge LLMs, given the limited resources available for conducting multiple experiments to determine optimal settings. In this context, we address three key questions. **First**, is it necessary to experiment with different LoRA settings to enhance model performance across various models and datasets? **Second**, is there an optimal range or a universal LoRA setting that is effective for all edge LLM applications? Answering these two questions can significantly benefit future edge LLM research and deployment by conserving time and resources in performance optimization. **Finally**, we aim at understanding the factors that cause LoRA to behave differently in edge LLMs compared with LLMs hosted on cloud servers.

Experiments and guidelines. Among the hyper-parameters of LoRA, *rank* and *alpha* are most closely related to the number of trainable parameters, which directly impact resource usage. Moreover, these two hyper-parameters significantly influence edge LLM customization [91]. While inappropriate combinations of *rank* and *alpha* can degrade edge LLM performance, identifying the optimal combination can be challenging. Different edge LLMs, operating under varying resource constraints, may require different optimal values for *rank* and *alpha*. Therefore, we investigate

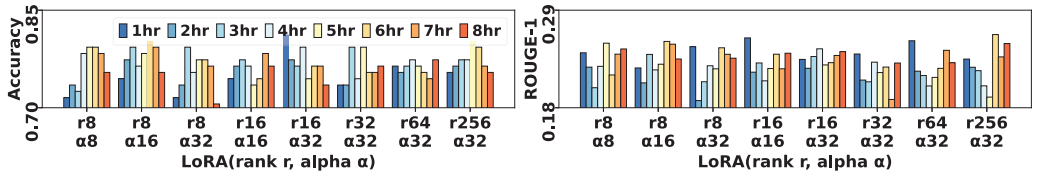


(a) Py-410m (α=32) (b) Py-1b (α=32) (c) Py-1.4b (α=32) (d) Py-2.8b (α=32)



(e) Py-410m (α=r) (f) Py-1b (α=r) (g) Py-1.4b (α=r) (h) Py-2.8b (α=r)

Fig. 3. For (a)-(h): performance comparisons for Pythia(Py) models on LaMP-1. From (a) to (d), we set *alpha* (α) to 32. From (e) to (h), we set α = *rank*(r). More are in Appendix A.1.2 and Appendix A.1.3.



(a) Sta-2-1.6b on LaMP-3 (b) Sta-3b on LaMP-5

Fig. 4. Performance comparisons for StableLM(Sta) models on LaMP-3 and LaMP-5 over eight combinations of alpha and rank. More results are in Appendix A.1.4.

the impact of LoRA settings from two perspectives: the number of trainable parameters and the specific combinations of *rank* and *alpha*.

In Figure 3, we examine the performance of different-sized Pythia models given the value of rank equal 8, 16, 32, 40, 48, 56, 64, 72, and 80, under the training time from one to eight hours. Additionally in Figure 4, we examine the performance of StableLM-2-1.6b and the larger StableLM-3b on eight commonly used *rank* and *alpha* combinations, under the training time 1 to 8 hours. Our findings are as follows:

— Within the limited edge device resource, through experiments over various combinations of *alpha* and *rank*, varying the value of *alpha* benefits less than fixing the value of *alpha*.
— Increasing the training time might either have negligible LLM performance improvement or even lower the performance (A more detailed analysis on the training time will be provided in Section 3.3). This situation remains consistent even when the value of *rank* is increased.
— As shown in Figures 3 and 4 and its supplemental results in Appendix A.1.2, Appendix A.1.3, and Appendix A.1.4, even in larger models like StableLM-3b and Pythia-2.8b deployed on edge devices with 10G or 16G RAM, increasing *rank* or *alpha* does not necessarily improve the model performance. Setting *alpha* and *rank* to (16, 16) or (16, 32) can work in most cases.

Remark 2: Why is fixing alpha necessary but not rank for LoRA fine-tuning? We hypothesize that fixing α serves as an implicit mechanism to mitigate overfitting. As increasing r enhances the model's adaptation to the training dataset, fixing the impact of LoRA on the original model with a constant $\frac{a}{r}$ could cause the model to focus excessively on the specifics of the training data, potentially leading to overfitting. However, we acknowledge that these findings may seem counterintuitive, and further research is needed to explore this topic, particularly in the context of resource-constrained fine-tuning.

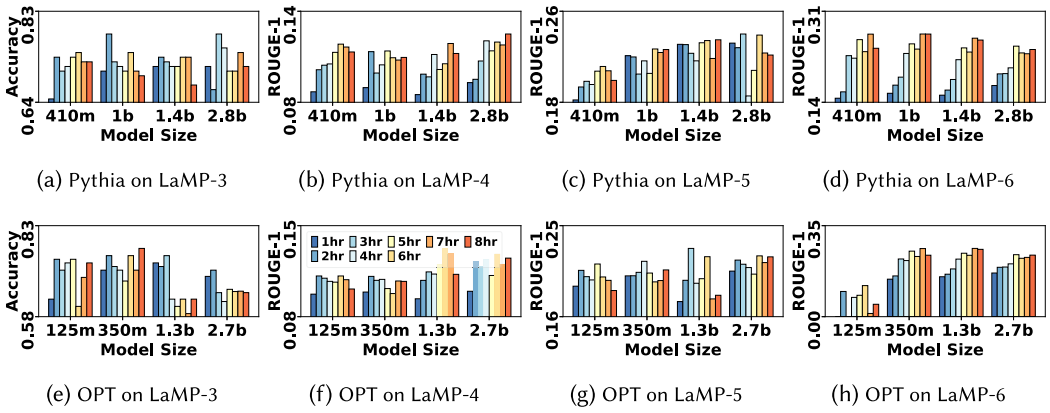
3.3 Trend of LoRA Training Time for Edge LLMs

Study target. In Section 3.2, we identified appropriate LoRA settings for various user cases. Building upon this, it is crucial to determine whether extending training time under each setting yields additional benefits. For instance, if we can conclusively establish that longer training periods offer negligible performance improvements, we should consistently keep edge LLM training brief for resource efficiency—a critical consideration for edge devices. Notably, longer training times allow for the incorporation of more user history data in edge LLM training. Intuitively, due to their limited pre-trained knowledge stemming from smaller pre-trained weights, certain edge LLMs may exhibit learning behaviors from user history data that differ from those of larger edge LLMs with more extensive pre-trained knowledge. Considering these two aspects, studying the LoRA performance trend over extended training periods is of paramount importance, allowing us to explore the edge LLM's learning boundary.

Experiments and guidelines. To conduct our investigations, we first quantify training time and correlate it with the amount of training data processed. For each model, we measure the number of data samples it can process within one hour. Detailed information can be found in Appendix 2.2. Longer training times allow for more data samples to be processed by the edge LLM. After establishing this quantitative relationship between training time and data volume, we proceed to select edge LLMs for various edge devices. To minimize potential differences arising from varied model architectures, we choose four Pythia models and four OPT models of different sizes, deploying them on edge devices with RAM requirements ranging from 4G to 16G. Detailed specifications are provided in Appendix 2.1. For LoRA settings, we adhere to the guidelines derived from Section 3.2, setting the *rank* to 8 and *alpha* to 32. We vary the training time from 1 to 8 hours, corresponding to typical idle periods for edge devices such as smartphones when users are asleep and the device is charging. The experimental results are presented in Figures 5 and 6. Our findings are as follows:

- More training data does not necessarily mean better performance. Under most tasks, training with 3–4 hours is usually enough for customizing the LLMs toward downstream tasks on the A14 Bionic chip, and the time could be adjusted accordingly based on the edge device.
- While fine-tuning with LoRA is useful for almost all cases, increasing the training time only shows consistent improvement over time on the LaMP-2 dataset, the easiest task out of all seven, as shown in Figures 20 and 21.

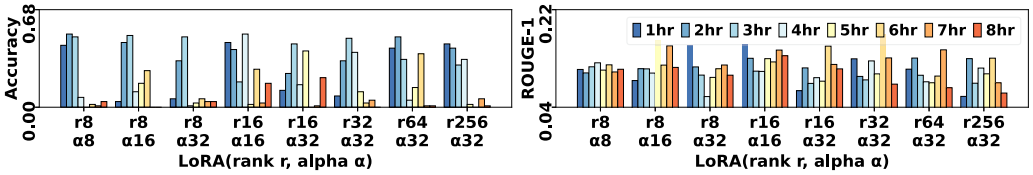
Remark 3: Why does test accuracy not always improve after training for some time? We hypothesize that this behavior may result from two factors. First, since the evaluated tasks are based on personalized datasets, the training data might lack diversity, increasing the risk of overfitting. In such cases, the LLM may learn patterns tied to specific tokens rather than capturing the desired features of the broader target population. Second, due to resource constraints on edge devices, we simulate practical scenarios by limiting the training data size. Recent work [11, 44, 56] discusses the phenomenon of the "pre-scale law", which suggests that loss during LLM fine-tuning



(a) Pythia on LaMP-3 (b) Pythia on LaMP-4 (c) Pythia on LaMP-5 (d) Pythia on LaMP-6

(e) OPT on LaMP-3 (f) OPT on LaMP-4 (g) OPT on LaMP-5 (h) OPT on LaMP-6

Fig. 5. Performance comparisons on multiple sized Pythia and OPT models on different amounts of training data. More performance comparisons can be found in Appendix A.1.5.



(a) Sta-2-1.6b on LaMP-1 (b) Sta-3b on LaMP-7

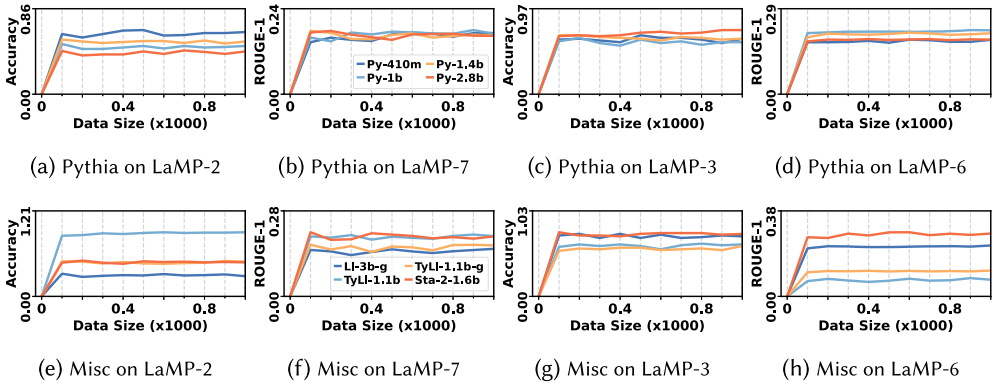
Fig. 6. Performance comparisons for StableLM(Sta) models on LaMP-1 and LaMP-7 over eight combinations of alpha and rank. More results are in Appendix A.1.4.

does not decrease linearly. Early in fine-tuning, test loss may not reduce quickly. Our observations in the experiments may align with the "pre-power law" stage, or even an earlier stage, where the LLM is learning an initial adaptation to the downstream task. In the first stage, the model improves as it learns the task representation. However, it then enters the pre-power-law stage, where performance oscillates. Finally, the model would reach the power-law stage, which requires at least 10K–100K samples, as noted by Lin et al. [44], a scale that is nearly impossible to achieve under the constraints of edge devices.

3.4 Impact of User History Data Volume on RAG Performance

Study target. Unlike LoRA, RAG involves simpler hyper-parameters, primarily related to MIPS, its core functionality [93]. While LoRA's challenge lies in parameter determination for edge LLMs, RAG's difficulty stems from increasing stored user history data and search latency [68]. Cloud-hosted LLMs benefit from this data growth due to their strong reasoning capabilities. However, edge LLMs, with significantly smaller pre-trained knowledge bases and lower reasoning capabilities [45], may not consistently gain the same advantages. This study aims to investigate whether expanding stored user history data consistently enhances RAG performance for edge LLMs or if there's a threshold beyond which data volume no longer improves performance, thus elucidating the relationship between data volume and RAG performance in edge LLMs.

Experiments and guidelines. We select 100 users and each user has up to 1,000 samples of user history data. For the history data, we randomly maintain 0% to 100% of them, where 0% corresponds to the case where no RAG is used. RAG takes an LLM as its content generator. We examine four



(a) Pythia on LaMP-2 (b) Pythia on LaMP-7 (c) Pythia on LaMP-3 (d) Pythia on LaMP-6

(e) Misc on LaMP-2 (f) Misc on LaMP-7 (g) Misc on LaMP-3 (h) Misc on LaMP-6

Fig. 7. RAG performance comparison on four Pythia(Py) models and four miscellaneous(Misc) models including Llama(Ll)-3b-GPTQ(G), StableLM(Sta)-2-1.6B, TinyLlama(TyLl)-1.1B-G, and TyLl-1.1B across different sizes of user history data (**Data Size**). More can be found in Figure 23 and Figure 23 in Appendix A.2.1.

Pythia models with different sizes and four miscellaneous models including Llama-3b-v2-GPTQ, TinyLlama-1.1b, TinyLlama-1.1b-GPTQ, and StableLM-2-1.6b. We calculate the performance improvement brought by RAG using different amounts of user history data.

As shown in Figure 7, we observe that increasing the amount of user history data to 1,000 samples does not significantly enhance RAG performance compared with using only 100 samples. The RAG performance based on eight models is quite consistent across all different sizes of user history data. Noted, in Figure 7(c), increasing user history data can even lower the RAG performance based on Pythia-1b and Pythia-1.4b. The performance of RAG can be more related to the internal reasoning capabilities of the LLM rather than the amount of user history data.

Remark 4: Why does using less history data have marginal effects on RAG performance? These results are not surprising, particularly when considering that personalized data may not be as diverse as general domain data. Additionally, connecting back to Section 3.1, we hypothesize that RAG elevates the LLM close to a stage just before the "pre-power law" by providing grounded context on the target's format and basic information. Such a hypothesis might be the reason why RAG only performs better than LoRA on tasks with moderate difficulty. For simple tasks, LoRA might be able to reach a stage beyond "pre-power law" (which is the upper bound of RAG). For complex tasks, RAG on edge LLMs cannot understand anything, whereas LoRA at least can learn some semantic structure, which helps the ROGUE metric.

3.5 Comparison of Model Compression Techniques on Edge LLMs

Study target. Other than customization methods, the model itself is the other key component in our empirical study, shown in the right dashed rectangle in Figure 1. When deploying an LLM onto edge devices, it normally can be expected that the model needs to be compressed to fit the devices with various resource constraints. Hence, the critical question is: what is the best way to compress it? As such large-scale LLMs have demonstrated promising reasoning abilities when they are deployed on cloud servers, their compressed versions are also expected to demonstrate similar reasoning abilities when they are deployed on edge devices.

Notably, one of the compression techniques is quantization, which has several implementations, including SmoothQuant [83], AWQ [45], and GPTQ [20]. However, only GPTQ provides robust support for training quantized models. In other words, when a model is quantized using GPTQ, it can still be further trained. In contrast, models quantized with SmoothQuant or AWQ—primarily designed to accelerate LLM inference—cannot be trained further unless they are fine-tuned on the

original unquantized model and then re-quantized, a process that may be impractical in resource-constrained environments. Consequently, as we train models using other compression techniques to explore LLM customization, we have selected GPTQ as our quantization implementation.

Experiments and guidelines. Among various model compression techniques, there are three main approaches including quantization, pruning, and knowledge distillation. While each of them may have the potential to conduct a decent compressed LLM performance, it remains unknown which one of the methods can be more promising and appropriate under the constraints of edge devices. We take the representative quantization method *GPTQ*, pruning method *structure pruning*, and knowledge distillation model *Phi* to make investigations. Our findings through the investigations are as follows (full experiments in Appendix B.1 to B.4):

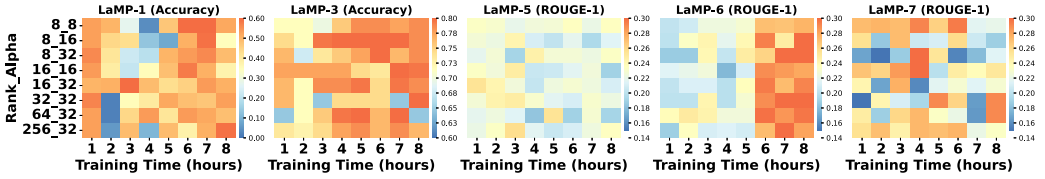
- Using distilled models such as Phi is a safe option when the type and difficulty of the downstream task cannot be determined. Phi-1.5 shows robustness towards both classification and summarization tasks, where it generally has a "brighter" performance heatmap shown in Figure 8, whereas Phi-3 excels in classification tasks.
- Different compression techniques are good at different types of tasks. Summarization tasks require larger (and better) LLMs to achieve emergent ability, thus GPTQ models such as OpenChat-3.5-GPTQ work better than the other two compression techniques.
- Shearing is a very promising technique that preserves better performance, but they are less efficient at saving RAM compared with quantization. Thus, shearing is generally not preferred for edge LLMs where RAM capacity is a critical constraint.

Remark 5: What are the foundations behind each compression technique? From the results, we observe that model distillation is the most stable method for fitting a large LLM on edge devices, although it rarely (if ever) stands out as the best option. In contrast, GPTQ is less stable but often achieves the best performance when combined with certain LoRA parameters and training data. Shearing and pruning, however, perform poorly in comparison to these two methods. It should be clear why sheared and pruned models are not ideal for edge devices. Pruning removes specific weights from the model, but since the models must be fine-tuned for downstream tasks, the missing weights are likely to be reintroduced during fine-tuning. As a result, pruned models offer no advantages in RAM usage and do not retain the semantic understanding abilities of larger models. Additionally, we hypothesize that the quality and memorization of pre-training datasets influence the fine-tuning of these models. It is well known that LLMs memorize parts of their training datasets [7, 90], but two uncertainties remain. First, we do not know the quality of the memorized data (i.e., do they memorize only high-quality/significant data, or do they memorize anything?). Second, we do not know **where** they memorize the data. It is possible that models trained on low-quantity pre-training data exhibit unstable behavior in downstream tasks, but further investigation is needed to confirm this.

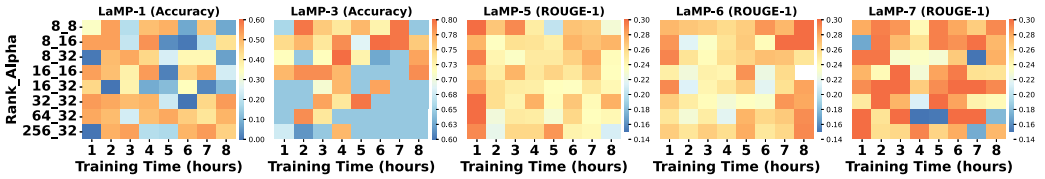
3.6 Comparison between Compressed and Uncompressed Edge LLMs

Study target. After we compare the three compression techniques for their adaption on edge devices, we further need to investigate the impact of such compression: Consider an edge device with 16G RAM which is sufficient to host some uncompressed large-scale LLMs. While the compressed models can learn and infer faster than their larger original versions, will such benefits compromise the model's performance? Other than model performance, learning efficiency under compressed and uncompressed models is also important to consider.

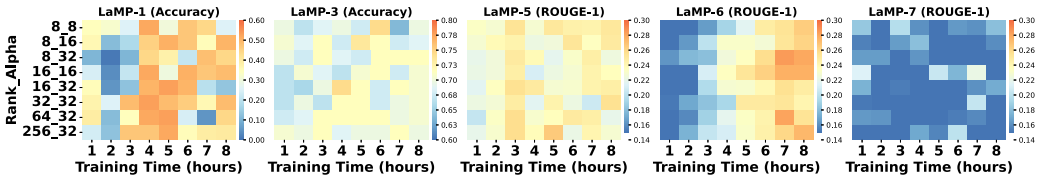
Experiments and guidelines: Based on the experiments and their settings in Section 3.5, we concentrate on two groups of comparisons: between quantized and un-quantized models, and between pruned and un-pruned models. We select Gemma, Llama, and StableLM models. The results are shown in Figure 9. Through our investigations, our findings are as follows:



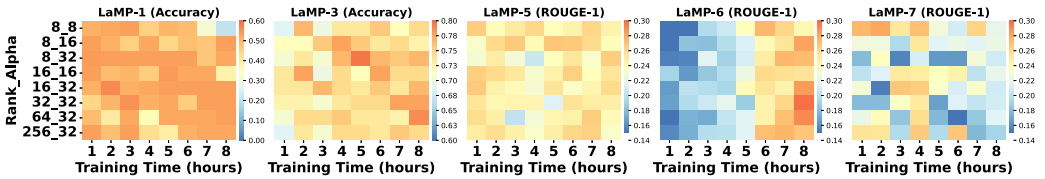
(a) Llama-2-3b-GPTQ



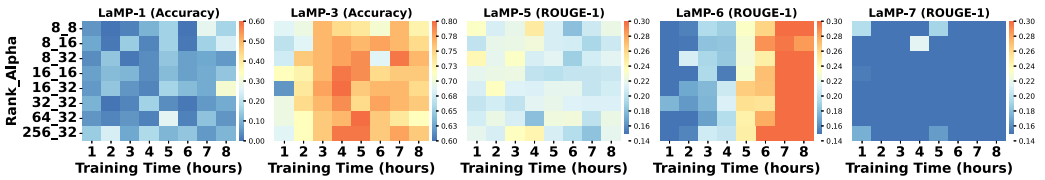
(b) OpenChat-3.5-GPTQ



(c) Phi-1.5



(d) Phi-2



(e) Sheared-Llama-2.7B-Pruned

Fig. 8. Performance comparisons of quantization, knowledge distillation, and pruning models over eight commonly used LoRA (*rank_alpha*) settings. More results in Figure 28–Figure 41 can be found in Appendix B.4.

