Empirical Guidelines for Deploying LLMs onto Resource-constrained Edge Devices

Ruiyang Qin[1*], **Dancheng Liu**[2*], **Zheyu Yan**[1], **Zhaoxuan Tan**[1],
Zixuan Pan[1], **Zhenge Jia**[1], **Meng Jiang**[1], **Ahmed Abbasi**[1], **Jinjun Xiong**[2], **Yiyu Shi**[1]
[1]University of Notre Dame, [2]University at Buffalo–SUNY
{rqin, zyan2, yshi4}@nd.edu, {dliu37, jinjun}@buffalo.edu

Abstract

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 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.

1 Introduction

The world has witnessed the growing interest in applying Large Language Models (LLMs) as a potential solution to personalized AI assistants [1]. This is particularly true when LLMs are deployed onto edge devices (edge LLMs) to meet the increasing needs of people's daily life assistance [2, 3], personalized companionship [4, 5], and real-time work assistance [6], where data privacy is of high priority [7] and constant internet connections may not be possible [8]. Some exemplar edge LLMs include *LLM on Nvidia IGX* [9], *Chat with RTX* [10], and *TinyChat* [11]. Edge LLMs can keep locally generated data from users and locally learn from those data. 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 [12].

The scaling law [13] has emerged as the standard for creating large language models (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 [14] because of the limited resources available on edge devices. It requires allocating limited resources to accommodate multiple needs as shown in Figure 1. An ideal resource allocation strategy needs to balance many key, yet sometimes conflicting, factors, such as the 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,

Preprint. Under review.

the compression methods of LLMs, the amount of time afforded to learn, and the difficulty levels of the target use 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 trade-offs between key design factors and their combined effects on learning efficiency and accuracy.

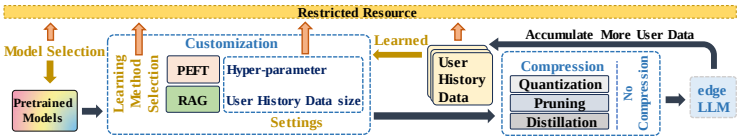


Figure 1: overview of the investigations on edge LLM

We formalize our empirical design guidelines for edge LLMs through extensive experimentation and benchmarking, 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.
- 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 trade-offs 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.

2 Preparation for Evaluation

2.1 Background: Running LLMs on Edge Devices

In this paper, we use a range of common edge devices to profile the performance of LLMs running on them, as summarized in Appendix A.1. Running an LLM on an edge device requires loading the model weights into random-access memory (RAM) of edge CPUs or edge GPUs, where model fine-tuning (or equivalently referred to as training in this paper) and inference can be performed. The RAM size on these devices commonly ranges from 4G to 16G, thereby limiting the size of model weights. Note that the RAM capacity constraint applies to both training and inference, and the former is usually the bottleneck as training consumes more memory than inference does. In addition, LLM parameter learning on edge devices is time-sensitive, and the training time is affected by both the edge device's hardware computation density (which depends on the hardware architecture as well as clock frequency), and the model complexity. To rule out the impact of pure hardware factor, the training times reported in this paper are all scaled as if each device were running at 1.5 TFLOPS (corresponding to Apple A14 GPU), and they range from one to eight hours. Note that in the same training time, different models are able to process different amounts of data, which is shown in detail in Table 5 in Appendix A.3.

2.2 Data

To investigate the customization of edge LLMs for each specific user, we concentrate on two task types: content classification and summarization. The edge LLMs should learn from the user history data and generate content to better adapt to the user's expectations. Hence, we use the datasets specifically designed to evaluate the LLM's personalization capabilities. Detailed reasons to use these datasets can be found in Appendix E.2. The detailed dataset structures and their prompt format can be found in Appendix A.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 3 and in Table 4 in Appendix A.2. 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 under multiple datasets (tasks), the difficulty of each task can be a latent factor impacting our evaluation [15]. Within the same dataset, we assume that utterances have similar levels of difficulty [16]. However, difficulties may vary across datasets, which is a non-negligible factor that requires attention. Zero-shot learning can be used to measure the difficulty of a provided dataset on a well-performing pre-trained model [17]. Therefore, we select GPT-4 [18], Claude 3 Opus [19], Gemini 1.0 Pro [20], and Llama 3 70B [21] to measure their performance on the testing data we choose from each dataset, and summarize the results in Table 6. For classification tasks, since each task has a different number of options, we consider a fairer assessment method similar to [22], where they normalize the accuracy with respect to human expert performance. However, since we could not get objective human performance on these datasets, we take an alternative approach by dividing the actual accuracy obtained from LLMs with the accuracy from random guessing, which we refer to as "normalized accuracy" in the table. Mathematically, the normalized accuracy is computed as $\frac{\text{accuracy}}{1/\text{No. choices}}$. For the summarization task, the normalized accuracy is the same as the accuracy. For the same type of task, the normalized accuracy can represent the difficulty of the task, where the higher the value, the easier the task is. Through our examination, the normalized accuracy for classification datasets are: LaMP-1 with 0.995, LaMP-2 with 5.157, and LaMP-3 with 3.236; the normalized accuracy for summarization datasets are: LaMP-4 with 0.1460, LaMP-5 with 0.3123, LaMP-6 with 0.3475, and LaMP-7 with 0.2385. The detailed results of each dataset in each cloud-based LLM can be found in Appendix A.6.

2.3 Learning Methods

There are two learning methods for edge LLM customization: parameter-efficient fine-tuning (PEFT) [23] and retrieval-augmented generation (RAG) [24]. While both in general can improve the cloud-based LLM performance [25], the performance can fluctuate given different settings in different datasets. It is important to investigate whether similar situations exist on edge LLMs.

PEFT: Compared to various PEFT implementations, low-rank adaption (LoRA) has demonstrated promising capabilities [26] in fine-tuning a wide range of LLMs and significantly improving their performance via updating a small portion of trainable parameters. The two hyperparameters *rank* and *alpha* can mostly impact the LoRA-tuning model performance [27]. Hence, we set all other hyperparameters to default, as described in Appendix A.4, and explore a wide range of rank and alpha value combinations and their corresponding trainable parameters in the performance of the edge LLM.

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 [28]. 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 [1]. In our experiments, we focus on the size of user history data and set all other hyperparameters to default, as shown in Appendix A.4.

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

2.4 Models

In addition to the data and learning method, the model itself can be undoubtedly important to the entire edge LLM framework. Under the constraints of edge devices, only the model with a certain size can be deployed. There are two types of models that can satisfy such constraints, either small-scale uncompressed models or models compressed from large-scale models. In our experiments, we examine both types of models.

Uncompressed models: Their performance can mostly related to their initial architectures and how they are trained. While they may not contain certain pre-trained knowledge due to the limitation of size, their performance will not risk being deteriorated by the compression methods. For such models, we mainly investigate Pythia [29], OPT [30], Llama [31], Gemma [32], and StableLM [33] series of models. To fit them into the corresponding edge device, we select each model with different weight sizes. Additionally, we profile these models under the edge environment settings to ensure that their peak resource usage will not exceed the resource limitation of the edge devices they are deployed on. The detailed model information along with their corresponding edge device information can be found in Appendix A.4, Table 5.

Compressed models: Other than their initial architectures and pre-training, the compression methods can also largely impact the model performance. There are three mainstream compression methods, including quantization [34], pruning [35], and knowledge distillation [36]. Model compression normally squeezes a large and pre-trained model into a smaller model. While each method can have certain advantages, the past works [37] mainly study such advantages for cloud-deployed LLMs, and it is not clear which one works best for compressing LLMs for edge devices. Among the existing compressed models, we select Llama, Mistral [38], Synthia [39], Gemma, and StableLM quantized by GPTQ [40] as the quantization models, Sheared-Llama [41] as the pruning models, and Phi [42] as the knowledge distillation models. Each series of the model contains various sizes, which allows us to deploy them on different edge devices and investigate their performance under a wide range of edge devices. The detailed model compression methods can be found in Appendix A.5