— While it is true that quantization will lead to some accuracy drops in most cases [90], it is noteworthy that with limited user history data for fine-tuning, the quantized model might perform better compared with unquantized counterparts on more challenging tasks.
— Different from quantization which can benefit edge LLMs, pruning as shown from Figure 9(j) to Figure 9(o) indicates a clear performance drop in most LoRA settings and various datasets. Hence, pruning in edge LLM fine-tuning is not recommended.

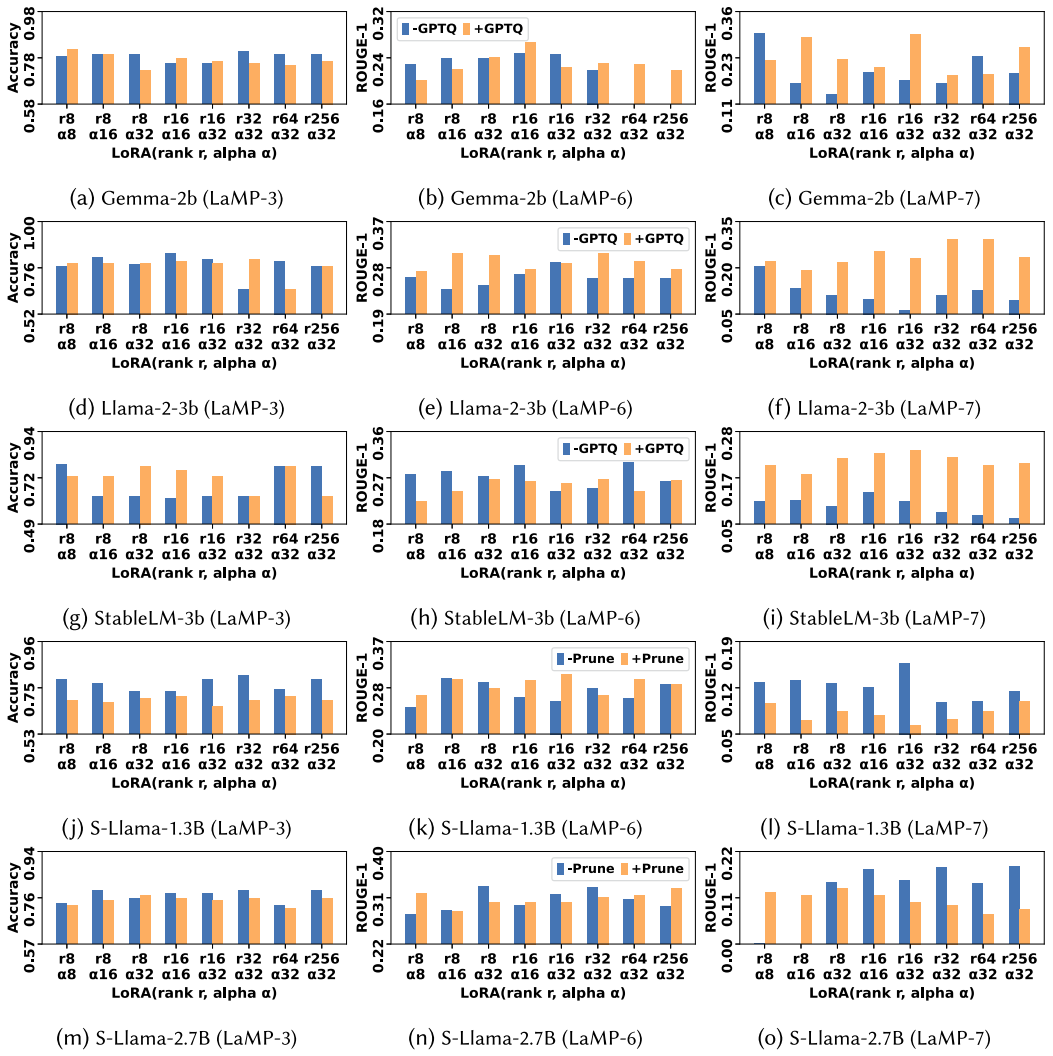


Fig. 9. Performance comparisons of four quantized models, two pruned models with their corresponding original models on eight commonly used LoRA settings. More experiments are presented in Appendix B.5.

— A larger model is not always better. Llama3-8B is not performing well on many of the benchmarks, maybe because it is too big and not able to adapt to downstream tasks with limited user history data. Especially during classification tasks, we see that 2B to 3B models (quantized) are good enough. On the other hand, referring to Table 8, bigger models work better with summarization tasks that require more semantic understanding of the context.

Remark 6: Why is a compressed model sometimes better? We hypothesize that such a phenomenon shares the same intuition with the Section 3.5. An LLM memorizes its pre-training data, and they are in the model's weights in a numerical form that we cannot interpret. Fine-tuning toward the downstream task is essentially the process of letting the LLM forget some of the pre-trained data and memorize data from the target domain. Thus, quantization removes a lot of the

Table 7. A Glance at Empirical Guidelines from Multiple Perspectives

	Model Size	Training Time	Data Volume	RAM Size
Overview	At least 1 billion parameters are needed to ensure the effectiveness of LLM	Longer training time benefits less than larger trainable parameters	Maintain a compact but high-quality data volume	Within certain RAM, large model benefits more than large data volume
Custmization	Parameter tuning has larger potential than information retrieval to improve the LLM performance	When training time is highly constrained, information retrieval can still provide decent improvements	Maintain such a good data volume can consume resource, but still be valuable to benefit both customization methods	For parameter tuning (LoRA), a large model with fixed settings (consuming small trainable parameters) like Rank 16 and Alpha 32
Compression	Quantization can achieve the best performance but with the risk of training failure. Distillation has stable and decent performance in all cases.	Training quantized model is better than training pruning or uncompressed models	Maintain such a good data volume can consume resource, but still be valuable to benefit both customization methods	Choose the largest possible quantized model with setting (Rank 16 and Alpha 32) can be an optimal choice for on-device learning
Difficulty	Easy Task: small LLM with parameter tuning. Medium Task: large LLM with information retrieval. Difficult Task: large LLM with parameter tuning.	Longer training time benefits more on the difficult tasks, while the easy task requires very few training	Maintain such a good data volume can consume resource, but still be valuable to benefit both customization methods	Choose the largest possible quantized model with setting (Rank 16 and Alpha 32) can be an optimal choice for on-device learning

pre-training information that is stored in the less significant bits of the LLM and makes the fine-tuning faster.

4 Conclusion and Future Directions

In conclusion, we present an empirical study for deploying LLMs onto edge devices with multiple resource constraints, and we provide guidelines for choosing the optimal strategy for the deployment. Through experiments, we show that the optimal choice of LLM and customization depends on the difficulty of the downstream task; during PEFT fine-tuning, the largest possible parameters and time are not the optimal settings; and compressed models are sometimes better than the original models on the edge due to their faster adaptation speed. While we provide detailed guidelines in the previous section, we have also summarized them into Table 7 to offer a general overview for easier reference.

The insights provided by this study not only offer empirical guidelines for future deployment efforts but also highlight key avenues for further research into adapting LLMs in resource-constrained environments. In addition to these findings, we put forth several recommendations for the LLM community, drawn from our experimental results:

— Small LLMs could sometimes solve domain-specific problems better than larger models. The community, especially the industry, should consider private small LLMs as alternatives to larger models on many edge services.
— When fine-tuning small LLMs on the edge, the tradeoff between time and performance shall be carefully considered. Larger trainable parameters and more training time are not always the best. Different from the cloud LLMs where there are often no validation datasets, edge LLMs should employ a validation set to ensure the best fine-tuning performance.
— The RAG ability of edge LLMs is largely unexplored. Existing RAG frameworks heavily rely on the semantic understanding ability of LLMs, which might not exist on the edge.

— Model distillation is shown to be an effective solution that migrates the larger model's semantic understanding ability to edge devices, but more research shall be directed into this area.

References

[1] OpenAI (2023). 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Sara Babakniya, Ahmed Roushdy Elkordy, Yahya H. Ezzeldin, Qingfeng Liu, Kee-Bong Song, Mostafa El-Khamy, and Salman Avestimehr. 2023. SLoRA: Federated Parameter Efficient Fine-Tuning of Language Models. arXiv:2308.0652. Retrieved from https://arxiv.org/abs/2308.0652

[3] Renu Balyan, Kathryn S McCarthy, and Danielle S McNamara. 2020. Applying natural language processing and hierarchical machine learning approaches to text difficulty classification. *International Journal of Artificial Intelligence in Education* 30, 3 (2020), 337–370.

[4] Marco Bellagente, Jonathan Tow, Dakota Mahan, Duy Phung, Maksym Zhuravinskyi, Reshinth Adithyan, James Baicoianu, Ben Brooks, Nathan Cooper, and Ashish Datta. 2024. Stable LM 2 1.6 B technical report. arXiv:2402.17834. Retrieved from https://arxiv.org/abs/2402.17834

[5] Marialena Bevilacqua, Kezia Oketch, Ruiyang Qin, Will Stamey, Xinyuan Zhang, Yi Gan, Kai Yang, and Ahmed Abbasi. 2025. When automated assessment meets automated content generation: Examining text quality in the era of GPTs. *ACM Transactions on Information Systems* 43, 2 (2025), 1–36.

[6] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*. PMLR, 2397–2430.

[7] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=TatRHT_1cK

[8] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2024. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.* 15, 3, Article 39 (mar 2024), 45 pages. DOI: https://doi.org/10.1145/3641289

[9] Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 17754–17762.

[10] Rung-Ching Chen, Christine Dewi, Su-Wen Huang, and Rezzy Eko Caraka. 2020. Selecting critical features for data classification based on machine learning methods. *Journal of Big Data* 7, 1 (2020), 52.

[11] Aidan Clark, Diego De Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, George Bm Van Den Driessche, Eliza Rutherford, Tom Hennigan, Matthew J Johnson, Albin Cassirer, Chris Jones, Elena Buchatskaya, David Budden, Laurent Sifre, Simon Osindero, Oriol Vinyals, Marc.Aurelio Ranzato, Jack Rae, Erich Elsen, Koray Kavukcuoglu, and Karen Simonyan. 2022. Unified scaling laws for routed language models. In *International Conference on Machine Learning*. PMLR, 4057–4086.

[12] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168* (2021).

[13] Julian Coda-Forno, Marcel Binz, Zeynep Akata, Matthew Botvinick, Jane X Wang, and Eric Schulz. 2023. Meta-in-context learning in large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*. Retrieved from https://openreview.net/forum?id=sx0xpaO0za

[14] Isaac Martín De Diego, Ana R Redondo, Rubén R Fernández, Jorge Navarro, and Javier M Moguerza. 2022. General performance score for classification problems. *Applied Intelligence* 52, 10 (2022), 12049–12063.

[15] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*. Retrieved from https://openreview.net/forum?id=OUIFPHEgJU

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 4171–4186.

[17] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence* 5, 3 (2023), 220–235.

[18] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, AyzaanWahid, StefanWelker, PaulWohlhart, JialinWu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818* (2023).

[19] Bahar Irfan, Sanna Kuoppamäki, Aida Hosseini, and Gabriel Skantze. 2023. Between reality and delusion: Challenges of applying large language models to companion robots for open-domain dialogues with older adults. *Autonomous Robots* (2023).

[20] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323* (2022).

[21] Yingqiang Ge, Wenyue Hua, Kai Mei, Juntao Tan, Shuyuan Xu, Zelong Li, and Yongfeng Zhang. 2024. Openagi: When llm meets domain experts. *Advances in Neural Information Processing Systems* 36 (2024).

[22] Yeow Chong Goh, Xin Qing Cai, Walter Theseira, Giovanni Ko, and Khiam Aik Khor. 2020. Evaluating human versus machine learning performance in classifying research abstracts. *Scientometrics* 125, 2 (Jul 2020), 1197–1212. DOI : https://doi.org/10.1007/s11192-020-03614-2

[23] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644* (2023).

[24] Zishan Guo, Renren Jin, Chuang Liu, Yufei Huang, Dan Shi, Supryadi, Linhao Yu, Yan Liu, Jiaxuan Li, Bojian Xiong, and Deyi Xiong. 2023. Evaluating Large Language Models: A Comprehensive Survey. arXiv:2310.19736 [cs.CL].

[25] Perttu Hämäläinen, Mikke Tavast, and Anton Kunnari. 2023. Evaluating large language models in generating synthetic hci research data: A case study. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–19.

[26] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300* (2020).

[27] Elad Hoffer, Itay Hubara, and Daniel Soudry. 2017. Train longer, generalize better: Closing the generalization gap in large batch training of neural networks. *Advances in Neural Information Processing Systems* 30 (2017).

[28] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301* (2023).

[29] 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. *ICLR* 1, 2 (2022), 3.

[30] Jerry Yao-Chieh Hu, Pei-Hsuan Chang, Robin Luo, Hong-Yu Chen, Weijian Li, Wei-Po Wang, and Han Liu. 2024. Outlier-efficient hopfield layers for large transformer-based models. *arXiv preprint arXiv:2404.03828* (2024).

[31] Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933* (2023).

[32] Hugging Face. 2021. all-MiniLM-L6-v2: A Sentence Embedding Model. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[33] Nam Hyeon-Woo, Moon Ye-Bin, and Tae-Hyun Oh. 2021. Fedpara: Low-rank hadamard product for communication-efficient federated learning. *arXiv preprint arXiv:2108.06098* (2021).

[34] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825* (2023).

[35] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).

[36] Darioush Kevian, Usman Syed, Xingang Guo, Aaron Havens, Geir Dullerud, Peter Seiler, Lianhui Qin, and Bin Hu. 2024. Capabilities of large language models in control engineering: A benchmark study on GPT-4, claude 3 opus, and gemini 1.0 Ultra. *arXiv preprint arXiv:2404.03647* (2024).

[37] Achintya Kundu, Fabian Lim, Aaron Chew, Laura Wynter, Penny Chong, and Rhui Dih Lee. 2024. Efficiently distilling LLMs for edge applications. *arXiv preprint arXiv:2404.01353* (2024).

[38] Robert Lakatos, Peter Pollner, Andras Hajdu, and Tamas Joo. 2024. Investigating the performance of retrieval-augmented generation and fine-tuning for the development of AI-driven knowledge-based systems. *arXiv preprint arXiv:2403.09727* (2024).
[39] Ariel N Lee, Cole J Hunter, and Nataniel Ruiz. 2023. Platypus: Quick, cheap, and powerful refinement of llms. *arXiv preprint arXiv:2308.07317* (2023).
[40] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691* (2021).
[41] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, and Tim Rocktäschel. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.
[42] Shiyao Li, Xuefei Ning, Luning Wang, Tengxuan Liu, Xiangsheng Shi, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. 2024. Evaluating Quantized Large Language Models. arXiv:2402.18158 [cs.CL].
[43] Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190* (2021).
[44] Haowei Lin, Baizhou Huang, Haotian Ye, Qinyu Chen, Zihao Wang, Sujian Li, Jianzhu Ma, Xiaojun Wan, James Zou, and Yitao Liang. 2024. Selecting large language model to fine-tune via rectified scaling law. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*. Retrieved from https://openreview.net/forum?id=VfA3PAQJQO
[45] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. *Proceedings of Machine Learning and Systems* 6 (2024), 87–100.
[46] Ji Lin, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, and Song Han. 2022. On-device training under 256kb memory. *Advances in Neural Information Processing Systems* 35 (2022), 22941–22954.
[47] Zheng Lin, Guanqiao Qu, Qiyuan Chen, Xianhao Chen, Zhe Chen, and Kaibin Huang. 2024. Pushing Large Language Models to the 6G Edge: Vision, Challenges, and Opportunities. arXiv:2309.16739 [cs.LG].
[48] Dancheng Liu, Amir Nassereldine, Ziming Yang, Chenhui Xu, Yuting Hu, Jiajie Li, Utkarsh Kumar, Changjae Lee, Ruiyang Qin, and Yiyu Shi. 2024. Large language models have intrinsic self-correction ability. *arXiv preprint arXiv:2406.15673* (2024).
[49] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems* 35 (2022), 1950–1965.
[50] Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).
[51] Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888* (2023).
[52] Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. 2024. MobileLLM: Optimizing Sub-billion Parameter Language Models for On-Device Use Cases. arXiv:2402.14905 [cs.LG]. https://arxiv.org/abs/2402.14905
[53] Haozheng Luo, Ruiyang Qin, Chenwei Xu, Guo Ye, and Zening Luo. 2023. Open-ended multi-modal relational reasoning for video question answering. In *2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 363–369.
[54] Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Advances in Neural Information Processing Systems* 36 (2023), 21702–21720.
[55] Meta. 2023. Meta Llama 3. Retrieved from https://ai.meta.com/blog/meta-llama-3/. Accessed: 2024-05-28.
[56] Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A Raffel. 2024. Scaling data-constrained language models. *Advances in Neural Information Processing Systems* 36 (2024).
[57] Zhenyu Pan, Haozheng Luo, Manling Li, and Han Liu. 2024. Conv-CoA: Improving open-domain question answering in large language models via conversational chain-of-action. *arXiv preprint arXiv:2405.17822* (2024).
[58] Letian Peng, Zilong Wang, Feng Yao, Zihan Wang, and Jingbo Shang. 2024. MetaIE: Distilling a meta model from LLM for all kinds of information extraction tasks. *arXiv preprint arXiv:2404.00457* (2024).
[59] Qiwei Peng, Yekun Chai, and Xuhong Li. 2024. HumanEval-XL: A multilingual code generation benchmark for cross-lingual natural language generalization. *arXiv preprint arXiv:2402.16694* (2024).
[60] George Pu, Anirudh Jain, Jihan Yin, and Russell Kaplan. 2023. Empirical analysis of the strengths and weaknesses of peft techniques for LLMs. *arXiv preprint arXiv:2304.14999* (2023).

[61] Ruiyang Qin. 2025. *Hardware and Algorithm Co-Exploration for Efficient On-Device Personalization of Large Language Models*. Ph. D. Dissertation. University of Notre Dame.
[62] Ruiyang Qin, Ryan Cook, Kai Yang, Ahmed Abbasi, David Dobolyi, Salman Seyedi, Emily Griner, Hyeokhyen Kwon, Robert Cotes, Zifan Jiang, et al. 2024. Language models for online depression detection: A review and benchmark analysis on remote interviews. *ACM Transactions on Management Information Systems* (2024).
[63] Ruiyang Qin, Yuting Hu, Zheyu Yan, Jinjun Xiong, Ahmed Abbasi, and Yiyu Shi. 2024. FL-NAS: Towards fairness of NAS for resource constrained devices via large language models. *arXiv preprint arXiv:2402.06696* (2024).
[64] Ruiyang Qin, Dancheng Liu, Gelei Xu, Zheyu Yan, Chenhui Xu, Yuting Hu, X Sharon Hu, Jinjun Xiong, and Yiyu Shi. 2024. Tiny-align: Bridging automatic speech recognition and large language model on the edge. *arXiv preprint arXiv:2411.13766* (2024).
[65] Ruiyang Qin, Haozheng Luo, Zheheng Fan, and Ziang Ren. 2021. IBERT: Idiom cloze-style reading comprehension with attention. *arXiv preprint arXiv:2112.02994* (2021).
[66] Ruiyang Qin, Pengyu Ren, Zheyu Yan, Liu Liu, Dancheng Liu, Amir Nassereldine, Jinjun Xiong, Kai Ni, Sharon Hu, and Yiyu Shi. 2024. NVCiM-PT: An NVCiM-assisted prompt tuning framework for edge LLMs. *arXiv preprint arXiv:2411.08244* (2024).
[67] Ruiyang Qin, Jun Xia, Zhenge Jia, Meng Jiang, Ahmed Abbasi, Peipei Zhou, Jingtong Hu, and Yiyu Shi. 2023. Enabling on-device large language model personalization with self-supervised data selection and synthesis. *arXiv preprint arXiv:2311.12275* (2023).
[68] Ruiyang Qin, Zheyu Yan, Dewen Zeng, Zhenge Jia, Dancheng Liu, Jianbo Liu, Zhi Zheng, Ningyuan Cao, Kai Ni, and Jinjun Xiong. 2024. Robust implementation of retrieval-augmented generation on edge-based computing-in-memory architectures. *arXiv preprint arXiv:2405.04700* (2024).
[69] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv:1910.10683 [cs.LG].
[70] Vyas Raina, Adian Liusie, and Mark Gales. 2024. Is LLM-as-a-judge robust? Investigating universal adversarial attacks on zero-shot LLM assessment. *arXiv preprint* arXiv:2402.14016 (2024).
[71] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022* (2023).
[72] Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2023. LaMP: When large language models meet personalization. *arXiv preprint arXiv:2304.11406* (2023).
[73] V Sanh. 2019. DistilBERT, A distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).
[74] Xuan Shen, Peiyan Dong, Lei Lu, Zhenglun Kong, Zhengang Li, Ming Lin, Chao Wu, and Yanzhi Wang. 2024. Agile-quant: Activation-guided quantization for faster inference of LLMs on the edge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 18944–18951.
[75] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695* (2023).
[76] Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. 2024. Improving LoRA in privacy-preserving federated learning. In *The Twelfth International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=NLPzL6HWNl
[77] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, and Anja Hauth. 2023. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* (2023).
[78] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, and Juliette Love. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295* (2024).
[79] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, and Faisal Azhar. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).
[80] Fangyun Wei, Xi Chen, and Lin Luo. 2024. Rethinking Generative Large Language Model Evaluation for Semantic Comprehension. arXiv:2403.07872 [cs.CL].
[81] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682* (2022).
[82] Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694* (2023).

[83] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*. PMLR, 38087–38099.

[84] Chenhui Xu, Fuxun Yu, Zirui Xu, Chenchen Liu, Jinjun Xiong, and Xiang Chen. 2024. Quadranet: Improving high-order neural interaction efficiency with hardware-aware quadratic neural networks. In *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 19–25.

[85] Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, XIAOPENG ZHANG, and Qi Tian. 2024. QA-LoRA: Quantization-aware low-rank adaptation of large language models. In *The Twelfth International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=WvFoJccpo8

[86] Zheng Xu, Yanxiang Zhang, Galen Andrew, Christopher A Choquette-Choo, Peter Kairouz, H Brendan McMahan, Jesse Rosenstock, and Yuanbo Zhang. 2023. Federated learning of gboard language models with differential privacy. *arXiv preprint arXiv:2305.18465* (2023).

[87] Divakar Yadav, Jalpa Desai, and Arun Kumar Yadav. 2022. Automatic Text Summarization Methods: A Comprehensive Review. arXiv:2204.01849 [cs.CL].

[88] Zhewei Yao, Xiaoxia Wu, Cheng Li, Stephen Youn, and Yuxiong He. 2024. Exploring post-training quantization in llms from comprehensive study to low rank compensation. In *Proceedings of the AAAI Conference on Artificial Intelligence* 38 (2024), 19377–19385.

[89] Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. 2023. Edgemoe: Fast on-device inference of moe-based large language models. *arXiv preprint* arXiv:2308.14352 (2023).

[90] Lu Yin, Ajay Jaiswal, Shiwei Liu, Souvik Kundu, and Zhangyang Wang. 2024. Pruning Small Pre-Trained Weights Irreversibly and Monotonically Impairs "Difficult" Downstream Tasks in LLMs. arXiv:2310.02277 [cs.LG].

[91] Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. 2024. When scaling meets llm finetuning: The effect of data, model and finetuning method. *arXiv preprint* arXiv:2402.17193 (2024).

[92] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. 2024. Instruction Tuning for Large Language Models: A Survey. arXiv:2308.10792 [cs.CL].

[93] Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint* arXiv:2402.19473 (2024).

Appendix

A Results of Edge LLM Learning

A.1 Experimental Results for PEFT

A.1.1 Experiments: Examine RAG and PEFT. The experiments contain a wide range of edge LLMs and compare their learning performance by PEFT and RAG. These experiments can correspond to Section 3.1.

Table 8. Performance Comparisons of PEFT and RAG for Selected LLMs

LLM	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-6		LaMP-7	
	FT	RAG	FT	RAG	FT	RAG	FT	RAG	FT	RAG	FT	RAG	FT	RAG
Pythia-70m	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Pythia-160m	0.019	0.000	0.000	0.000	0.647	0.000	0.054	0.000	0.076	0.000	0.001	0.000	0.011	0.000
Pythia-410m	0.275	0.500	0.460	0.010	0.725	0.657	0.119	0.055	0.198	0.157	0.253	0.182	0.128	0.156
Pythia-1b	0.039	0.500	0.465	0.185	0.716	0.578	0.109	0.051	0.211	0.165	0.273	0.221	0.096	0.169
Pythia-1.4b	0.559	0.490	0.470	0.510	0.716	0.667	0.118	0.061	0.216	0.187	0.281	0.253	0.128	0.163
Pythia-2.8b	0.500	0.451	0.475	0.320	0.716	0.716	0.129	0.065	0.207	0.220	0.285	0.186	0.154	0.170
opt-125m	0.049	0.284	0.535	0.135	0.725	0.588	0.113	0.033	0.189	0.135	0.121	0.212	0.211	0.187
opt-350m	0.196	0.500	0.470	0.260	0.735	0.569	0.110	0.039	0.204	0.171	0.091	0.236	0.156	0.199
opt-1.3b	0.363	0.373	0.475	0.270	0.627	0.657	0.120	0.057	0.199	0.175	0.123	0.112	0.121	0.173
opt-2.7b	0.010	0.520	0.480	0.110	0.627	0.696	0.127	0.060	0.222	0.184	0.155	0.134	0.168	0.156
TinyLlama-1.1b	0.500	0.520	0.230	0.085	0.775	0.618	0.088	0.051	0.182	0.150	0.158	0.081	0.181	0.206
Llama-v2-3b	0.320	0.569	0.430	0.295	0.765	0.461	0.129	0.026	0.252	0.136	0.264	0.253	0.204	0.149
Llama-v3-8b	0.324	0.590	0.130	0.235	0.627	0.696	0.065	0.091	0.180	0.150	0.172	0.280	0.189	0.213
StableLM-2-1.6b	0.480	0.390	0.440	0.210	0.755	0.765	0.118	0.098	0.181	0.208	0.265	0.276	0.097	0.200
StableLM-3b	0.529	0.520	0.480	0.245	0.784	0.667	0.136	0.094	0.246	0.248	0.280	0.256	0.108	0.166
Gemma-2b	0.471	0.010	0.430	0.205	0.784	0.765	0.119	0.079	0.239	0.245	0.295	0.299	0.201	0.270
Phi-1.5	0.255	0.451	0.405	0.270	0.696	0.422	0.119	0.068	0.252	0.203	0.265	0.197	0.181	0.270
Phi-2	0.206	0.529	0.450	0.340	0.745	0.627	0.133	0.090	0.243	0.241	0.241	0.208	0.220	0.197
Phi-3	0.471	0.333	0.470	0.345	0.784	0.647	0.103	0.048	0.211	0.242	0.198	0.285	0.199	0.158
Llama-v2-3b-G[1]	0.520	0.559	0.470	0.305	0.784	0.706	0.113	0.095	0.234	0.215	0.275	0.218	0.220	0.151
StableLM-3b-G	0.490	0.559	0.440	0.390	0.725	0.598	0.118	0.082	0.234	0.251	0.226	0.258	0.200	0.144
TinyLlama-1.1B-G	0.265	0.422	0.410	0.120	0.765	0.588	0.115	0.057	0.225	0.182	0.173	0.112	0.197	0.171
Gemma-2b-G	0.167	0.461	0.490	0.275	0.814	0.716	0.124	0.091	0.240	0.247	0.201	0.273	0.227	0.169
Llama-v2-7b-G	0.451	0.520	0.495	0.365	0.755	0.657	0.129	0.105	0.250	0.278	0.172	0.275	0.249	0.157
Llama-v3-8b-G	0.539	0.520	0.140	0.365	0.461	0.631	0.073	0.115	0.229	0.245	0.174	0.280	0.277	0.156
Mistral-7b-G	0.529	0.529	0.485	0.375	0.814	0.755	0.128	0.097	0.255	0.274	0.296	0.327	0.199	0.181
OpenChat-3.5-G	0.539	0.520	0.460	0.290	0.647	0.814	0.129	0.113	0.227	0.279	0.304	0.355	0.290	0.231
Synthia-7b-G	0.451	0.529	0.480	0.365	0.725	0.549	0.123	0.088	0.244	0.302	0.301	0.338	0.263	0.213
Gemma-7b-G	0.529	0.539	0.420	0.365	0.775	0.480	0.105	0.048	0.213	0.184	0.198	0.223	0.030	0.126
Llama-13b-G	0.529	0.569	0.465	0.350	0.804	0.657	0.143	0.106	0.259	0.264	0.253	0.303	0.121	0.138
S^3-Llama-1.3b-P[2]	0.069	0.049	0.410	0.100	0.686	0.569	0.090	0.040	0.215	0.157	0.268	0.195	0.096	0.178
S-Llama-1.3B	0.461	0.471	0.510	0.255	0.784	0.559	0.113	0.064	0.205	0.197	0.247	0.234	0.126	0.183
S-Llama-2.7B-P	0.167	0.120	0.455	0.035	0.725	0.578	0.100	0.051	0.221	0.207	0.319	0.240	0.124	0.188
S-Llama-2.7B	0.294	0.529	0.415	0.250	0.775	0.627	0.108	0.069	0.216	0.242	0.218	0.260	0.234	0.181

For PEFT, we use the default experimental settings and let *rank* = 8, *alpha* =16. Additionally, we keep the training hours to 8 so the model can fully learn from the user history data. The PEFT performance compares with RAG. Their zero-shot learning performance can be found in Table 9.

Table 9. Performance Comparisons of Zero-Shot Learning for All Selected LLMs

LLM	LaMP-1	LaMP-2	LaMP-3	LaMP-4	LaMP-5	LaMP-6	LaMP-7
Pythia-70m	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Pythia-160m	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Pythia-410m	0.420	0.230	0.441	0.034	0.103	0.045	0.112
Pythia-1b	0.420	0.310	0.343	0.028	0.134	0.047	0.115
Pythia-1.4b	0.400	0.240	0.373	0.033	0.120	0.000	0.058
Pythia-2.8b	0.429	0.130	0.598	0.035	0.128	0.053	0.107
opt-125m	0.039	0.031	0.304	0.018	0.098	0.034	0.096
opt-350m	0.429	0.210	0.255	0.021	0.106	0.056	0.116
opt-1.3b	0.429	0.035	0.284	0.033	0.130	0.041	0.102
opt-2.7b	0.429	0.150	0.451	0.025	0.113	0.051	0.152
TinyLlama-1.1b	0.471	0.090	0.588	0.032	0.088	0.029	0.143
Llama-v2-3b	0.469	0.130	0.461	0.026	0.136	0.053	0.150
Llama-v3-8b	0.469	0.220	0.695	0.035	0.159	0.081	0.120
StableLM-2-1.6b	0.449	0.109	0.735	0.033	0.138	0.066	0.135
StableLM-3b	0.429	0.145	0.441	0.040	0.177	0.067	0.257
Gemma-2b	0.078	0.140	0.696	0.025	0.145	0.077	0.130
Phi-1.5	0.500	0.155	0.382	0.023	0.094	0.056	0.062
Phi-2	0.429	0.110	0.471	0.064	0.096	0.059	0.259
Phi-3	0.441	0.175	0.549	0.049	0.108	0.104	0.098
Llama-v2-3b-G[2]	0.402	0.275	0.647	0.041	0.173	0.065	0.158
StableLM-3b-G	0.490	0.190	0.559	0.049	0.225	0.098	0.166
TinyLlama-1.1B-G	0.059	0.090	0.176	0.026	0.069	0.043	0.109
Gemma-2b-G	0.402	0.210	0.676	0.026	0.114	0.051	0.107
Llama-v2-7b-G	0.429	0.275	0.696	0.048	0.190	0.147	0.181
Llama-v3-8b-G	0.429	0.210	0.715	0.026	0.205	0.120	0.155
Mistral-7b-G	0.429	0.190	0.343	0.034	0.210	0.079	0.166
OpenChat-3.5-G	0.480	0.210	0.735	0.052	0.182	0.134	0.202
Synthia-7b-G	0.500	0.275	0.373	0.033	0.120	0.000	0.271
Gemma-7b-G	0.402	0.230	0.676	0.026	0.114	0.051	0.112
Llama-13b-G	0.429	0.155	0.696	0.027	0.252	0.068	0.174
S[4]-Llama-1.3b-P[3]	0.294	0.113	0.255	0.028	0.111	0.038	0.176
S-Llama-1.3B	0.429	0.135	0.206	0.030	0.107	0.029	0.122
S-Llama-2.7B-P	0.010	0.120	0.422	0.027	0.128	0.043	0.124
S-Llama-2.7B	0.429	0.220	0.549	0.029	0.174	0.074	0.118

[2]G represents GPTQ, the quantization technique
[3]P represents Pruned
[4]S represents Sheared

A.1.2 Experiments: Trainable Parameters Based on Fixed Alpha and Varying Rank. The experiments for fixed value of alpha to 32 and changing value of *rank* from 8 to 80 have results shown in Figure 10, Figure 11, Figure 12, Figure 13 correspond to Section 3.2.

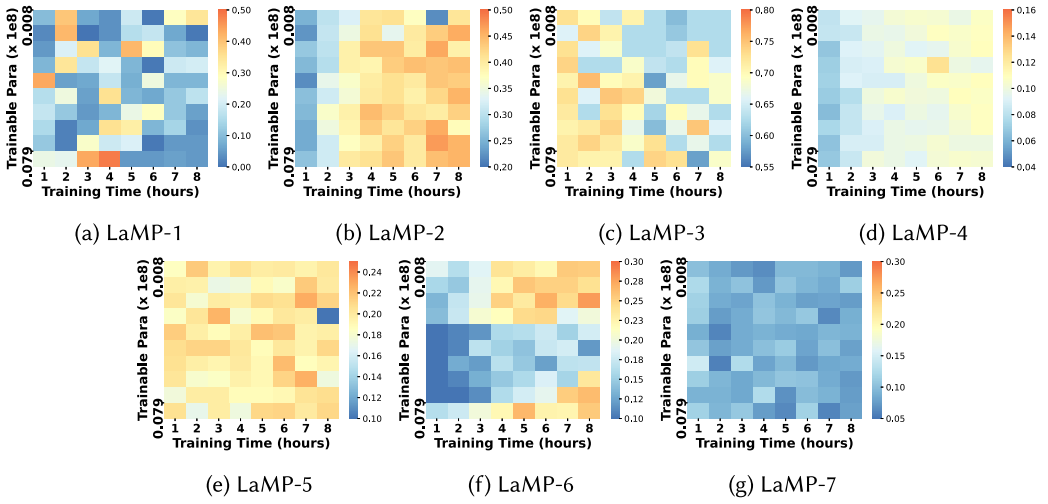


Fig. 10. Performance heatmaps for Pythia-410m on dataset LaMP-1 to LaMP-7, given *alpha* = 32 and *rank* = 8, 16, 24, 32, 40, 48, 56, 72, 80. In each heatmap, we use the number of trainable parameters of different *rank* values as y-axis , and the training hours as the x-axis.

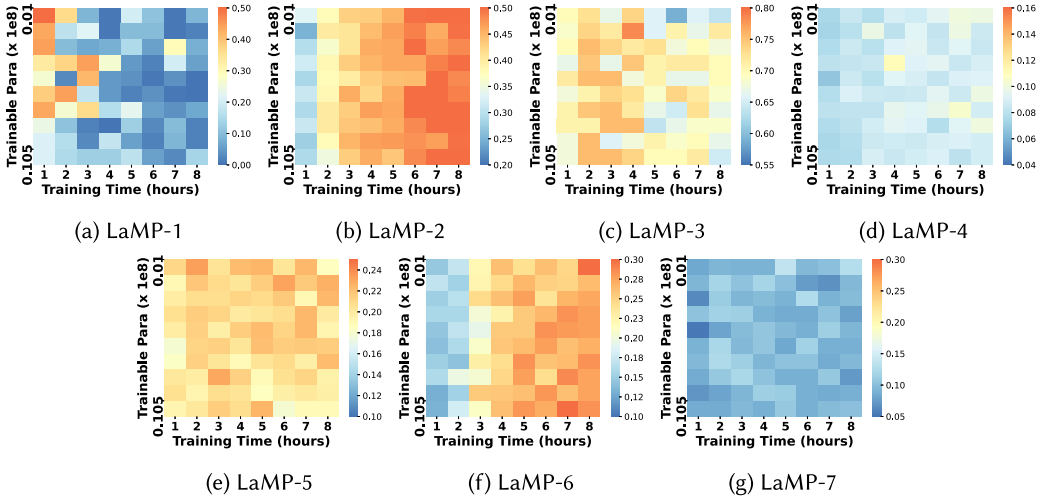
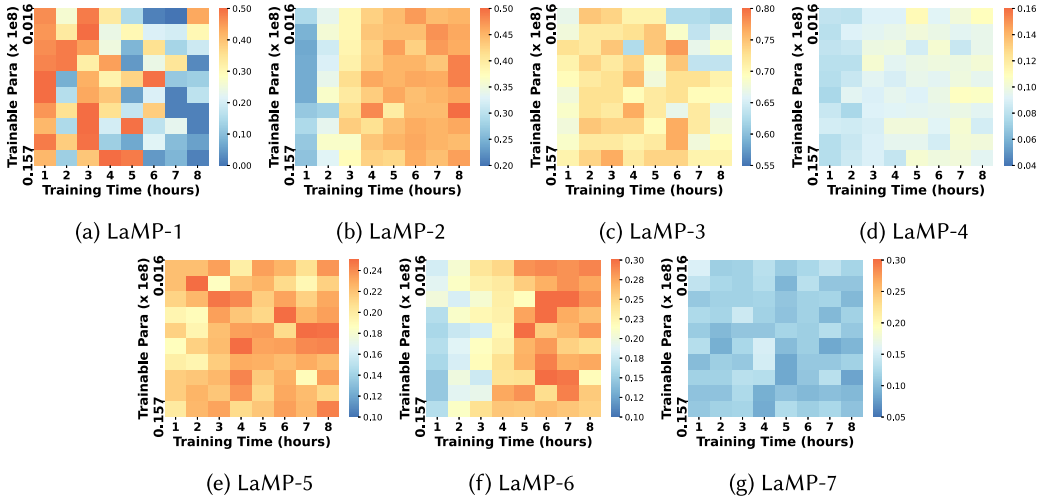


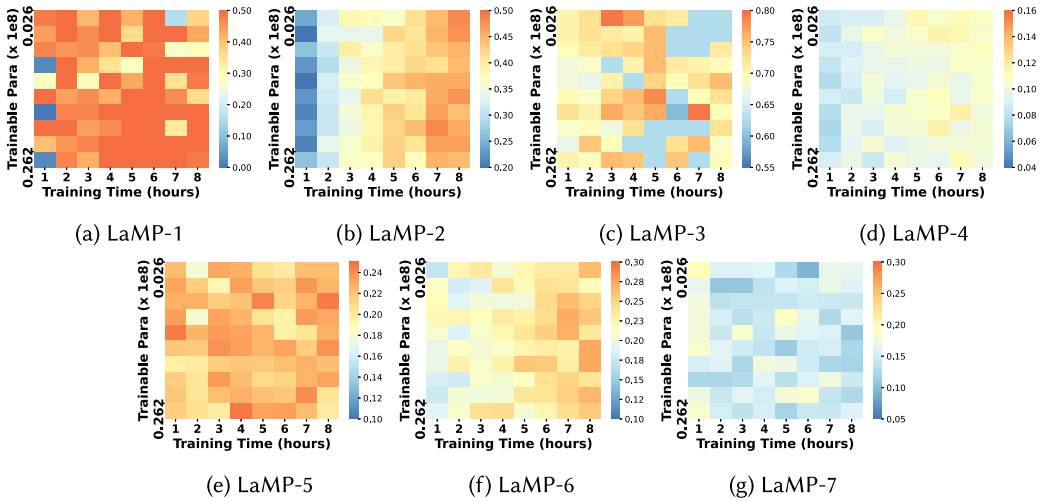
Fig. 11. Performance heatmaps for Pythia-1b on dataset LaMP-1 to LaMP-7, given *alpha* = 32 and *rank* = 8, 16, 24, 32, 40, 48, 56, 72, 80. In each heatmap, we use the number of trainable parameters of different *rank* values as y-axis , and the training hours as the x-axis.



(a) LaMP-1 (b) LaMP-2 (c) LaMP-3 (d) LaMP-4

(e) LaMP-5 (f) LaMP-6 (g) LaMP-7

Fig. 12. Performance heatmaps for Pythia-1.4b on dataset LaMP-1 to LaMP-7, given *alpha* = 32 and *rank* = 8, 16, 24, 32, 40, 48, 56, 72, 80. In each heatmap, we use the number of trainable parameters of different *rank* values as y-axis , and the training hours as the x-axis.



(a) LaMP-1 (b) LaMP-2 (c) LaMP-3 (d) LaMP-4

(e) LaMP-5 (f) LaMP-6 (g) LaMP-7

Fig. 13. Performance heatmaps for Pythia-2.8b on dataset LaMP-1 to LaMP-7, given *alpha* = 32 and *rank* = 8, 16, 24, 32, 40, 48, 56, 72, 80. In each heatmap, we use the number of trainable parameters of different *rank* values as y-axis , and the training hours as the x-axis.

A.1.3 Experiments: Trainable Parameters Based on Varying Aligned Alpha and Rank. The experiments for aligning alpha and rank and changing their values from 8 to 80 have results shown in Figure 14, Figure 15, Figure 16, Figure 17 correspond to Section 3.2.

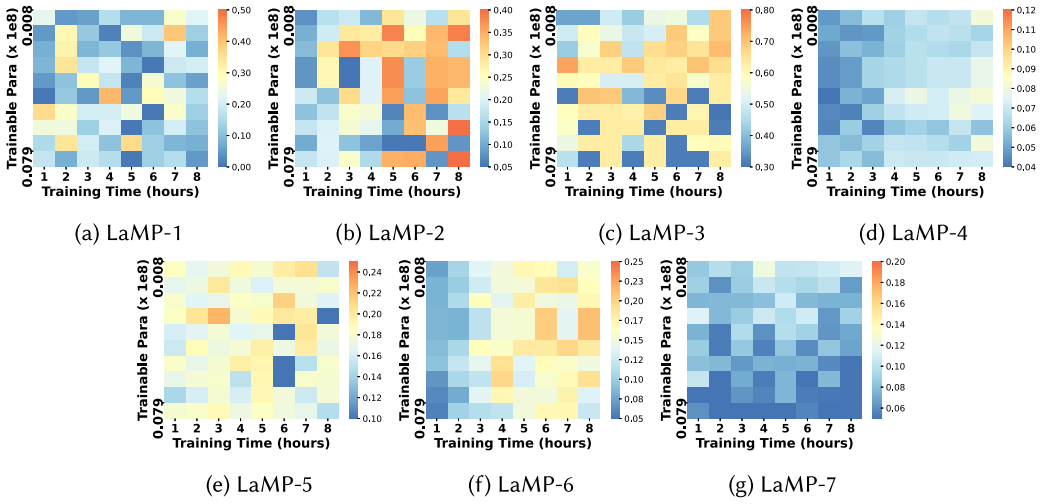


Fig. 14. Performance heatmaps for Pythia-410m on dataset LaMP-1 to LaMP-7, given *alpha* = *rank* = 8, 16, 24, 32, 40, 48, 56, 72, 80. In each heatmap, we use the number of trainable parameters of different *rank* values as y-axis , and the training hours as the x-axis.