3 Results Analysis and Empirical Guidelines

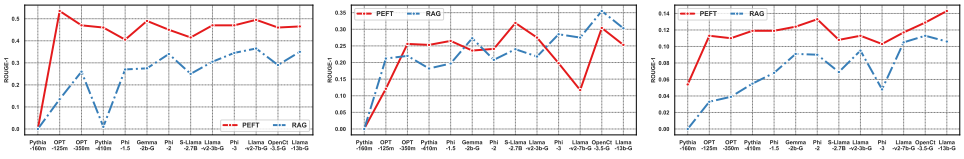
In this section, we provide experimental results analyses and explain how these analyses can lead us to formalize the empirical guidelines for edge LLMs. Due to the limitation of space, some of the detailed experimental results and all the discussion on hypotheses behind these observations are placed in Appendix B to D. 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, as shown in Table 6. While there is no existing literature directly comparing the difficulty across different types of tasks, we find that models often exceed human performance on classification tasks [43], whereas summarization tasks remain an open problem [44]. Thus, we argue that classification tasks are generally easier than summarization tasks. In the following 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

The first step before anything to deploy an LLM on the edge device is to select the model and then the customization. Thus, we start our guidelines with the most important question: which LLM models and customizations are best suited for edge devices?

To answer the above question, we conduct experiments on over 30 different LLM models and seven datasets as described in Section 2. Due to the space limit, we provide the full set of experiments in Appendix B.2.2, and we select some representative results, shown in Figure 2 and Table 1. Figure 2 includes 13 models of different sizes ranging from 160M to 13B and their performance on three tasks of increasing difficulty. Based on the ranking above, we choose LaMP-2 as the easy classification task, LaMP-6 as the easy summarization task, and LaMP-4 as the hard summarization task. In addition, we provide numerical results of 10 relatively big models on four tasks in Table 1, also in increasing difficulty. The models are Pythia-2.8B, OPT-2.7B, Llama-2-3B, StableLM-3B, Gemma-2B, Phi-2, Mistral-7B quantized, OpenChat quantized, Gemma-7B quantized, and Sheared Llama-2.7B pruned, respectively. Through experiments, we conclude that the choice of model and customization depends on the difficulty of the downstream task. We provide the general guidelines below:

- For simple classification tasks, the optimal choice should be a small LLM with PEFT.
- As the task difficulty increases such as complex classification tasks and simple summarization tasks, the choice should be gradually shifted to RAG with the strongest model. Here, the strongest models are (quantized) LLMs that excel at general benchmarks and fit within the RAM constraint. Currently, some of the promising candidates of strong models include OpenChat-3.5 and Llama3-8b [21]. Their quantized versions fit in an 8G RAM.
- As the difficulty of downstream tasks further increases to difficult summarization tasks, RAG will not be sufficient, and PEFT with strong and quantized models will stand out as the optimal choice.
- When the difficulty of the task could not be assessed, the Phi family combined with PEFT is a safe option that provides decent performance.



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

Figure 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.

Models		Py-2.8b	OPT-2.7b	Lla-2-3b	Sta-3b	Gem-2b	Phi-2	Mis-7b-G	OpenC-G	Gem-7b-G	S-Lla-2.7b-P
LaMP-2	PEFT	**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	0.320	0.345	0.290	0.300	0.035
LaMP-3	PEFT	**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**	0.696	0.667	0.765	0.627	0.755	**0.814**	0.480	0.578
LaMP-6	PEFT	**0.285**	**0.155**	**0.264**	**0.280**	0.229	**0.241**	0.296	0.304	**0.244**	**0.319**
	RAG	0.186	0.134	0.209	0.256	**0.299**	0.208	**0.327**	**0.355**	0.223	0.240
LaMP-7	PEFT	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**

Table 1: Performance comparisons between parameter learning and RAG, across ten relatively big models on four datasets of increasing difficulty. The complete table can be found in Appendix B.2.2.