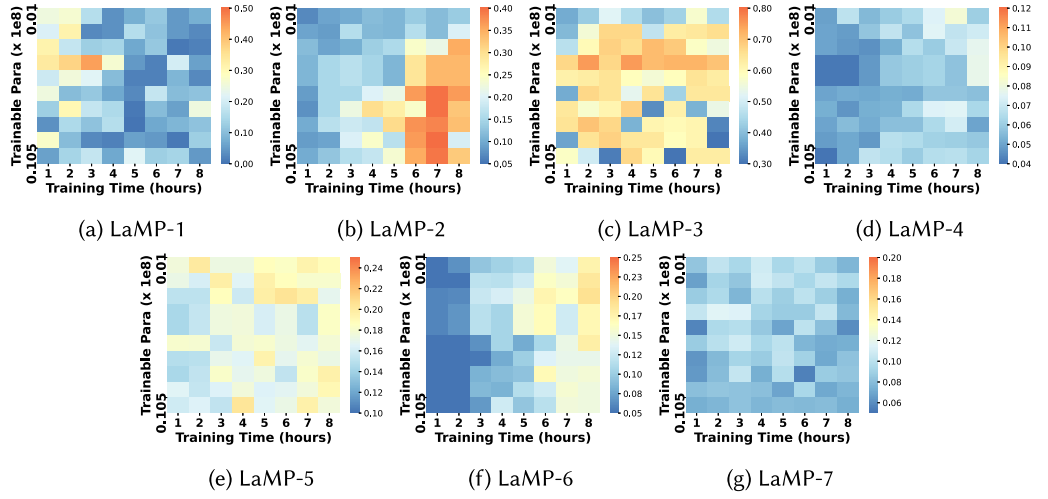
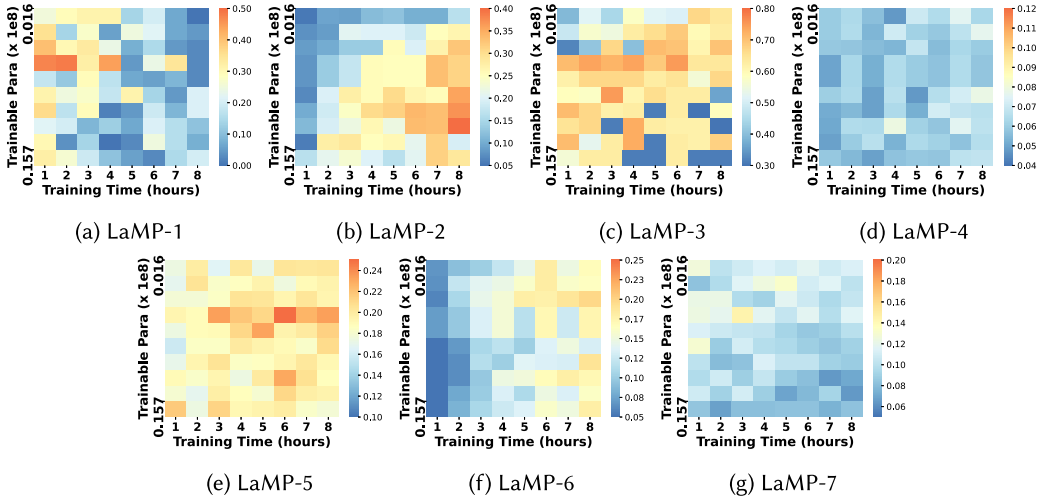


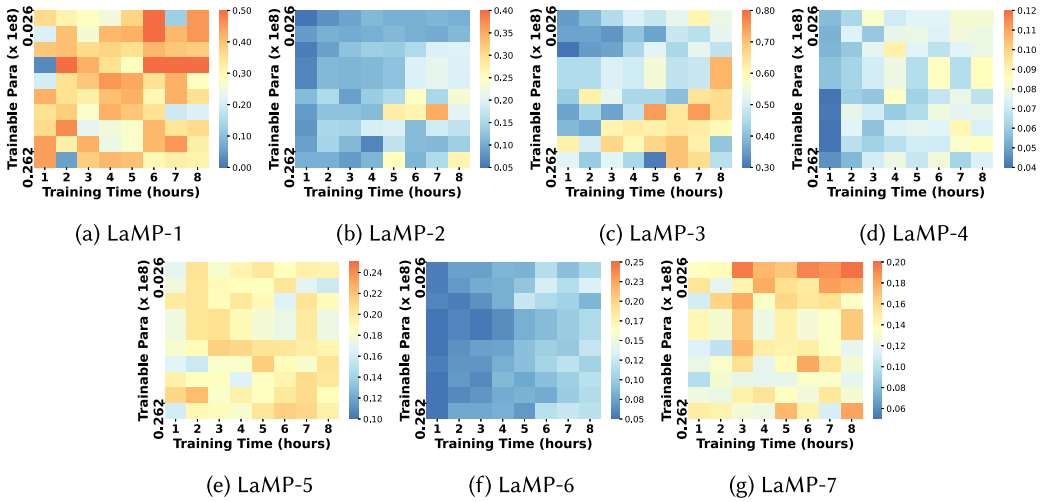
Fig. 15. Performance heatmaps for Pythia-1b on dataset LaMP-1 to LaMP-7, given *alpha* = *rank* = 8, 16, 24, 32, 40, 48, 56, 72, 80. In each heatmap, we use the number of trainable parameters of different *rank* values as y-axis , and the training hours as the x-axis.



(a) LaMP-1 (b) LaMP-2 (c) LaMP-3 (d) LaMP-4

(e) LaMP-5 (f) LaMP-6 (g) LaMP-7

Fig. 16. Performance heatmaps for Pythia-1.4b on dataset LaMP-1 to LaMP-7, given *alpha* = *rank* = 8, 16, 24, 32, 40, 48, 56, 72, 80. In each heatmap, we use the number of trainable parameters of different *rank* values as y-axis , and the training hours as the x-axis.

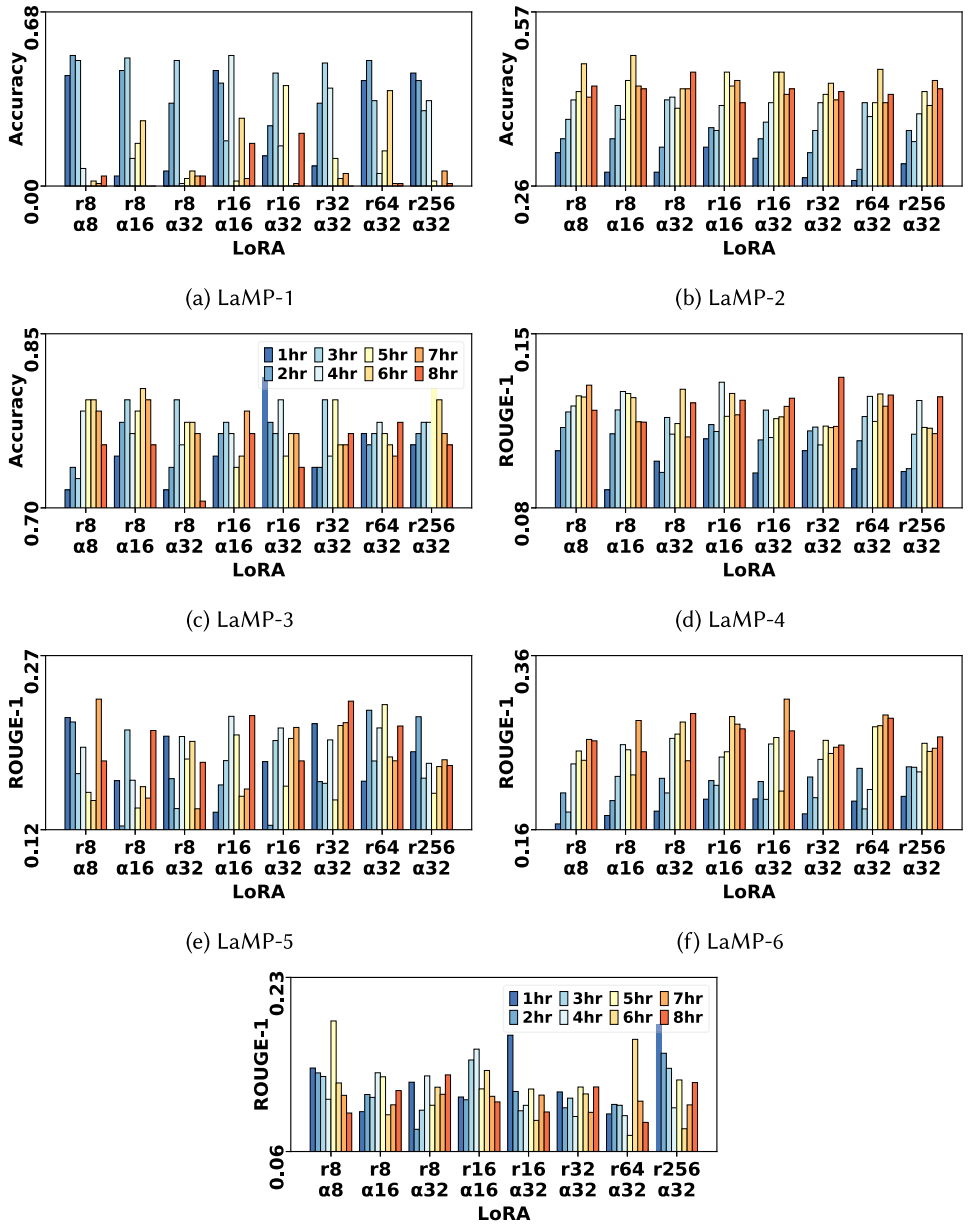


(a) LaMP-1 (b) LaMP-2 (c) LaMP-3 (d) LaMP-4

(e) LaMP-5 (f) LaMP-6 (g) LaMP-7

Fig. 17. Performance heatmaps for Pythia-2.8b on dataset LaMP-1 to LaMP-7, given *alpha* = *rank* = 8, 16, 24, 32, 40, 48, 56, 72, 80. In each heatmap, we use the number of trainable parameters of different *rank* values as y-axis , and the training hours as the x-axis.

A.1.4 Experiments: Combinations of Several Commonly Used Alpha and Rank. The experiments for different combinations of commonly used alpha and rank values as shown in Figure 18 and Figure 19. These experiments can correspond to Section 3.2 and Section 3.3.



(a) LaMP-1

(b) LaMP-2

(c) LaMP-3

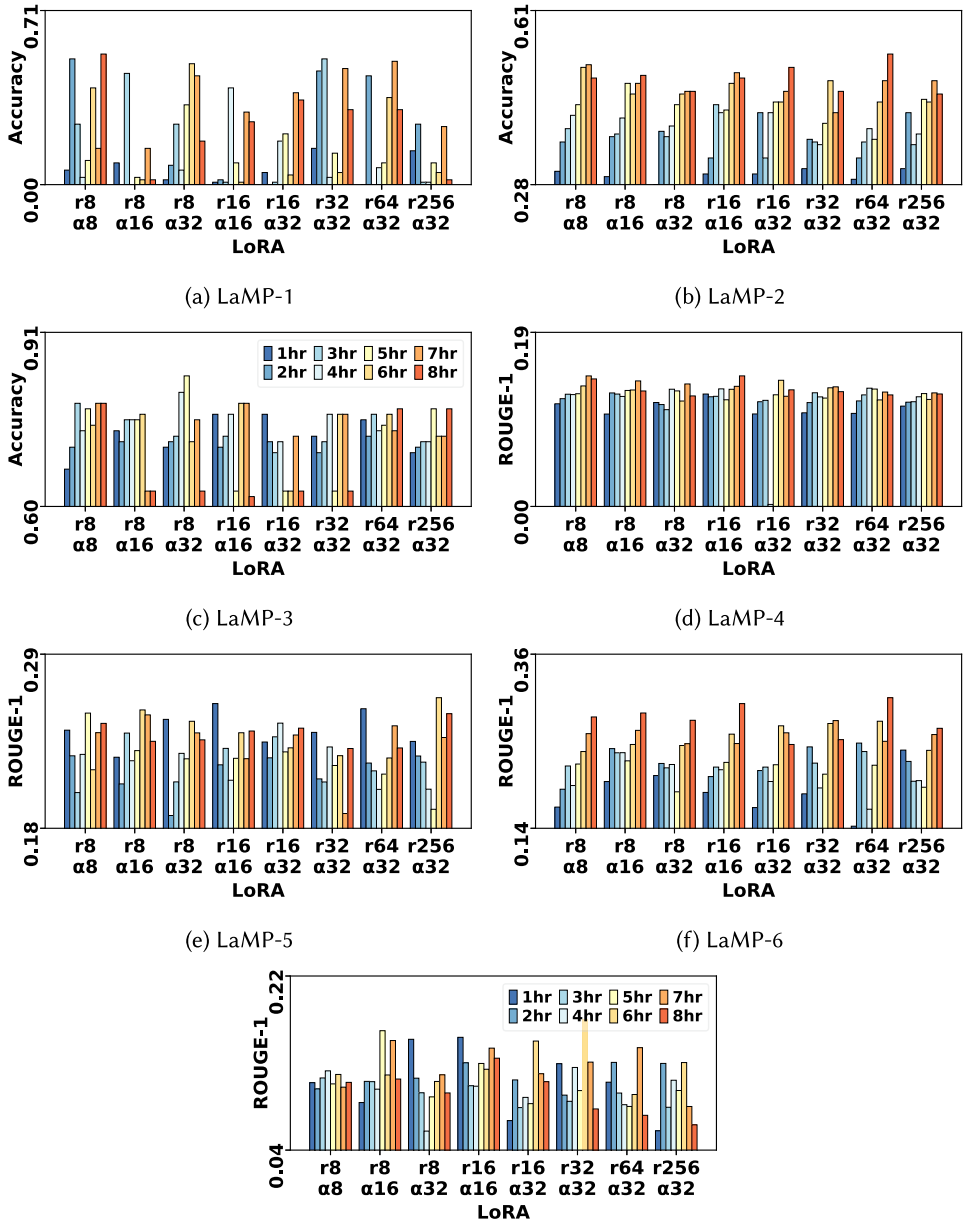
(d) LaMP-4

(e) LaMP-5

(f) LaMP-6

(g) LaMP-7

Fig. 18. Performance comparison for the selected model StableLM-2-1.6b on seven datasets given different common rank and alpha combinations. The experiments are using the default subsec:PEFT_Settings]settings.



(a) LaMP-1

(b) LaMP-2

(c) LaMP-3

(d) LaMP-4

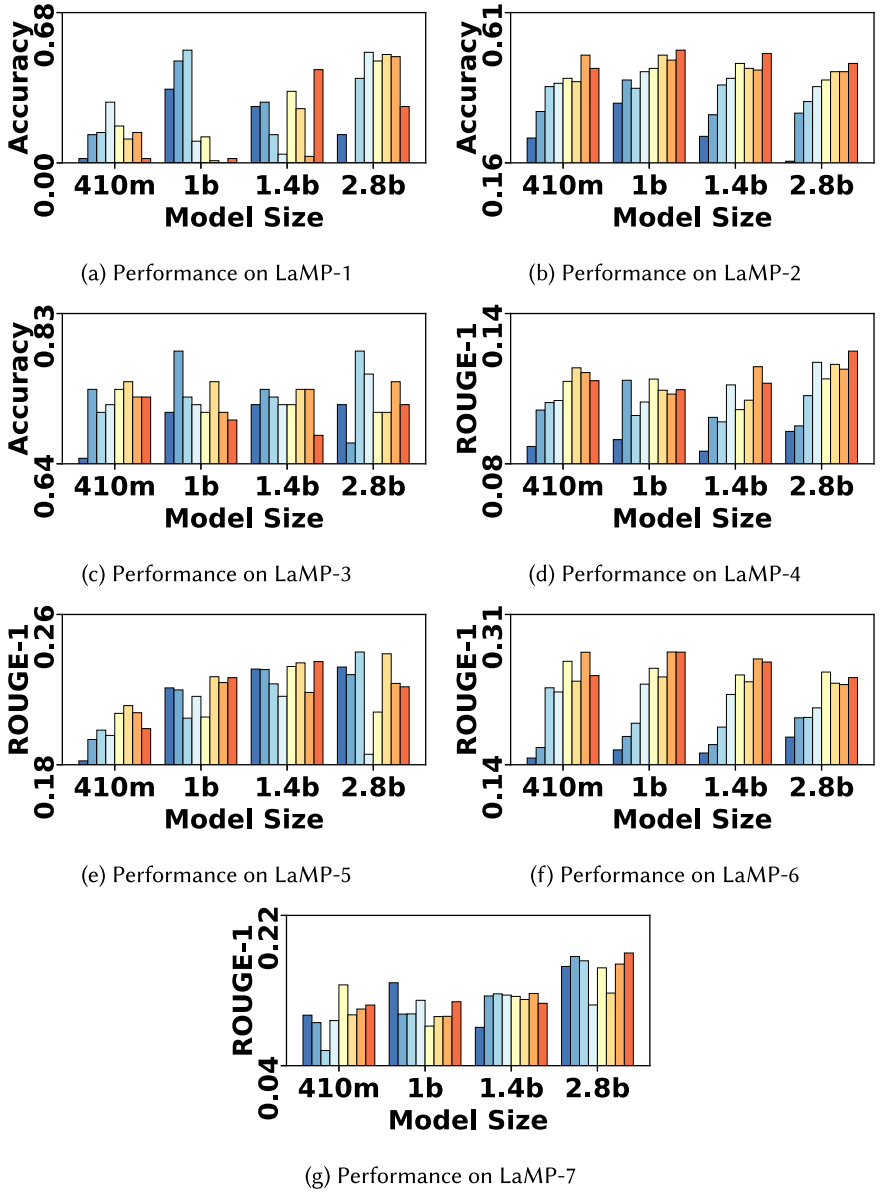
(e) LaMP-5

(f) LaMP-6

(g) LaMP-7

Fig. 19. Performance comparison for the selected model StableLM-3b-4e1t on seven datasets given different common rank and alpha combinations. The experiments are using the default subsec:PEFT_Settings]settings.

A.1.5 Experiments: Model Size and Training Time. The following results shown in Figure 20 and Figure 21 correspond to Section 3.3.



(a) Performance on LaMP-1

(b) Performance on LaMP-2

(c) Performance on LaMP-3

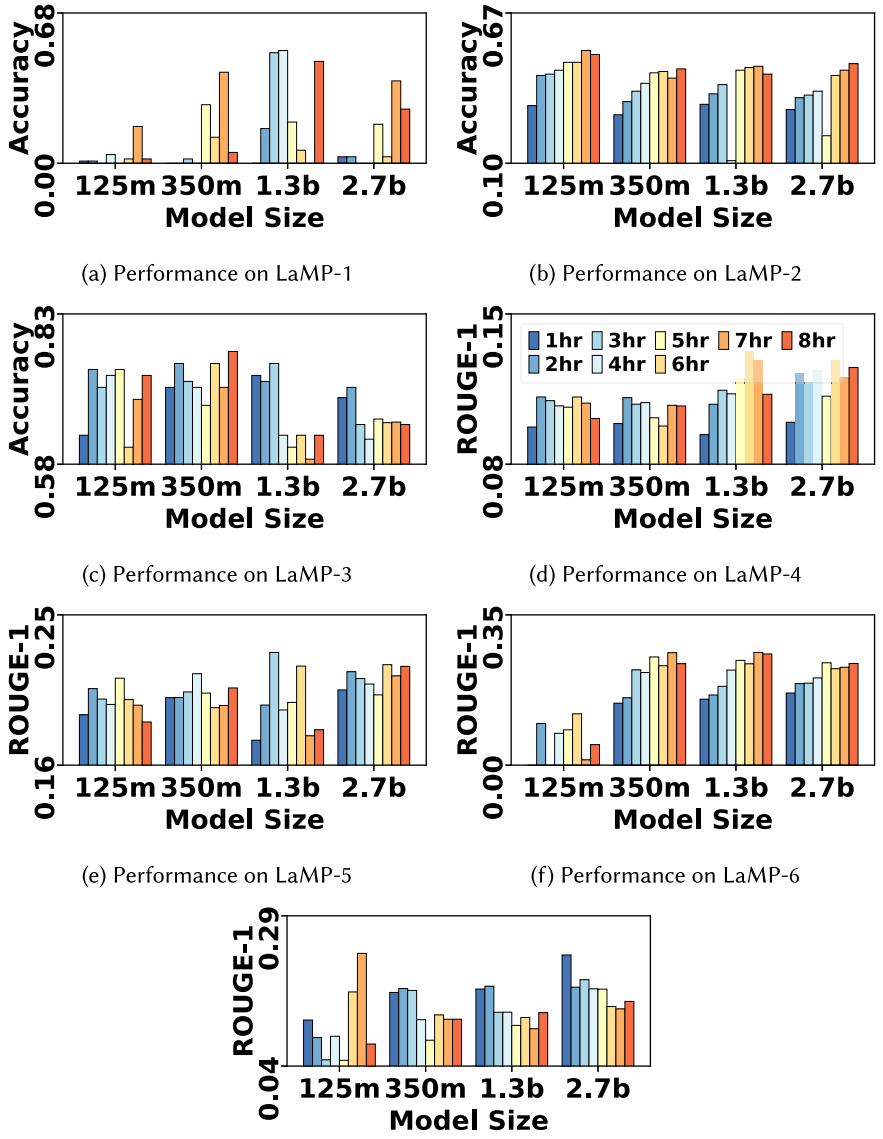
(d) Performance on LaMP-4

(e) Performance on LaMP-5

(f) Performance on LaMP-6

(g) Performance on LaMP-7

Fig. 20. Performance comparison of selected LLMs on different RAM sizes, including Pythia(Py)-410m on RAM 4G, Pythia-1b on RAM 6G, Pythia-1.4b on RAM 8G, StableLM(Sta)-1.6b on RAM 10G, and Pythia-2.8b on RAM 16G. Examine their performance across datasets with different difficulty levels. Legend 1 to 8 represent the number of throughputs, where the detailed data size per unit hour can be found in Table 2 Experiments are based on the default subsec:PEFT_Settings]settings with *rank* = 8 and *alpha* = 8.



(a) Performance on LaMP-1

(b) Performance on LaMP-2

(c) Performance on LaMP-3

(d) Performance on LaMP-4

(e) Performance on LaMP-5

(f) Performance on LaMP-6

(g) Performance on LaMP-7

Fig. 21. Performance comparison of selected LLMs on different RAM sizes, including OPT-125m on RAM 4G, OPT-350m on RAM 6G, OPT-1.3b on RAM 8G, and OPT-2.7b on RAM 16G. Examine their performance across datasets with different difficulty levels. Legend 1 to 8 represent the number of hours of training, where the detailed data size per unit hour can be found in Table 2 Experiments are based on the default subsec:PEFT_Settings]settings with *rank* = 8 and *alpha* = 8.

A.2 Experimental Results for RAG

A.2.1 Experiments: RAG Performance on Different Sizes of User History Data. Below, we present the experiments for the impact of the amount of user history data on RAG performance. These experiments can correspond to Section 3.4

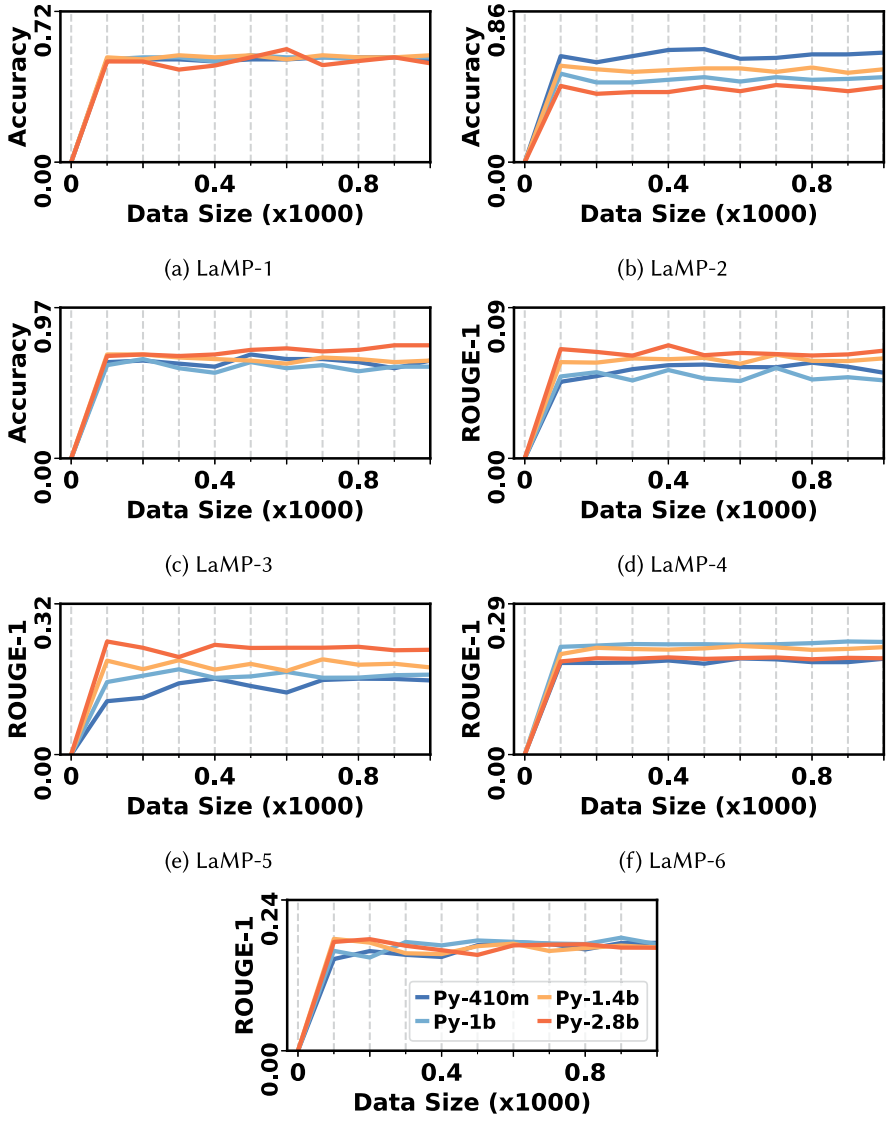


Fig. 22. Performance improvement brought by RAG on four Pythia models of different sizes across different sizes of user history data on all seven datasets.

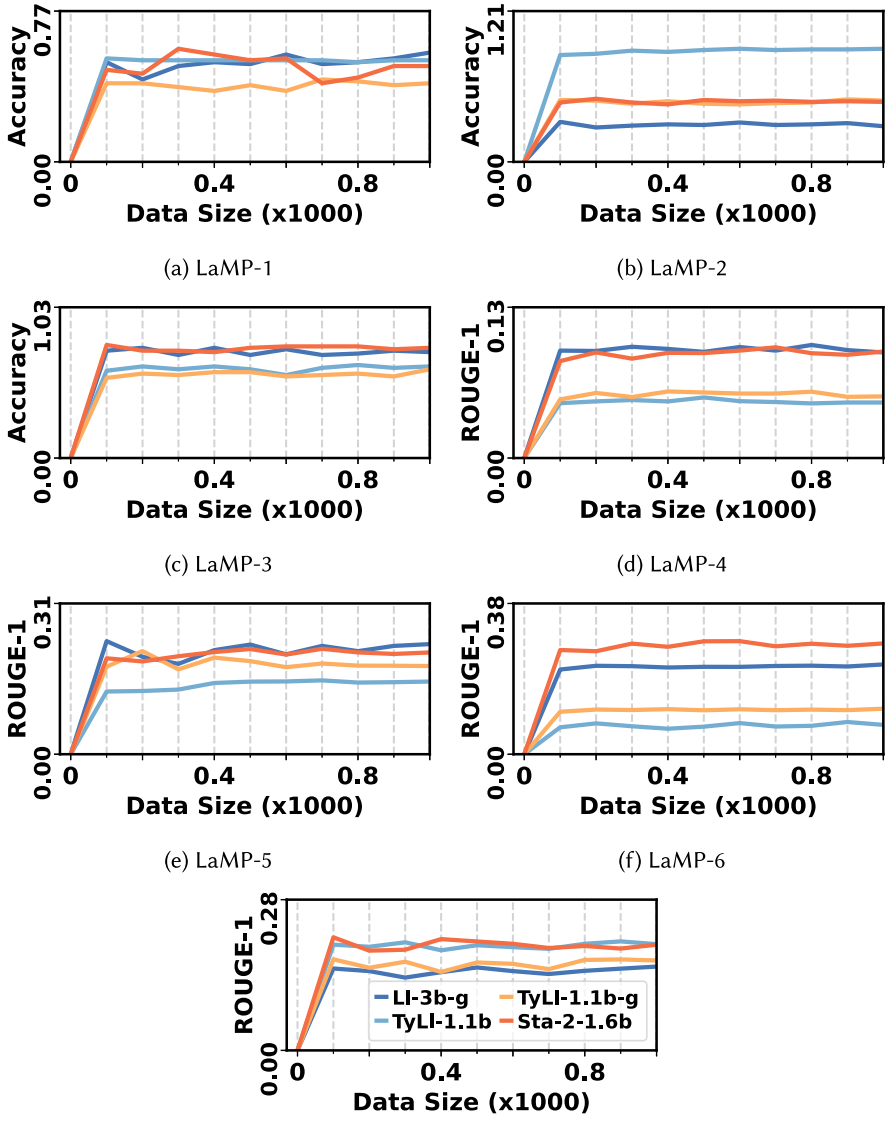


Fig. 23. Performance improvement brought by RAG on four models of different sizes across different sizes of user history data on all seven datasets. The four models are Llama(Ll)-3b-GPTQ(G), StableLM(Sta)-2-1.6B, TinyLlama(TyLl)-1.1B-G, and TinyLlama(TyLl)-1.1B.

B Results of Edge LLM Design

B.1 Experiments: quantization models

Experiments about quantization models only. These experiments can correspond to Section 3.5. We first examine quantized Gemma-2b and Gemma-7b models, showing results in Table 10 and Table 11. Then, we examine the quantized Llama family models, showing results in Figure 24. Additionally, we examine the quantized models with similar size but different architectures, showing results in Figure 25. These experimental results correspond to Section 3.5, serving as the supplemental materials for LLM quantization.

Table 10. Performance Comparisons between Two Quantized Models with Different Sizes After Training 8 Hours

Gemma-GPTQ	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-6		LaMP-7	
	2b	7b	2b	7b	2b	7b	2b	7b	2b	7b	2b	7b	2b	7b
R(8) A(8)	0.520	0.529	0.490	0.420	0.814	0.775	0.124	0.105	0.240	0.213	0.201	0.244	0.227	0.030
R(8) A(16)	0.408	0.451	0.455	0.440	0.794	0.784	0.122	0.099	0.240	0.184	0.220	0.220	0.289	0.032
R(8) A(32)	0.475	0.304	0.440	0.425	0.725	0.784	0.119	0.095	0.209	0.187	0.241	0.228	0.231	0.093
R(16) A(16)	0.570	0.441	0.490	0.400	0.775	0.833	0.125	0.099	0.226	0.203	0.267	0.245	0.209	0.032
R(16) A(32)	0.545	0.510	0.470	0.430	0.765	0.725	0.129	0.092	0.221	0.228	0.224	0.263	0.300	0.052
R(32) A(32)	0.421	0.441	0.505	0.415	0.755	0.755	0.113	0.096	0.202	0.200	0.231	0.221	0.186	0.088
R(64) A(32)	0.520	0.480	0.455	0.435	0.745	0.745	0.108	0.094	0.214	0.190	0.229	0.208	0.190	0.090
R(256) A(32)	0.480	0.520	0.435	0.440	0.765	0.755	0.114	0.094	0.203	0.199	0.218	0.231	0.263	0.051

Table 11. Performance Comparisons between Two Quantized Models with Different Normalized Data Sizes Under the LoRA Setting: *Rank* = 8 and *Alpha* = 32

Gemma-GPTQ	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-6		LaMP-7	
	2b	7b	2b	7b	2b	7b	2b	7b	2b	7b	2b	7b	2b	7b
Training 1 Hour	0.469	0.480	0.265	0.240	0.745	0.706	0.094	0.074	0.223	0.222	0.119	0.160	0.300	0.037
Training 2 Hours	0.445	0.480	0.310	0.285	0.725	0.765	0.104	0.076	0.201	0.204	0.155	0.171	0.283	0.036
Training 3 Hours	0.520	0.471	0.360	0.365	0.784	0.735	0.105	0.073	0.226	0.207	0.205	0.186	0.294	0.048
Training 4 Hours	0.410	0.186	0.415	0.350	0.784	0.775	0.116	0.088	0.205	0.200	0.169	0.195	0.263	0.036
Training 5 Hours	0.441	0.373	0.450	0.305	0.775	0.765	0.122	0.089	0.208	0.215	0.171	0.188	0.294	0.023
Training 6 Hours	0.429	0.480	0.470	0.360	0.765	0.716	0.110	0.079	0.200	0.195	0.226	0.205	0.244	0.090
Training 7 Hours	0.408	0.480	0.465	0.395	0.775	0.775	0.114	0.099	0.243	0.201	0.208	0.177	0.277	0.061
Training 8 Hours	0.421	0.304	0.440	0.425	0.725	0.784	0.119	0.095	0.209	0.187	0.241	0.228	0.231	0.093

[5]G represents GPTQ, the quantization technique

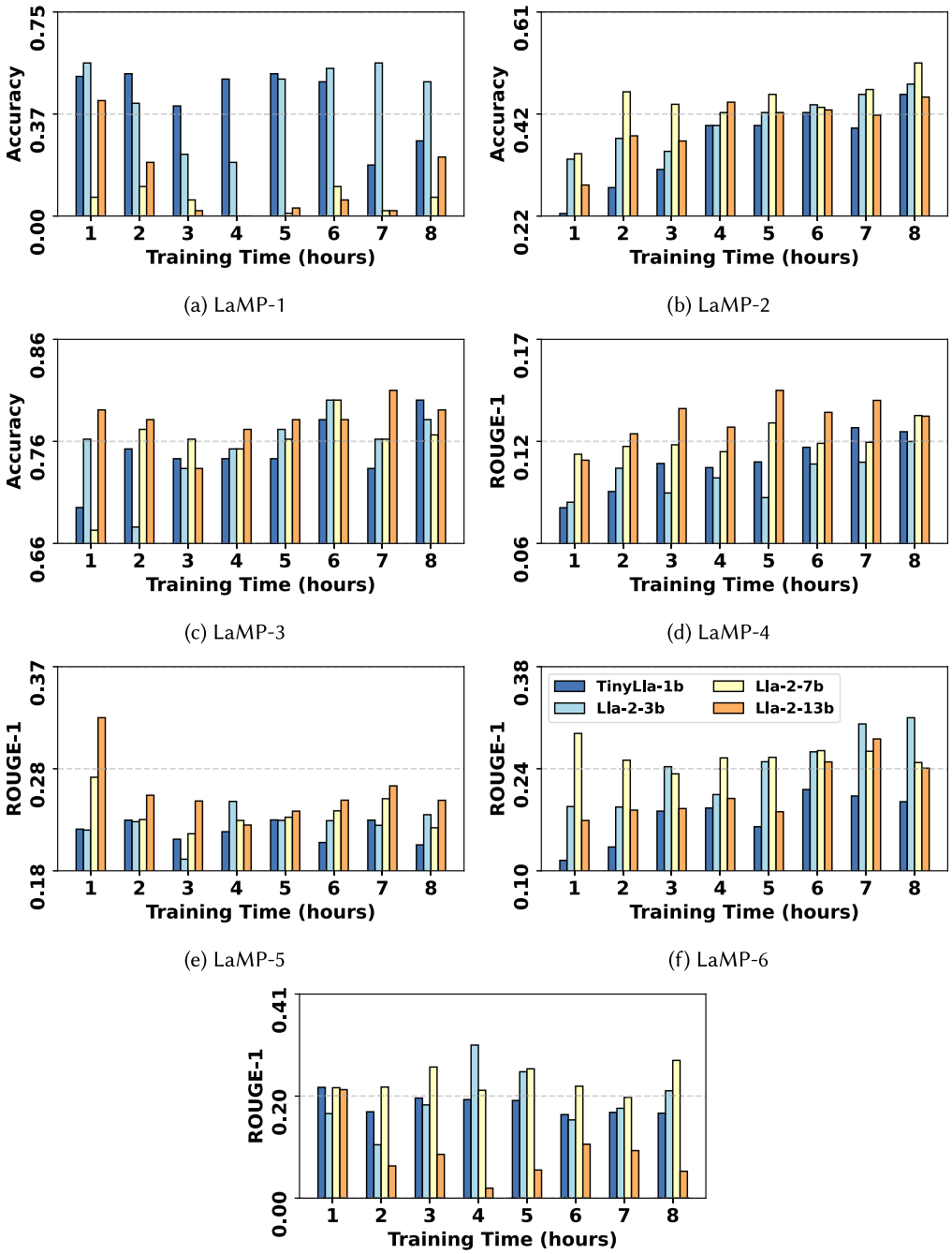
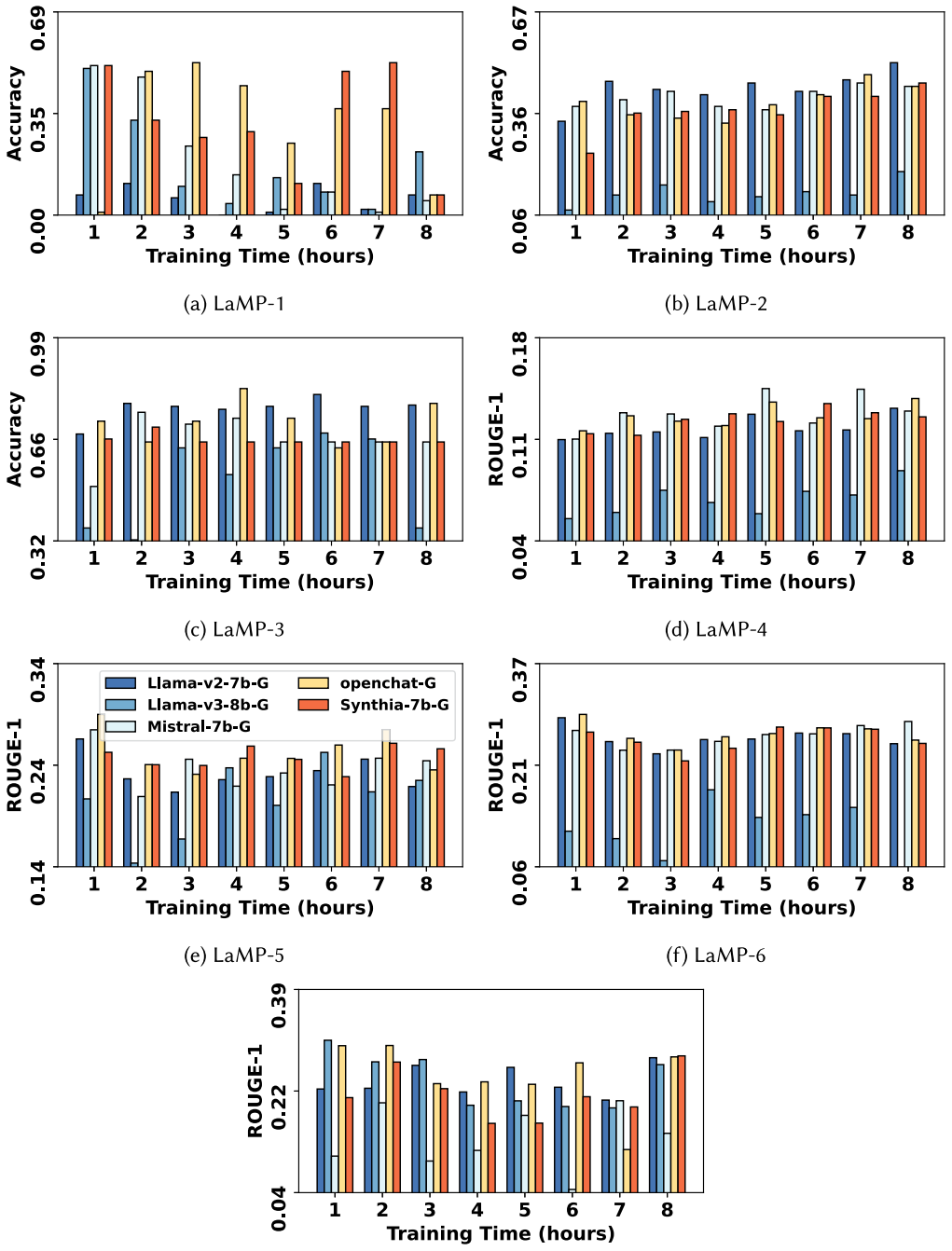


Fig. 24. Performance comparison between the quantized Llama model with different sizes across seven datasets. The learning performance can be observed along with the increase of training data size. We use the default settings and set *rank* = 8 and *alpha* = 32 for LoRA.



(a) LaMP-1

(b) LaMP-2

(c) LaMP-3

(d) LaMP-4

(e) LaMP-5

(f) LaMP-6

(g) LaMP-7

Fig. 25. Performance comparison between the different quantized models with the same size across seven datasets. The learning performance can be observed along with the increase in training data size. We use the default settings and set *rank* = 8 and *alpha* = 32 for LoRA.

B.2 Evaluation LLMs Pruning

Experiments to investigate the pruned models: Sheared-Llama-1.3b-Prune and Sheared-Llama-2.7b-Prune. These experiments can correspond to Section 3.5 and Section 3.6.

We first examine Sheared-Llama-1.3b-Prune (S-Llama-1.3b-P) and Sheared-Llama-1.3b (S-Llama-1.3b) across different data sizes and LoRA settings. The results can be seen in Table 12 and Table 13. Then, we examine Sheared-Llama-1.3b-Prune vs Sheared-Llama-1.3b vs TinyLlama-1.1b. The results can be seen in Figure 26. Additionally, we examine Sheared-Llama-2.7b-Prune vs Sheared-Llama-2.7b across different data size and different lora setting. The results can be seen in Table 14 and Table 15. Finally, we examine Sheared-Llama-2.7b-Prune vs Sheared-Llama-2.7b vs Sheared-Llama-1.3b-Prune vs Sheared-Llama-1.3b vs Llama-3b vs Llama-3b-gptq. The results can be seen in Figure 27.