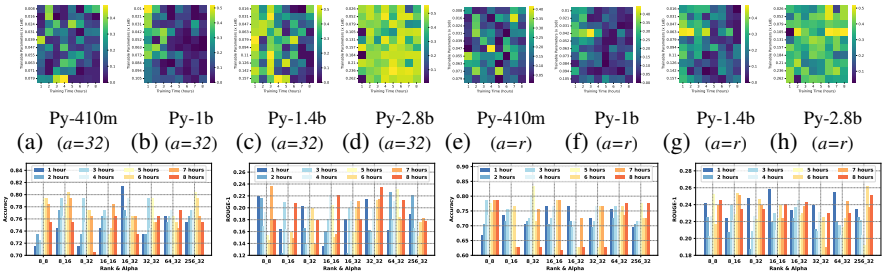
3.2 Choice of PEFT Strategies

As the usage of an edge LLM grows, it is expected to generate content that better satisfies user preferences. To achieve this, the model needs to learn from user history data. Typically, this is done through PEFT. When an LLM hosted on a cloud server is trained, the vast resources and large-scale training data available mean that different settings of PEFT can significantly impact the model's performance. Achieving better performance for such models often requires substantial human labor and effort to test various settings. However, for an edge LLM, resources and training data are often highly restricted. In this context, we aim to study three questions. **First**, is it necessary to try different PEFT settings to achieve better model performance given different models and data? **Second**, is there an optimal range or fixed PEFT setting that works for all edge LLM PEFT? Knowing the answers to these two questions can benefit future edge LLM research and usage by saving the time and resources needed to optimize edge LLM performance. **Furthermore**, we want to understand the factors that lead to different PEFT behavior in edge LLMs compared to LLMs hosted on cloud servers.

For the two main hyper-parameters *rank* and *alpha* in LoRA, different settings of them can lead to different trainable parameters. Over the model with different sizes and different architectures, we study 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. In Figure 3, we examine the performance of StableLM-2-1.6b and StableLM-3b on eight commonly used *rank* and *alpha* combinations, under the training time 1 to 8 hours. Our findings are as follows:

- Compared to varying the *alpha* value with *rank*, fixing *alpha* can benefit edge LLM PEFT more.
- There are no benefits of increasing training time even when the rank increases.
- As shown in Figure 3 and its supplemental results in Appendix B.1.4, even in larger StableLM models 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.



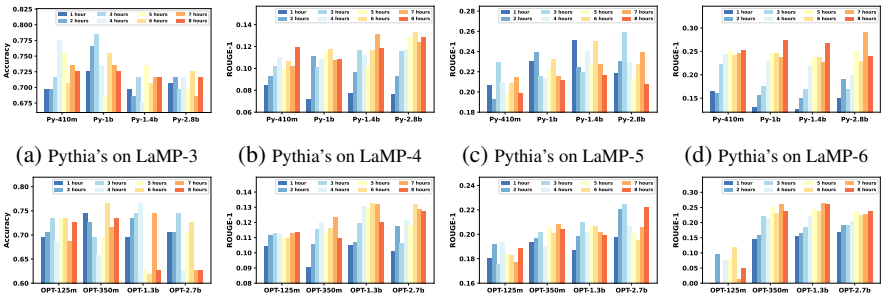
(a) (*a=32*) (b) (*a=32*) (c) (*a=32*) (d) (*a=32*) (e) (*a=r*) (f) (*a=r*) (g) (*a=r*) (h) (*a=r*)

(i) Sta-2-1.6b on LaMP-3 (j) Sta-2-1.6b on LaMP-5 (k) Sta-3b on LaMP-3 (l) Sta-3b on LaMP-5

Figure 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 (f), we set α = rank(r). More results are in Appendix B.1.2 and B.1.3. For (i)-(l): performance comparisons for two StableLM(Sta) models on LaMP-3 and LaMP-5 over eight combinations of *alpha* and *rank*. More results are in Appendix B.1.4.