Table 12. Performance Comparisons between Pruned (+P) and Unpruned (-P) Models with Different LoRA Settings Under Normalized Data Size of 8

Llama-1.1b	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-6		LaMP-7	
	-P[6]	+P	-P	+P	-P	+P	-P	+P	-P	+P	-P	+P	-P	+P
R(8) A(8)	0.461	0.069	0.510	0.410	0.784	0.686	0.113	0.090	0.205	0.215	0.247	0.268	0.126	0.096
R(8) A(16)	0.245	0.118	0.480	0.445	0.765	0.676	0.099	0.097	0.193	0.188	0.298	0.297	0.129	0.071
R(8) A(32)	0.010	0.186	0.500	0.450	0.725	0.696	0.093	0.091	0.221	0.184	0.291	0.281	0.125	0.084
R(16) A(16)	0.108	0.108	0.475	0.430	0.725	0.706	0.099	0.084	0.231	0.184	0.264	0.296	0.119	0.079
R(16) A(32)	0.029	0.098	0.520	0.445	0.784	0.657	0.098	0.079	0.194	0.204	0.258	0.305	0.154	0.065
R(32) A(32)	0.000	0.059	0.480	0.455	0.804	0.686	0.103	0.083	0.208	0.212	0.280	0.268	0.098	0.074
R(64) A(32)	0.108	0.157	0.535	0.435	0.735	0.706	0.105	0.093	0.237	0.212	0.263	0.297	0.099	0.084
R(256) A(32)	0.020	0.137	0.485	0.440	0.784	0.686	0.107	0.079	0.232	0.191	0.288	0.287	0.114	0.100

Table 13. Performance Comparisons between Pruned (+P) and Unpruned (-P) Models Given LoRA Setting Where *Rank* = 8 and *alpha* = 32, Under Training Hours from 1 to 8

Llama-1.1b	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-6		LaMP-7	
	-P	+P	-P	+P	-P	+P	-P	+P	-P	+P	-P	+P	-P	+P
Training 1 Hour	0.069	0.020	0.325	0.220	0.735	0.735	0.091	0.058	0.239	0.188	0.172	0.170	0.192	0.042
Training 2 Hours	0.000	0.039	0.380	0.335	0.755	0.667	0.080	0.074	0.259	0.188	0.169	0.190	0.139	0.060
Training 3 Hours	0.059	0.069	0.445	0.370	0.725	0.755	0.114	0.071	0.233	0.176	0.252	0.232	0.134	0.126
Training 4 Hours	0.049	0.069	0.490	0.400	0.843	0.755	0.103	0.083	0.191	0.186	0.264	0.260	0.126	0.080
Training 5 Hours	0.049	0.049	0.495	0.395	0.775	0.716	0.107	0.076	0.208	0.208	0.234	0.238	0.141	0.086
Training 6 Hours	0.000	0.059	0.480	0.415	0.784	0.647	0.098	0.089	0.204	0.213	0.313	0.288	0.146	0.089
Training 7 Hours	0.010	0.078	0.505	0.440	0.716	0.686	0.097	0.087	0.230	0.196	0.323	0.293	0.113	0.097
Training 8 Hours	0.010	0.186	0.500	0.450	0.725	0.696	0.093	0.091	0.221	0.184	0.291	0.281	0.125	0.084

[6]P represents Pruning

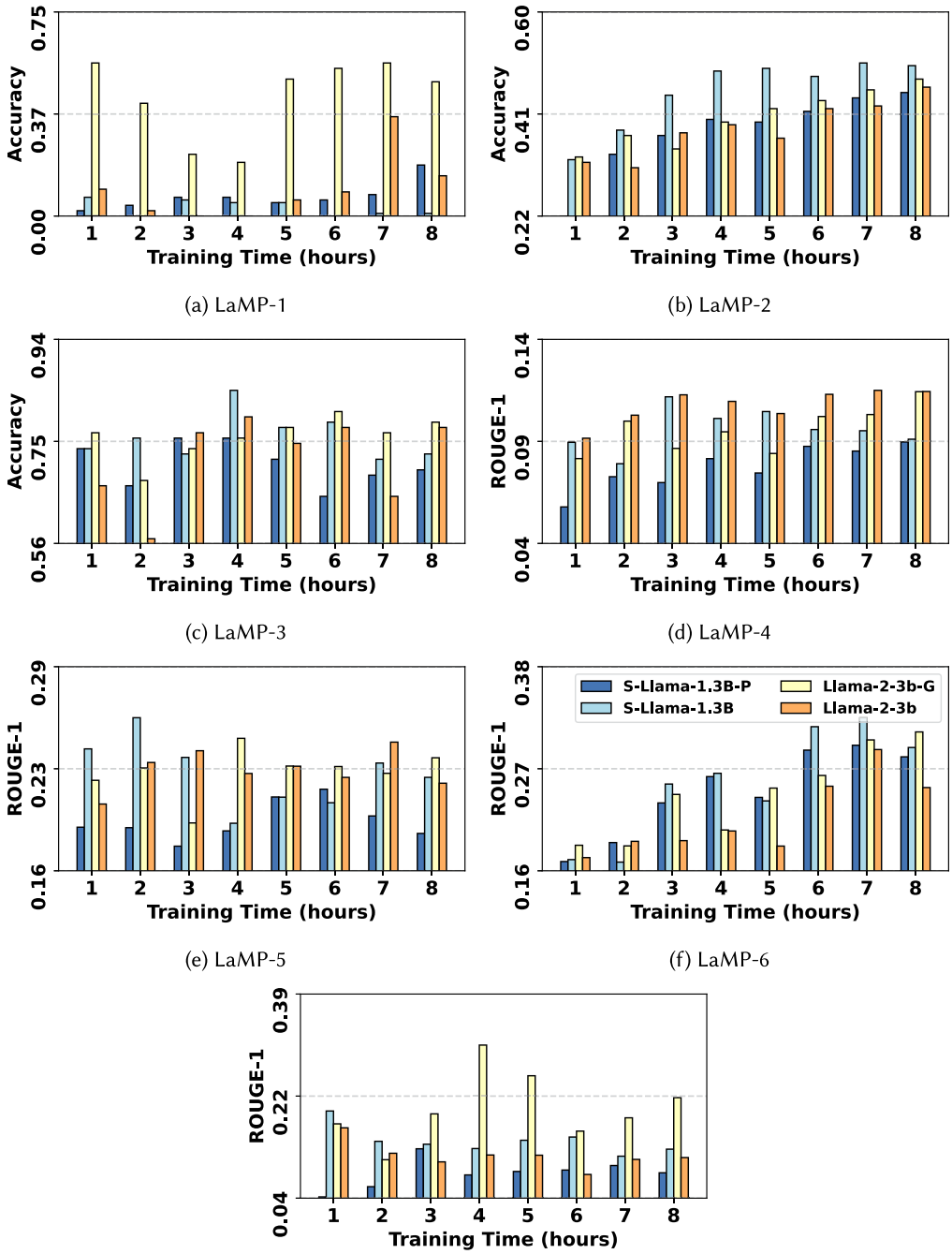
Table 14. Performance Comparisons between Pruned (+P) and Unpruned (-P) Models with Different LoRA Settings Under Training Hours of 8

Sheared-Llama-2.7b	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-6		LaMP-7	
	-P[7]	+P	-P	+P	-P	+P	-P	+P	-P	+P	-P	+P	-P	+P
R(8) A(8)	0.020	0.167	0.480	0.455	0.735	0.725	0.112	0.100	0.213	0.221	0.278	0.319	0.001	0.124
R(8) A(16)	0.069	0.245	0.490	0.455	0.784	0.745	0.109	0.089	0.201	0.206	0.286	0.285	0.000	0.116
R(8) A(32)	0.078	0.059	0.495	0.450	0.755	0.765	0.112	0.089	0.208	0.217	0.332	0.301	0.147	0.133
R(16) A(16)	0.029	0.137	0.495	0.445	0.775	0.755	0.110	0.091	0.181	0.205	0.296	0.302	0.178	0.117
R(16) A(32)	0.039	0.314	0.485	0.460	0.775	0.745	0.103	0.085	0.223	0.209	0.316	0.302	0.151	0.100
R(32) A(32)	0.029	0.088	0.510	0.445	0.784	0.755	0.109	0.102	0.227	0.213	0.329	0.312	0.183	0.093
R(64) A(32)	0.039	0.049	0.495	0.435	0.725	0.716	0.103	0.086	0.208	0.222	0.307	0.314	0.144	0.071
R(256) A(32)	0.020	0.108	0.480	0.420	0.784	0.755	0.112	0.080	0.227	0.219	0.294	0.327	0.184	0.082

Table 15. Performance Comparisons between Pruned (+P) and Unpruned (-P) Models Given LoRA Setting where *rank* = 8 and *alpha* = 32, Under Training Hours from 1 to 8

Sheared-Llama-1.3b	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-6		LaMP-7	
	-P	+P	-P	+P	-P	+P	-P	+P	-P	+P	-P	+P	-P	+P
Training 1 Hour	0.010	0.039	0.320	0.200	0.706	0.676	0.097	0.074	0.212	0.241	0.180	0.093	0.000	0.128
Training 2 Hours	0.059	0.137	0.410	0.285	0.765	0.755	0.100	0.079	0.232	0.219	0.205	0.211	0.000	0.113
Training 3 Hours	0.069	0.010	0.400	0.345	0.804	0.765	0.101	0.072	0.220	0.234	0.219	0.193	0.000	0.087
Training 4 Hours	0.294	0.039	0.415	0.395	0.775	0.775	0.108	0.070	0.216	0.210	0.218	0.185	0.234	0.131
Training 5 Hours	0.029	0.137	0.440	0.460	0.755	0.784	0.101	0.087	0.204	0.216	0.257	0.227	0.128	0.069
Training 6 Hours	0.020	0.069	0.475	0.440	0.755	0.686	0.104	0.083	0.207	0.188	0.278	0.246	0.168	0.123
Training 7 Hours	0.000	0.078	0.505	0.470	0.755	0.794	0.112	0.093	0.202	0.209	0.311	0.304	0.172	0.108
Training 8 Hours	0.078	0.059	0.495	0.450	0.755	0.765	0.112	0.089	0.208	0.217	0.332	0.301	0.147	0.133

[7]P represents Pruning



(a) LaMP-1

(b) LaMP-2

(c) LaMP-3

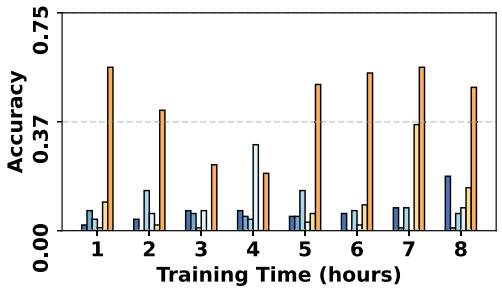
(d) LaMP-4

(e) LaMP-5

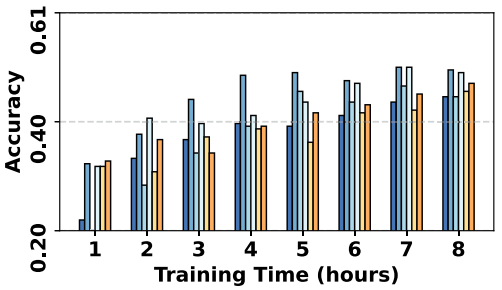
(f) LaMP-6

(g) LaMP-7

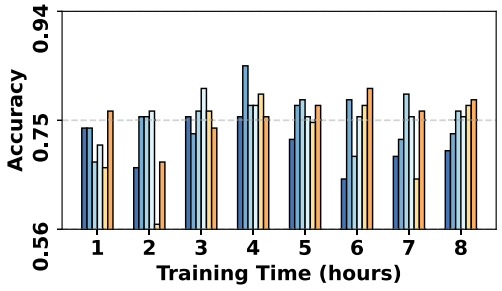
Fig. 26. Performance comparison between the different quantized models with the same size across seven datasets. The learning performance can be observed along with the increase of training data size. We use the default settings and set *rank* = 8 and *alpha* = 32 for LoRA.



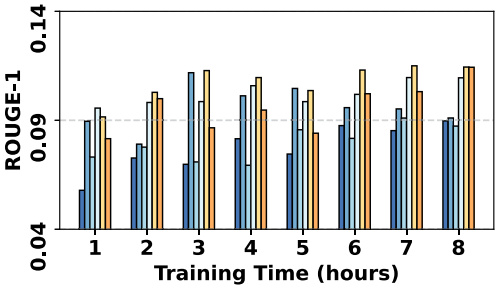
(a) LaMP-1



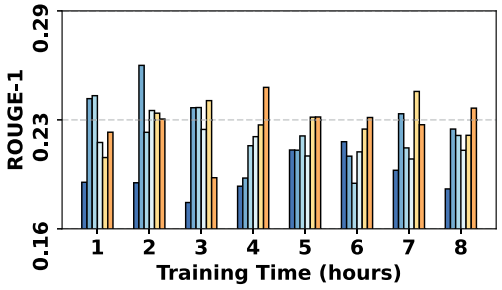
(b) LaMP-2



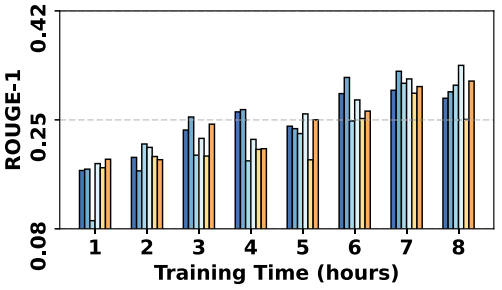
(c) LaMP-3



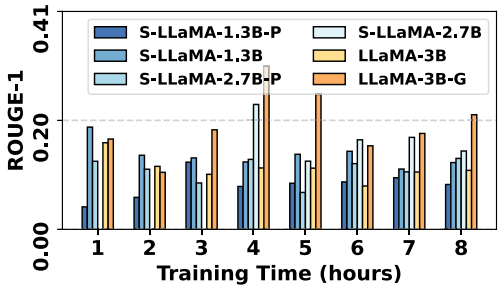
(d) LaMP-4



(e) LaMP-5



(f) LaMP-6



(g) LaMP-7

Fig. 27. Performance comparison between the different quantized models with the same size across seven datasets. The learning performance can be observed along with the increase in training data size. We use the default settings and set *rank* = 8 and *alpha* = 32 for LoRA.

B.3 Evaluation LLMs Distillation

The experiments examine distillation LLM including Phi-1.5, Phi-2, and Phi-3 across different data size and different lora setting. The results can be seen in Table 16 and Table 17. These experiments can correspond to Section 3.5.

Table 16. Performance Comparisons Knowledge Distillation Models Under Different LoRA Settings Under Training Hours of 8

		LaMP-1	LaMP-2	LaMP-3	LaMP-4	LaMP-5	LaMP-6	LaMP-7
Phi-1.5	R(8) A(8)	0.255	0.405	0.696	0.119	0.252	0.265	0.181
	R(8) A(16)	0.500	0.430	0.686	0.115	0.250	0.261	0.159
	R(8) A(32)	0.441	0.435	0.716	0.119	0.224	0.280	0.112
	R(16) A(16)	0.490	0.410	0.706	0.115	0.237	0.277	0.133
	R(16) A(32)	0.137	0.440	0.706	0.114	0.247	0.248	0.171
	R(32) A(32)	0.490	0.420	0.706	0.120	0.256	0.243	0.148
	R(64) A(32)	0.441	0.435	0.706	0.116	0.220	0.255	0.104
	R(256) A(32)	0.402	0.430	0.696	0.117	0.229	0.273	0.139
Phi-2	R(8) A(8)	0.206	0.450	0.745	0.133	0.243	0.241	0.220
	R(8) A(16)	0.529	0.445	0.725	0.132	0.237	0.281	0.221
	R(8) A(32)	0.520	0.465	0.765	0.137	0.264	0.273	0.208
	R(16) A(16)	0.373	0.435	0.745	0.130	0.253	0.254	0.219
	R(16) A(32)	0.529	0.455	0.765	0.141	0.259	0.285	0.208
	R(32) A(32)	0.529	0.455	0.775	0.137	0.247	0.297	0.209
	R(64) A(32)	0.529	0.450	0.784	0.134	0.240	0.290	0.215
	R(256) A(32)	0.520	0.440	0.745	0.135	0.232	0.268	0.191
Phi-3	R(8) A(8)	0.471	0.470	0.784	0.103	0.211	0.198	0.199
	R(8) A(16)	0.529	0.450	0.804	0.102	0.209	0.248	0.176
	R(8) A(32)	0.520	0.470	0.794	0.109	0.224	0.221	0.159
	R(16) A(16)	0.549	0.455	0.833	0.100	0.210	0.226	0.142
	R(16) A(32)	0.529	0.465	0.745	0.098	0.210	0.222	0.197
	R(32) A(32)	0.216	0.505	0.775	0.094	0.218	nan	0.125
	R(64) A(32)	0.255	0.480	0.804	0.090	0.233	0.125	0.191
	R(256) A(32)	0.402	0.490	0.833	0.094	0.252	0.129	0.127

Table 17. Performance Comparisons between Knowledge Distillation Models Under Given LoRA Setting Where *rank* = 8 and *alpha* = 32, Under Training Hours from 1 to 8

		LaMP-1	LaMP-2	LaMP-3	LaMP-4	LaMP-5	LaMP-6	LaMP-7
Phi-1.5	Training 1 Hour	0.108	0.255	0.696	0.084	0.219	0.163	0.166
	Training 2 Hours	0.010	0.280	0.676	0.095	0.226	0.178	0.165
	Training 3 Hours	0.108	0.390	0.716	0.100	0.249	0.180	0.135
	Training 4 Hours	0.441	0.370	0.686	0.106	0.239	0.226	0.163
	Training 5 Hours	0.382	0.415	0.676	0.119	0.218	0.234	0.098
	Training 6 Hours	0.216	0.380	0.716	0.116	0.232	0.258	0.145
	Training 7 Hours	0.480	0.435	0.716	0.114	0.247	0.285	0.095
	Training 8 Hours	0.441	0.435	0.716	0.119	0.224	0.280	0.112
Phi-2	Training 1 Hour	0.539	0.295	0.706	0.108	0.228	0.156	0.179
	Training 2 Hours	0.529	0.315	0.735	0.116	0.249	0.157	0.235
	Training 3 Hours	0.529	0.330	0.706	0.129	0.228	0.217	0.156
	Training 4 Hours	0.529	0.350	0.765	0.126	0.204	0.190	0.211
	Training 5 Hours	0.529	0.375	0.794	0.126	0.236	0.210	0.164
	Training 6 Hours	0.529	0.420	0.765	0.136	0.251	0.249	0.164
	Training 7 Hours	0.471	0.425	0.755	0.127	0.231	0.274	0.227
	Training 8 Hours	0.520	0.465	0.765	0.137	0.264	0.273	0.208
Phi-3	Training 1 Hour	0.529	0.325	0.765	0.094	0.213	0.158	0.091
	Training 2 Hours	0.431	0.315	0.775	0.076	0.199	0.205	0.107
	Training 3 Hours	0.500	0.395	0.725	0.100	0.208	0.200	0.130
	Training 4 Hours	0.569	0.385	0.775	0.088	0.202	0.192	0.111
	Training 5 Hours	0.549	0.385	0.775	0.095	0.215	0.186	0.145
	Training 6 Hours	0.510	0.470	0.814	0.092	0.221	0.215	0.121
	Training 7 Hours	0.441	0.425	0.755	0.102	0.245	0.260	0.114
	Training 8 Hours	0.520	0.470	0.794	0.109	0.224	0.221	0.159

B.4 Experiments: Compare the three compression techniques

The experiments examine how each compressed model can perform compared with each other. We select a wide range of models from each compression technique. For each task, we use the same heatmap scale to show the optimal model and condition. These experiments correspond to the Section 3.5.