3.3 Identification of PEFT Trends for Edge LLMs

During our investigation of PEFT settings, we provide the edge LLM with the largest training time to investigate its learning boundary. However, for the usage of edge devices, such a large training time can be less practical since training on them can fully occupy the devices for hours. For the edge LLM, if its one-hour learning can bring it similar performance lifting as eight-hour learning, then the one-hour learning is definitely more efficient. Hence, we further need to investigate whether the performance of edge LLM can consistently increase along with the longer training time with more training data.



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

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

Figure 4: Performance comparisons on multiple sized Pythia and OPT models on different amounts of training data. More performance comparisons can be found in Appendix B.1.1.

To conduct our investigations, we selected four Pythia models of different sizes and four OPT models of different sizes. Each model can be deployed on an edge device with varying RAM requirements, including 4G, 6G, 8G, and 16G. We used the PEFT settings obtained from previous experiments,

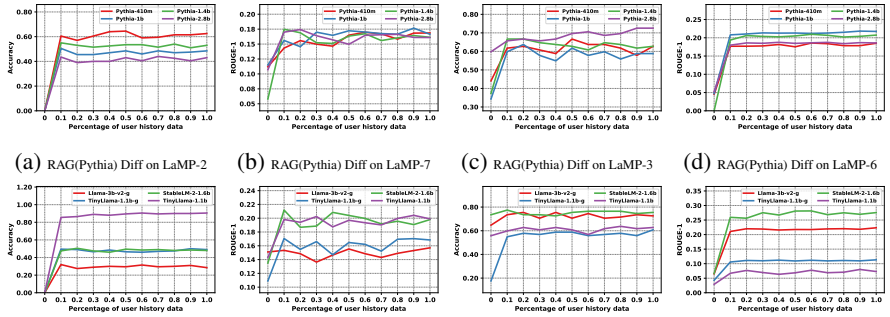
setting the *rank* to 8 and *alpha* to 32. Each model was given eight experiments to examine its performance across training time from 1 to 8 hours, allowing us to observe its performance when training from one hour up to eight hours. The experimental results are shown in Figure 4. Due to limited space, the performance on datasets LaMP-1, LaMP-2, and LaMP-7 is included in Appendix B.1.1. 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 towards downstream tasks on the A14 Bionic chip, and the time could be adjusted accordingly based on the edge device.
- While fine-tuning with PEFT 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 Figure 8 and Figure 9.

3.4 Impact of User History Data Volume on RAG Performance

Unlike PEFT, the resource-related settings for RAG can be simpler. RAG relies on max inner product search (MIPS) to find the appropriate information in user history data. To ensure the instant MIPS, all user history data will be converted into sentence embeddings and saved in RAM [1]. Compared to the computation cost of MIPS, the RAM usage of holding these embeddings might cost a higher resource burden, especially on the size of RAM [1]. Hence, we examine whether the higher volume of user history data necessarily leads to RAG performance improvement.

We select 100 users and each user has up to 1000 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 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 B.2.1. In general, we observe that increasing the amount of user history data to 1000 samples does not significantly enhance RAG performance compared to using only 100 samples. The RAG performance based on eight models is quite consistent across all different sizes of user history data. Furthermore, in part (c), more 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.



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

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

Figure 5: Performance improvement brought by RAG on four Pythia models and four miscellaneous(Misc) models across different sizes of user history data. More in Appendix B.2.1.

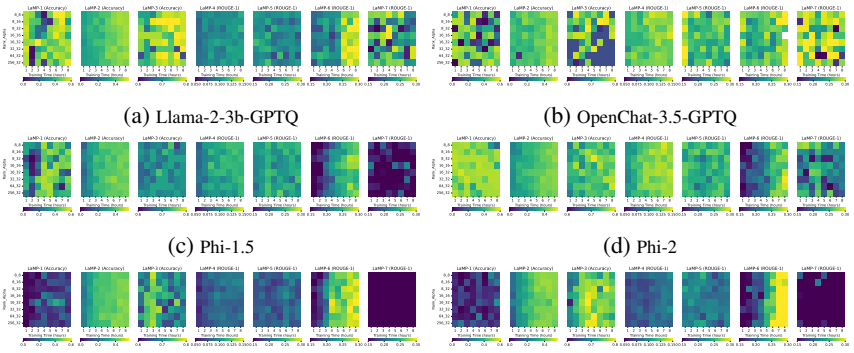
3.5 Comparison of Model Compression Techniques on Edge LLMs