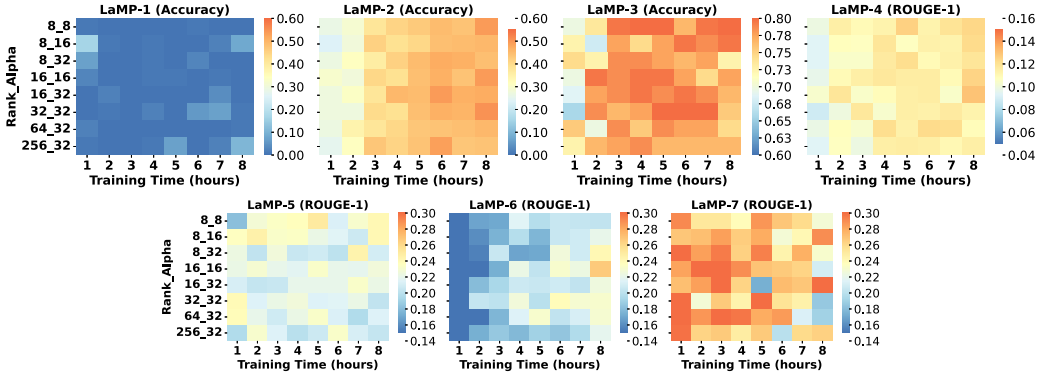


Fig. 28. Performances of **Gemma-2b-GPTQ** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

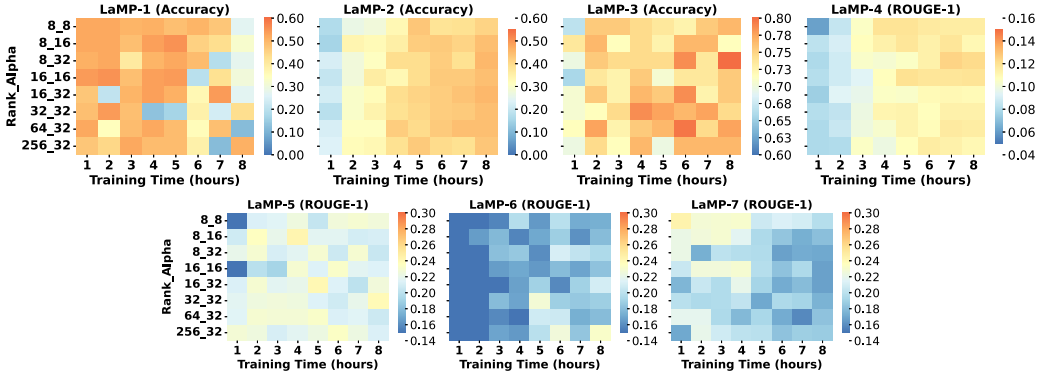


Fig. 29. Performances of **TinyLlama-1.1B-GPTQ** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

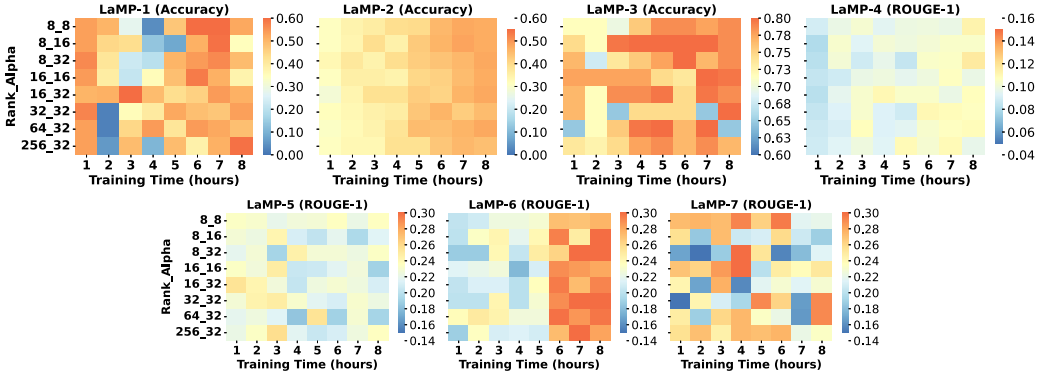


Fig. 30. Performances of **Llama-2-3b-GPTQ** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

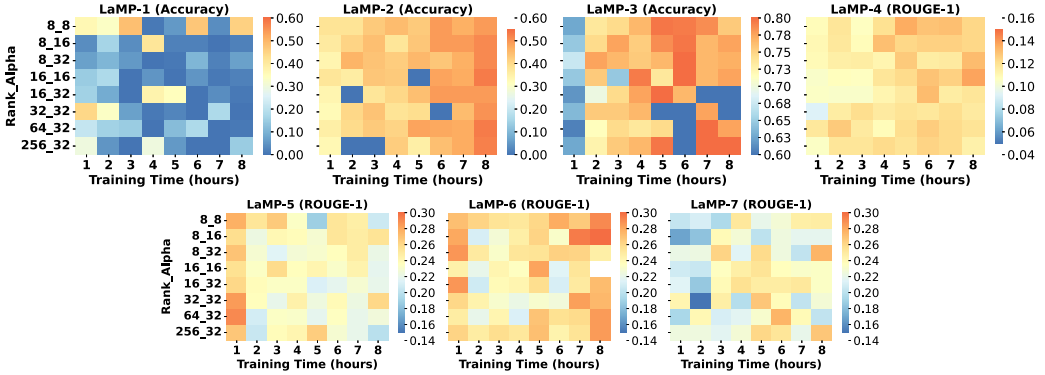


Fig. 31. Performances of **Llama-2-7B-GPTQ** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

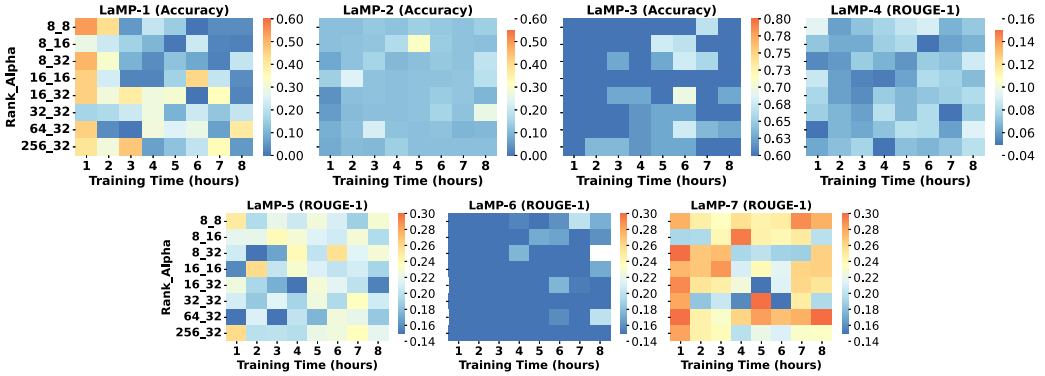


Fig. 32. Performances of **Llama-3-8B-GPTQ** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

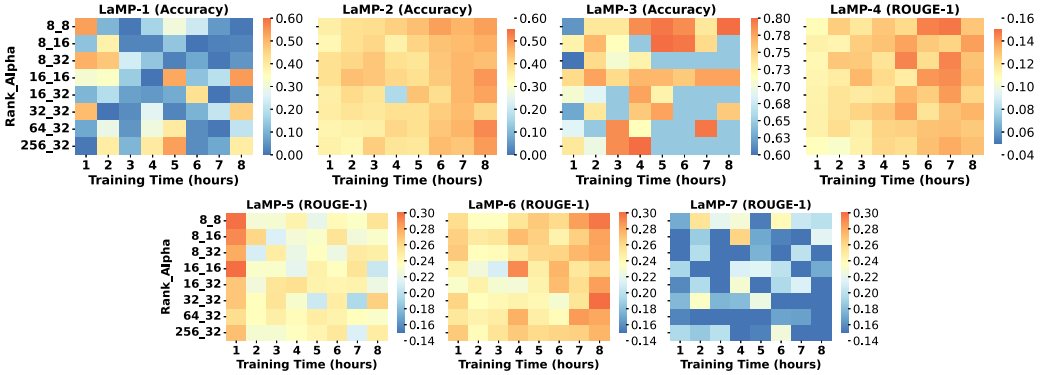


Fig. 33. Performances of **Mistral-7b-GPTQ** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

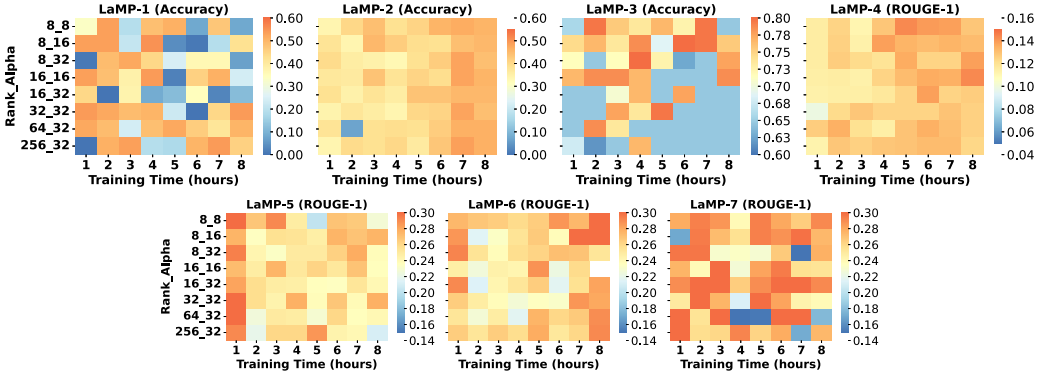


Fig. 34. Performances of **OpenChat-3.5-GPTQ** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

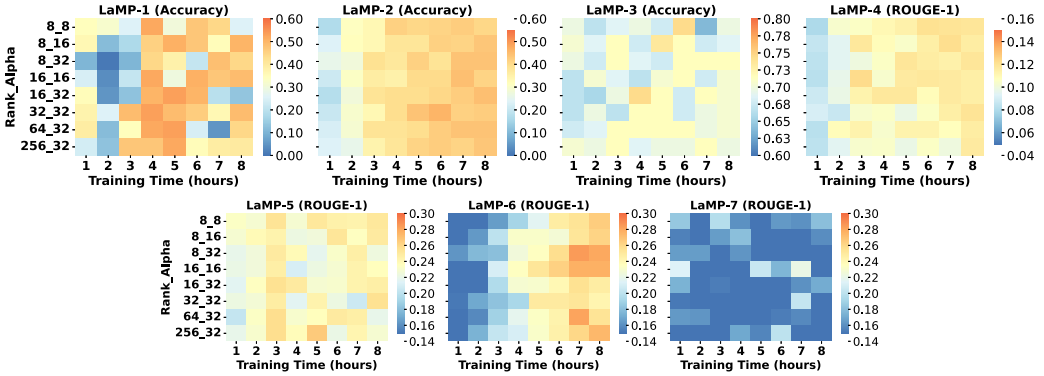


Fig. 35. Performances of **Phi-1.5** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

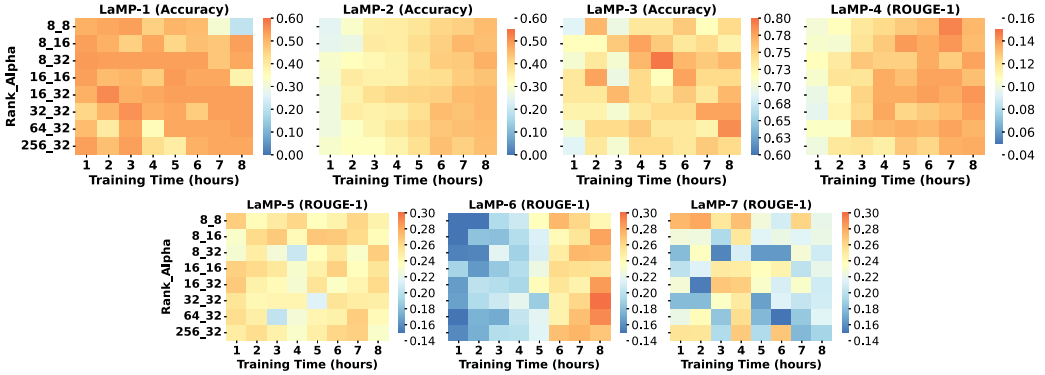


Fig. 36. Performances of **Phi-2** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

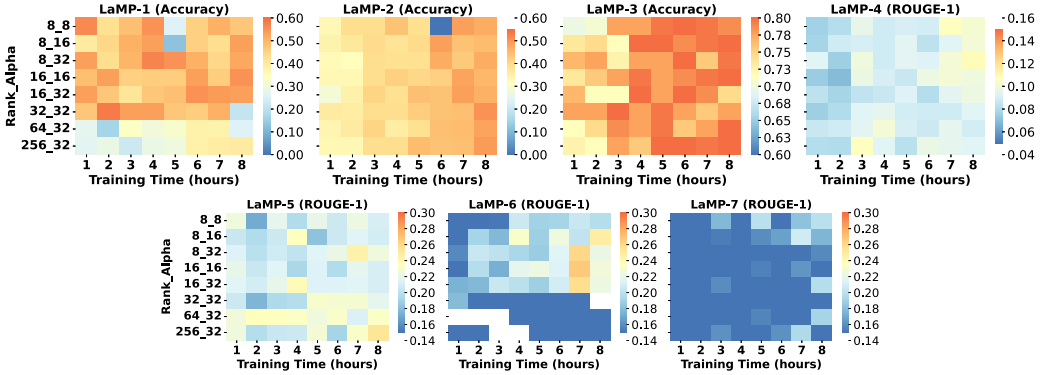


Fig. 37. Performances of **Phi-3** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

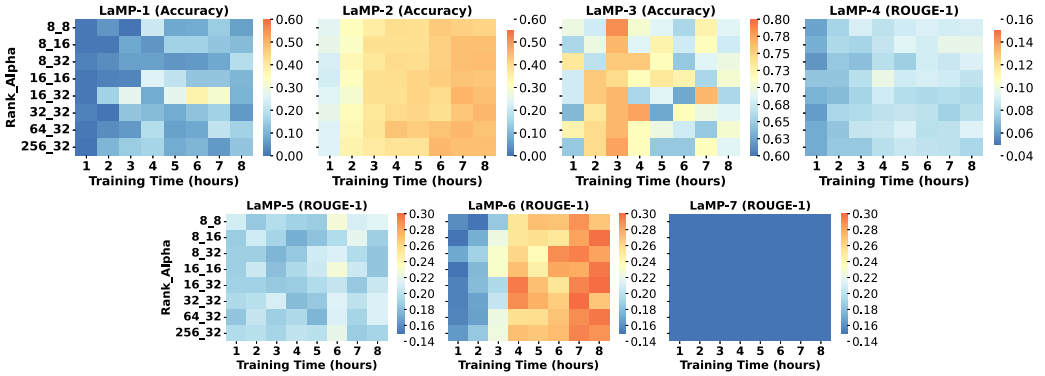


Fig. 38. Performances of **Sheared-LLaMA-1.3B-Pruned** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

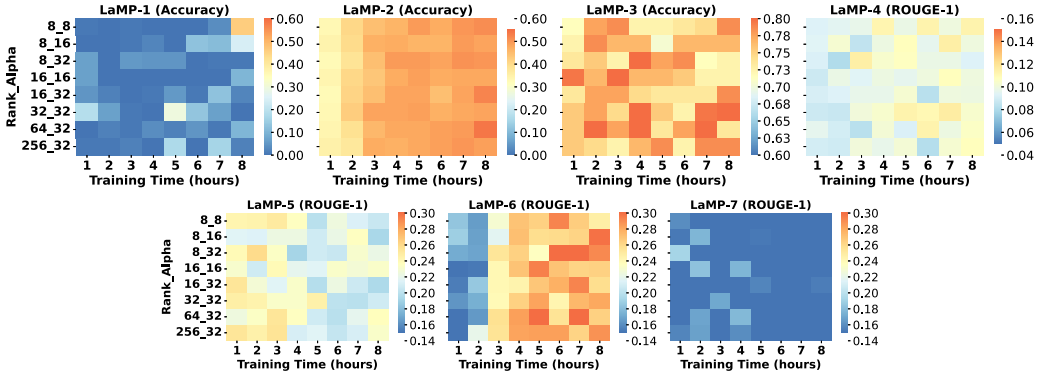


Fig. 39. Performances of **Sheared-LLaMA-1.3B** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

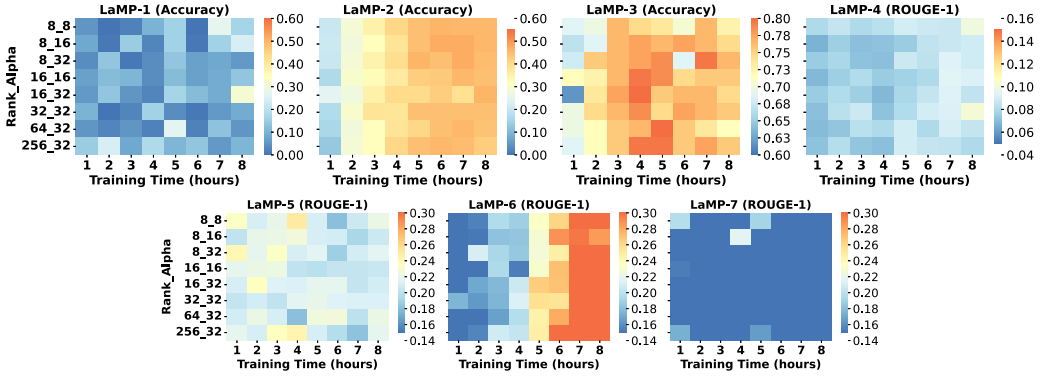


Fig. 40. Performances of **Sheared-LLaMA-2.7B-Pruned** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

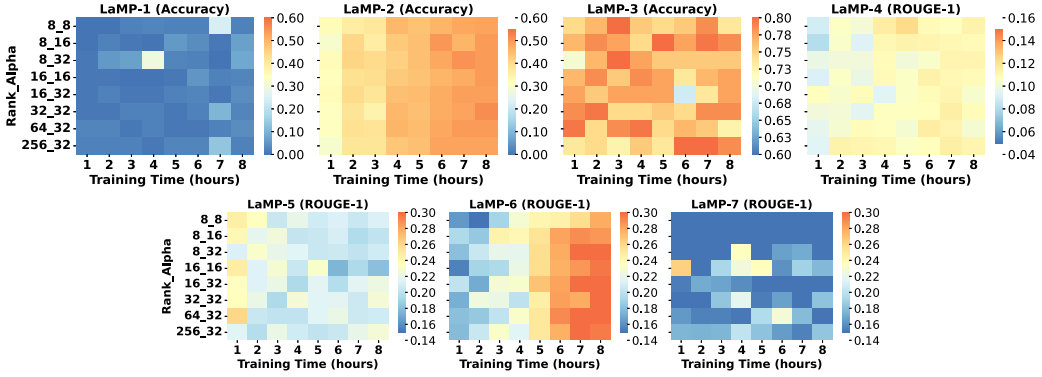


Fig. 41. Performances of **Sheared-LLaMA-2.7B** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

B.5 Experiments: compare the compressed model with uncompressed models

These experiments provide additional study concentrating on quantization due to the popular trend of this technique. These experiments can correspond to Section 3.6. This section includes Table 18, Table 19, Figure 42, Figure 43, and Figure 44

Table 18. Performance Comparisons between Quantized (+G) and Quantized (-G) Models with Different LoRA Settings Under Training Hours of 8