In the previous sections, we have already explored the appropriate learning method and their settings for edge LLMs. Now, we investigate the other side of the problem: given an LLM, what is the best way to compress it? As such large-scale LLMs demonstrated promising reasoning abilities when they are deployed on the cloud servers, their compressed versions are also expected to demonstrate similar reasoning abilities when they are deployed on edge devices.

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 C.1 to C.4):

- Using distilled models such as Phi is a safe option when the type and difficulty of the downstream task cannot be determined. Phi-2 shows robustness towards both classification and summarization tasks, where it generally has a "brighter" performance heatmap shown in Figure 6, 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 to quantization. Thus, shearing is generally not preferred for edge LLMs where RAM capacity is a critical constraint.

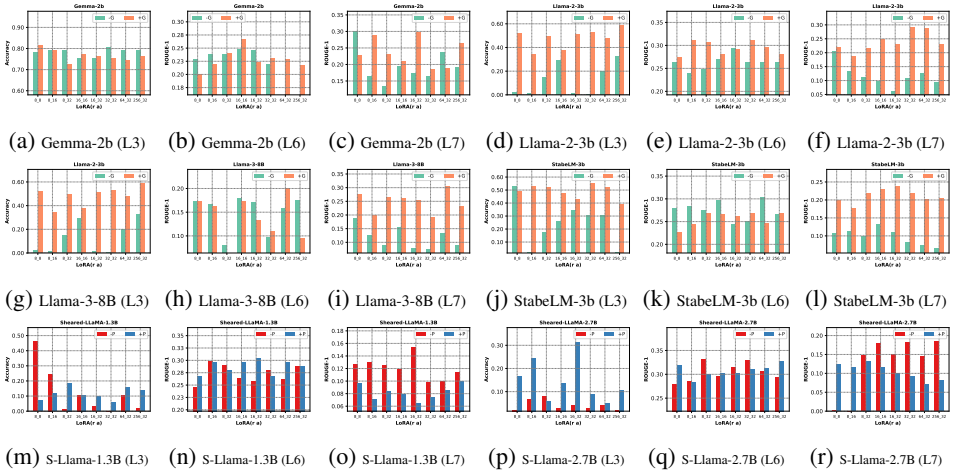


(a) Llama-2-3b-GPTQ (b) OpenChat-3.5-GPTQ

(c) Phi-1.5 (d) Phi-2

(e) Sheared-Llama-1.3b-Pruned (f) Sheared-Llama-2.7b-Pruned

Figure 6: Performance comparisons of two quantization models, two knowledge distillation models, and two pruning models over seven datasets across eight commonly used LoRA settings. More results can be found in Appendix C.4.



(a) Gemma-2b (L3) (b) Gemma-2b (L6) (c) Gemma-2b (L7) (d) Llama-2-3b (L3) (e) Llama-2-3b (L6) (f) Llama-2-3b (L7)

(g) Llama-3-8B (L3) (h) Llama-3-8B (L6) (i) Llama-3-8B (L7) (j) StabeLM-3b (L3) (k) StabeLM-3b (L6) (l) StabeLM-3b (L7)

(m) S-Llama-1.3B (L3) (n) S-Llama-1.3B (L6) (o) S-Llama-1.3B (L7) (p) S-Llama-2.7B (L3) (q) S-Llama-2.7B (L6) (r) S-Llama-2.7B (L7)

Figure 7: 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 C.2.

3.6 Comparison between Compressed and Uncompressed Edge LLMs

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. The results as shown in Figure 7. Through the investigations, our findings are as follows:

- While it is true that quantization will lead to some accuracy drops in most cases [22], it is noteworthy that with limited user history data for fine-tuning, the quantized model might perform better compared to unquantized counterparts on more challenging tasks.
- A larger model is not always better. Llama3-8B is not performing well on many of the benchmarks 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 7, bigger models work better with summarization tasks that require more semantic understanding of the context.