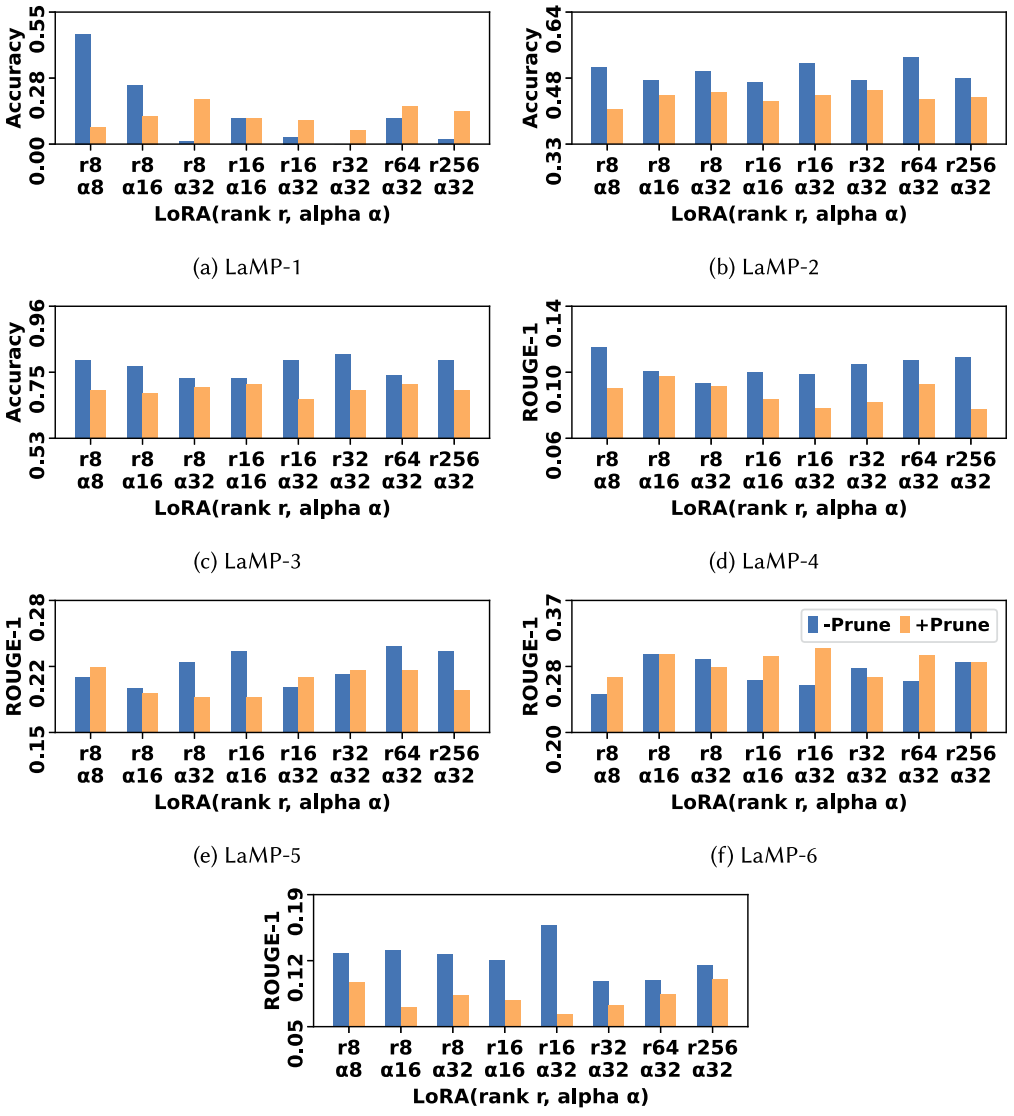
Llama-v2-3b	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-6		LaMP-7	
	-G[8]	+G	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G
R(8) A(8)	0.020	0.520	0.430	0.470	0.765	0.784	0.129	0.113	0.252	0.234	0.264	0.275	0.204	0.220
R(8) A(16)	0.010	0.343	0.475	0.470	0.814	0.784	0.118	0.104	0.251	0.216	0.241	0.312	0.132	0.189
R(8) A(32)	0.147	0.490	0.460	0.475	0.775	0.784	0.117	0.117	0.217	0.233	0.248	0.307	0.111	0.215
R(16) A(16)	0.294	0.373	0.450	0.470	0.833	0.794	0.116	0.098	0.230	0.190	0.270	0.280	0.097	0.251
R(16) A(32)	0.010	0.510	0.470	0.475	0.804	0.784	0.114	0.108	0.232	0.211	0.294	0.291	0.061	0.229
R(32) A(32)	0.000	0.529	0.460	0.450	0.647	0.804	0.117	0.106	0.215	0.223	0.263	0.311	0.109	0.291
R(64) A(32)	0.196	0.480	0.455	0.475	0.794	0.647	0.115	0.109	0.235	0.191	0.263	0.296	0.125	0.290
R(256) A(32)	0.324	0.588	0.465	0.440	0.765	0.765	0.112	0.096	0.244	0.234	nan	0.281	0.095	0.233
Llama-v3-8b	**LaMP-1**		**LaMP-2**		**LaMP-3**		**LaMP-4**		**LaMP-5**		**LaMP-6**		**LaMP-7**	
	-G[8]	+G	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G
R(8) A(8)	0.324	0.039	0.130	0.140	0.627	0.461	0.065	0.073	0.180	0.229	0.172	0.174	0.189	0.277
R(8) A(16)	0.529	0.020	0.125	0.115	0.676	0.559	0.088	0.067	0.246	0.195	0.166	0.162	0.128	0.199
R(8) A(32)	0.088	0.216	0.150	0.190	0.363	0.363	0.091	0.088	0.147	0.227	0.080	0.134	0.089	0.264
R(16) A(16)	0.265	0.039	0.115	0.180	0.637	0.412	0.094	0.070	0.186	0.192	0.179	0.172	0.155	0.263
R(16) A(32)	0.049	0.029	0.110	0.095	0.608	0.627	0.074	0.081	0.190	0.155	0.172	0.134	0.079	0.253
R(32) A(32)	0.029	0.216	0.135	0.260	0.324	0.363	0.057	0.075	0.203	0.207	0.097	0.110	0.076	0.194
R(64) A(32)	0.098	0.402	0.115	0.100	0.627	0.627	0.069	0.092	0.211	0.185	0.158	0.200	0.135	0.306
R(256) A(32)	0.010	0.059	0.115	0.110	0.588	0.363	0.089	0.082	0.199	0.218	nan	0.095	0.089	0.233
Gemma-2b	**LaMP-1**		**LaMP-2**		**LaMP-3**		**LaMP-4**		**LaMP-5**		**LaMP-6**		**LaMP-7**	
	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G
R(8) A(8)	0.010	0.000	0.430	0.490	0.784	0.814	0.119	0.124	0.239	0.240	0.229	0.201	0.301	0.227
R(8) A(16)	0.000	0.078	0.465	0.455	0.794	0.794	0.126	0.122	0.202	0.240	0.239	0.220	0.165	0.289
R(8) A(32)	0.000	0.000	0.445	0.440	0.794	0.725	0.125	0.119	0.242	0.209	0.239	0.241	0.135	0.231
R(16) A(16)	0.039	0.000	0.460	0.490	0.755	0.775	0.121	0.125	0.211	0.226	0.248	0.267	0.194	0.209
R(16) A(32)	0.020	0.000	0.445	0.470	0.755	0.765	0.121	0.129	0.227	0.221	0.246	0.224	0.173	0.300
R(32) A(32)	0.059	0.010	0.450	0.505	0.804	0.755	0.121	0.113	0.234	0.202	0.220	0.231	0.164	0.186
R(64) A(32)	0.029	0.010	0.485	0.455	0.794	0.745	0.125	0.108	0.208	0.214	nan	0.229	0.238	0.190
R(256) A(32)	0.000	0.108	0.440	0.435	0.794	0.765	0.121	0.114	0.225	0.203	nan	0.218	0.193	0.263
StabLM-3b	**LaMP-1**		**LaMP-2**		**LaMP-3**		**LaMP-4**		**LaMP-5**		**LaMP-6**		**LaMP-7**	
	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G
R(8) A(8)	0.529	0.490	0.480	0.440	0.784	0.725	0.136	0.118	0.246	0.234	0.280	0.226	0.108	0.200
R(8) A(16)	0.020	0.529	0.485	0.485	0.627	0.725	0.123	0.101	0.234	0.205	0.285	0.245	0.112	0.176
R(8) A(32)	0.176	0.520	0.455	0.440	0.627	0.775	0.118	0.105	0.235	0.251	0.276	0.269	0.098	0.218
R(16) A(16)	0.255	0.471	0.480	0.425	0.618	0.755	0.139	0.115	0.241	0.207	0.297	0.266	0.133	0.231
R(16) A(32)	0.343	0.431	0.500	0.485	0.627	0.725	0.124	0.104	0.243	0.232	0.245	0.262	0.109	0.237
R(32) A(32)	0.304	0.549	0.455	0.420	0.627	0.627	0.122	0.105	0.230	0.184	0.251	0.269	0.082	0.219
R(64) A(32)	0.304	0.520	0.525	0.435	0.775	0.775	0.119	0.109	0.230	0.218	0.304	0.246	0.075	0.201
R(256) A(32)	0.020	0.392	0.450	0.450	0.775	0.627	0.120	0.111	0.252	0.226	0.266	0.268	0.066	0.204

[8]G represents GPTQ, the quantization technique

Table 19. Performance Comparisons between Quantized (+G) and Quantized (-G) Models Given LoRA Setting Where *rank* = 8 and *alpha* = 32, Under Training Hours from 1 to 8

Llama-v2-3b	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-6		LaMP-7	
	-G[9]	+G	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G
Training 1 Hour	0.098	0.559	0.320	0.330	0.667	0.765	0.093	0.083	0.203	0.219	0.174	0.187	0.162	0.169
Training 2 Hours	0.020	0.412	0.310	0.370	0.569	0.676	0.105	0.102	0.230	0.227	0.191	0.186	0.118	0.107
Training 3 Hours	0.000	0.225	0.375	0.345	0.765	0.735	0.115	0.088	0.238	0.191	0.192	0.241	0.103	0.187
Training 4 Hours	0.000	0.196	0.390	0.395	0.794	0.755	0.112	0.097	0.223	0.246	0.202	0.203	0.115	0.306
Training 5 Hours	0.059	0.500	0.365	0.420	0.745	0.775	0.106	0.086	0.228	0.228	0.186	0.248	0.115	0.253
Training 6 Hours	0.088	0.539	0.420	0.435	0.775	0.804	0.116	0.104	0.221	0.228	0.250	0.261	0.081	0.157
Training 7 Hours	0.363	0.559	0.425	0.455	0.647	0.765	0.118	0.105	0.243	0.223	0.289	0.299	0.107	0.180
Training 8 Hours	0.147	0.490	0.460	0.475	0.775	0.784	0.117	0.117	0.217	0.233	0.248	0.307	0.111	0.215
Llama-v3-8b	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-6		LaMP-7	
	-G[9]	+G	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G
Training 1 Hour	0.353	0.500	0.065	0.075	0.569	0.363	0.077	0.055	0.263	0.208	0.052	0.114	0.087	0.306
Training 2 Hours	0.304	0.324	0.095	0.120	0.569	0.324	0.066	0.060	0.239	0.144	0.067	0.103	0.107	0.268
Training 3 Hours	0.098	0.098	0.110	0.150	0.696	0.627	0.060	0.075	0.174	0.168	0.106	0.069	0.115	0.272
Training 4 Hours	0.020	0.039	0.120	0.100	0.627	0.539	0.066	0.066	0.157	0.240	0.112	0.177	0.089	0.193
Training 5 Hours	0.020	0.127	0.125	0.115	0.627	0.627	0.083	0.059	0.253	0.202	0.088	0.135	0.103	0.200
Training 6 Hours	0.029	0.078	0.100	0.130	0.559	0.676	0.088	0.074	0.155	0.255	0.151	0.139	0.077	0.190
Training 7 Hours	0.039	0.020	0.115	0.120	0.627	0.657	0.064	0.072	0.132	0.216	0.130	0.150	0.082	0.188
Training 8 Hours	0.088	0.216	0.150	0.190	0.363	0.363	0.091	0.088	0.147	0.227	0.080	0.134	0.089	0.264
Gemma-2b	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-6		LaMP-7	
	-G[9]	+G	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G
Training 1 Hour	0.010	0.069	0.275	0.265	0.696	0.745	0.102	0.094	0.231	0.223	0.127	0.119	0.154	0.300
Training 2 Hours	0.098	0.000	0.320	0.310	0.755	0.725	0.113	0.104	0.226	0.201	0.159	0.155	0.140	0.283
Training 3 Hours	0.000	0.000	0.390	0.360	0.706	0.784	0.112	0.105	0.208	0.226	0.176	0.205	0.165	0.294
Training 4 Hours	0.186	0.010	0.390	0.415	0.794	0.784	0.114	0.116	0.233	0.205	0.207	0.169	0.139	0.263
Training 5 Hours	0.010	0.000	0.460	0.450	0.784	0.775	0.116	0.122	0.213	0.208	0.197	0.171	0.167	0.294
Training 6 Hours	0.010	0.029	0.435	0.470	0.784	0.765	0.115	0.110	0.223	0.200	0.213	0.226	0.141	0.244
Training 7 Hours	0.039	0.000	0.480	0.465	0.824	0.775	0.118	0.114	0.200	0.243	0.217	0.208	0.153	0.277
Training 8 Hours	0.000	0.000	0.445	0.440	0.794	0.725	0.125	0.119	0.242	0.209	0.239	0.241	0.135	0.231
StableLM-3b	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-6		LaMP-7	
	-G[9]	+G	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G	-G	+G
Training 1 Hour	0.020	0.480	0.280	0.325	0.706	0.676	0.111	0.088	0.248	0.200	0.206	0.191	0.152	0.137
Training 2 Hours	0.078	0.510	0.380	0.285	0.716	0.716	0.108	0.094	0.188	0.179	0.221	0.210	0.113	0.128
Training 3 Hours	0.245	0.539	0.370	0.370	0.725	0.716	0.103	0.103	0.209	0.223	0.216	0.211	0.098	0.152
Training 4 Hours	0.059	0.500	0.390	0.365	0.804	0.784	0.125	0.094	0.227	0.193	0.220	0.227	0.059	0.149
Training 5 Hours	0.324	0.510	0.430	0.380	0.833	0.735	0.123	0.101	0.223	0.208	0.186	0.234	0.094	0.235
Training 6 Hours	0.490	0.510	0.450	0.380	0.716	0.765	0.112	0.113	0.247	0.238	0.244	0.244	0.109	0.180
Training 7 Hours	0.441	0.471	0.455	0.390	0.755	nan	0.130	0.105	0.240	0.229	0.246	0.242	0.116	0.227
Training 8 Hours	0.176	0.520	0.455	0.440	0.627	0.775	0.118	0.105	0.235	0.251	0.276	0.269	0.098	0.218

[9]G represents GPTQ, the quantization technique



(a) LaMP-1

(b) LaMP-2

(c) LaMP-3

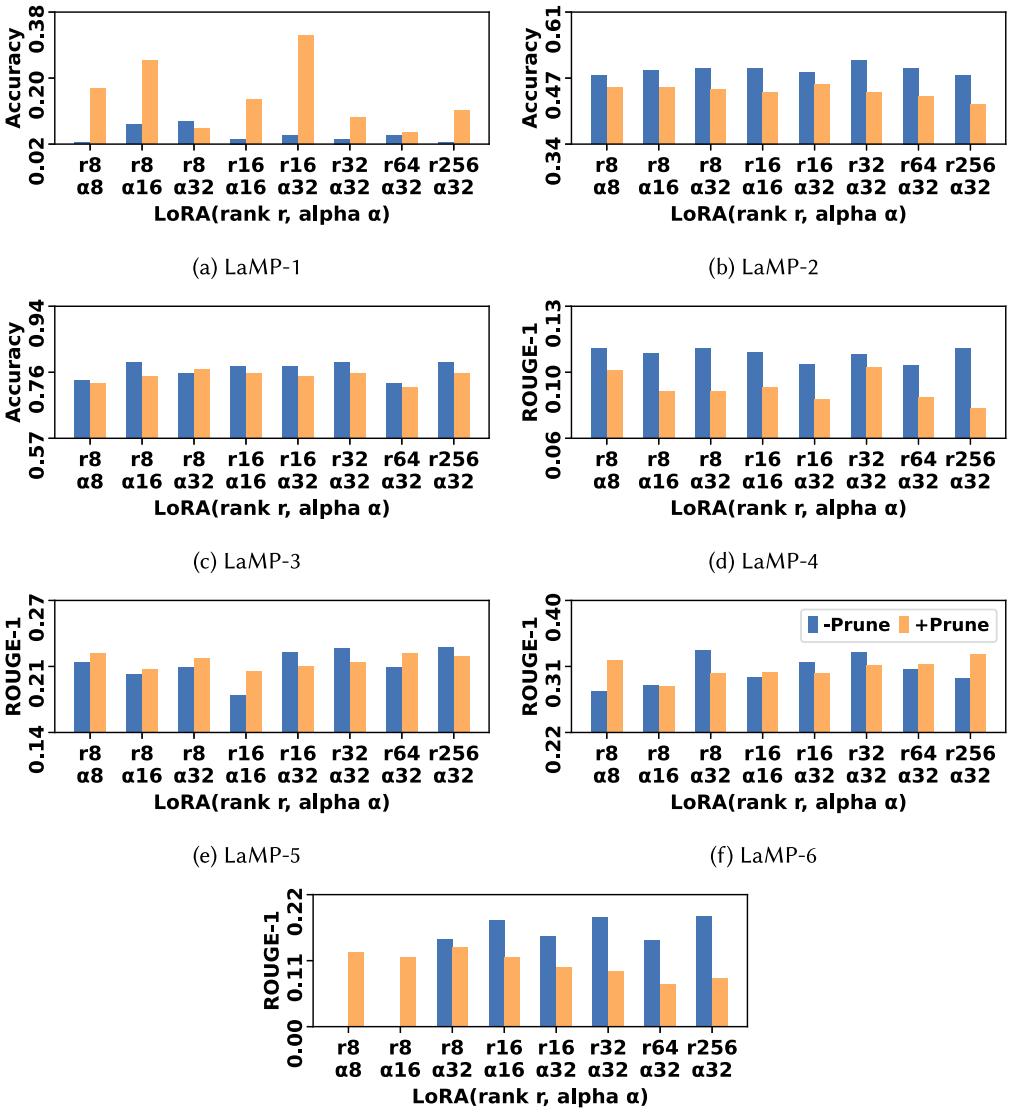
(d) LaMP-4

(e) LaMP-5

(f) LaMP-6

(g) LaMP-7

Fig. 42. Performance comparison between pruned and unpruned **Sheared-LLaMA-1.3B** under training hours from 1 to 8.



(a) LaMP-1

(b) LaMP-2

(c) LaMP-3

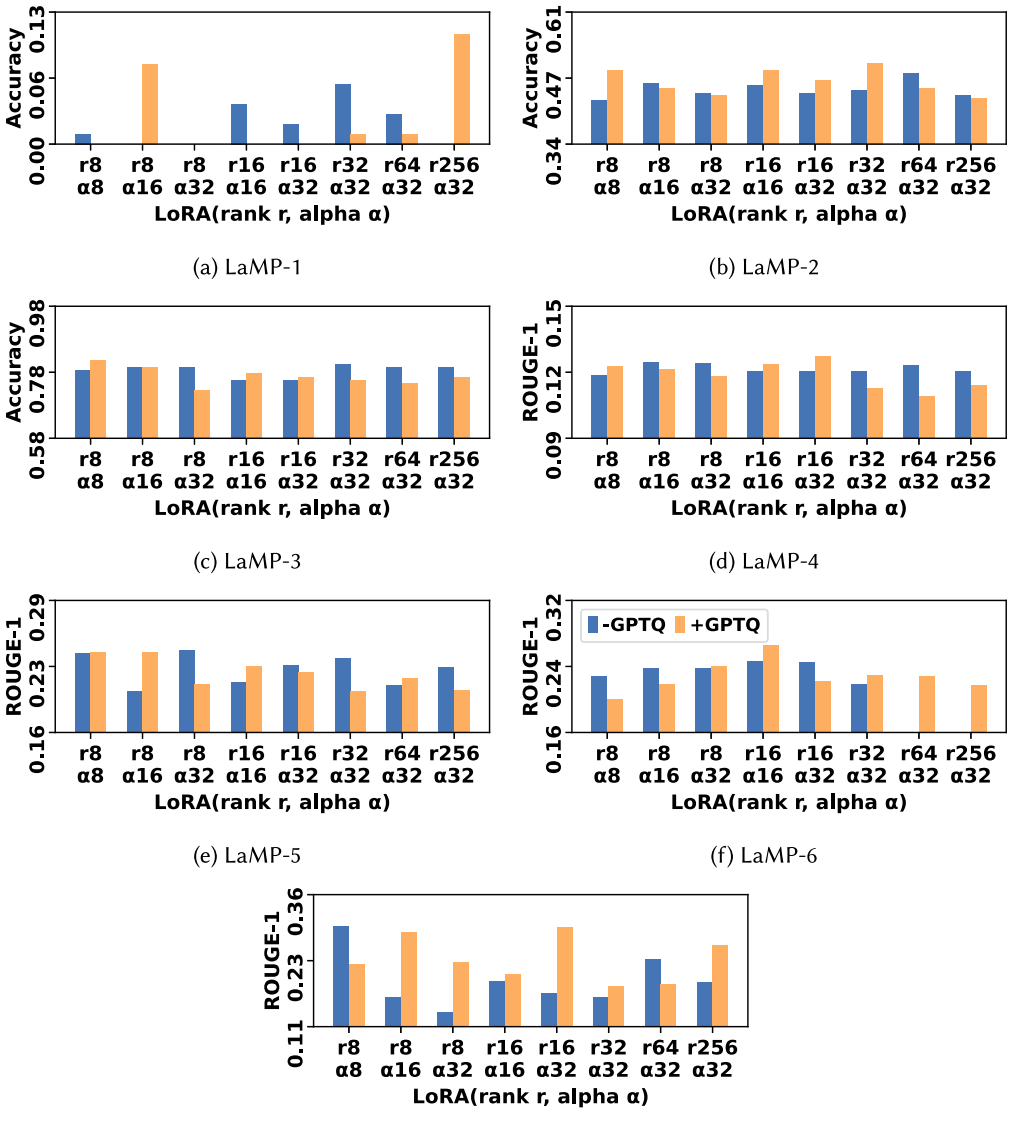
(d) LaMP-4

(e) LaMP-5

(f) LaMP-6

(g) LaMP-7

Fig. 43. Performance comparison between pruned and unpruned **Sheared-LLaMA-2.7B** under training hours from 1 to 8.



(a) LaMP-1

(b) LaMP-2

(c) LaMP-3

(d) LaMP-4

(e) LaMP-5

(f) LaMP-6

(g) LaMP-7

Fig. 44. Performance comparison between quantized and unquantized **Gemma-2b** under training hours from 1 to 8.

C Background and Related Works

C.1 Large Language Models Customization for Downstream Tasks

Different from BERT family language models [16, 50, 65, 73], LLMs, such as GPT-4 [1], Llama [79], and T5 [69], are pre-trained on vast amounts of text data [57]. This pretraining enables them to learn a wide range of language patterns, structures, and knowledge. However, to achieve optimal performance on specific downstream tasks such as classification, summarization, or translation, these models often require further customizations [30, 92]. Several approaches can be employed to tailor LLMs for specific downstream tasks, and we can generally group them into two categories: fine-tuning and prompt engineering. Fine-tuning uses targetted datasets to update the weights of LLMs so that the models can be familiar with the desired knowledge and structure of the outputs [92]. The dominant approach uses variations of **low-rank approximation** (**LoRA**) techniques [2, 15, 29, 76, 85], which freeze the original LLM weights and add additional low-rank decomposition weights that simulate weight updates. On the other hand, prompt engineering achieves downstream optimization without changing the LLM's weights. Typical approaches include few-shot in-context learning [13] and RAG [41]. In particular, RAG retrieves relevant documents from a knowledge base and adds the knowledge pertinent to the prompt. It grounds the generated answer with the retrieved knowledge and significantly increases response quality on commercial Cloud LLMs.

C.2 Large Language Model Evaluations

The field of LLMs has seen a significant amount of research focusing on various aspects of model evaluation [8, 24, 80]. Numerous studies have proposed benchmarks to systematically evaluate the capabilities of LLMs across different tasks. On the benchmark side, notable benchmarks include MMLU [26], GPQA [71], HumanEval [59], and GSM-8K [12], but they are not suitable for edge LLMs due to the reasons explained in the introduction section. In this article, we focus on the LaMP dataset [72], which closely aligns with the common use cases of edge LLMs. On the side of optimizations, research has explored the impact of quantization on LLM performance, aiming to make these models more efficient without significantly degrading their accuracy [42]. Optimal fine-tuning techniques have been extensively studied by works like Pu et al. [60] that provides a framework for choosing the optimal fine-tuning techniques given the task type and data availability, Liu et al. [52] that optimizes transformer architecture for edge devices, and Lin et al. [44] that focuses on the selection of LLM. However, to the best of our knowledge, those existing evaluation papers either focus on cloud LLMs or do not fully address the constraints outlined in the introduction, thus unable to provide a guidebook for edge LLMs.

Received 2 December 2024; revised 10 March 2025; accepted 17 May 2025