4 Limitations

In this work we have only considered the RAM capacity limit on edge devices; however, there are other factors that may be important for certain applications, such as the reliability of hardware, potential thermal-induced changes, etc, which may vary across different hardware. In addition, even though our results are the average from 100 users, due to the extensive amount of results that need to be presented, we are not able to include error bars to reflect the variances. Nonetheless, we categorize this work as an exploratory work in a field that is emergent, and future studies need to be done to provide more nuanced conclusions.

5 Conclusion and Future Directions

In conclusion, we present an empirical study for deploying large language models 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.

We hope that this work provides guidelines for future works and deployment of LLMs on the edge, and we also hope that this work will provide valuable directions that guide future research on LLMs under resource-constrained conditions. In addition to the findings, we offer some insights to the LLM community based on the experiment 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 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] Ruiyang Qin, Zheyu Yan, Dewen Zeng, Zhenge Jia, Dancheng Liu, Jianbo Liu, Zhi Zheng, Ningyuan Cao, Kai Ni, Jinjun Xiong, et al. Robust implementation of retrieval-augmented generation on edge-based computing-in-memory architectures. *arXiv preprint arXiv:2405.04700*, 2024.

[2] Marialena Bevilacqua, Kezia Oketch, Ruiyang Qin, Will Stamey, Xinyuan Zhang, Yi Gan, Kai Yang, and Ahmed Abbasi. When automated assessment meets automated content generation: Examining text quality in the era of gpts. *arXiv preprint arXiv:2309.14488*, 2023.

[3] Haozheng Luo, Ruiyang Qin, Chenwei Xu, Guo Ye, and Zening Luo. Open-ended multi-modal relational reasoning for video question answering. In *2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 363–369. IEEE, 2023.

[4] Irfan et al. Between reality and delusion: Challenges of applying large language models to companion robots for open-domain dialogues with older adults. 2023.

[5] Ruiyang Qin, Jun Xia, Zhenge Jia, Meng Jiang, Ahmed Abbasi, Peipei Zhou, Jingtong Hu, and Yiyu Shi. Enabling on-device large language model personalization with self-supervised data selection and synthesis. *arXiv preprint arXiv:2311.12275*, 2023.

[6] Brohan et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[7] Xu et al. Federated learning of gboard language models with differential privacy. *arXiv preprint arXiv:2305.18465*, 2023.

[8] Ji Lin, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, and Song Han. On-device training under 256kb memory. *Advances in Neural Information Processing Systems*, 35:22941–22954, 2022.

[9] NVIDIA. Deploy large language models at the edge with nvidia igx orin developer kit. `https://developer.nvidia.com/blog/deploy-large-language-models-at-the-edge-with-nvidia-igx-orin-developer-kit/`, 2024. Accessed: 2024-05-28.

[10] NVIDIA. Chat with rtx available now. `https://blogs.nvidia.com/blog/chat-with-rtx-available-now/`, 2024. Accessed: 2024-05-28.

[11] MIT Han Lab. Tinychat: Efficient and scalable chatbot models. `https://hanlab.mit.edu/blog/tinychat`, 2024. Accessed: 2024-05-28.

[12] Perttu Hämäläinen, Mikke Tavast, and Anton Kunnari. 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*, pages 1–19, 2023.

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

[14] Zheng Lin, Guanqiao Qu, Qiyuan Chen, Xianhao Chen, Zhe Chen, and Kaibin Huang. Pushing large language models to the 6g edge: Vision, challenges, and opportunities, 2024.

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

[16] Yuqiao Wen, Guoqing Luo, and Lili Mou. An empirical study on the overlapping problem of open-domain dialogue datasets. *arXiv preprint arXiv:2201.06219*, 2022.

[17] Adian Liusie, Potsawee Manakul, and Mark Gales. Llm comparative assessment: Zero-shot nlg evaluation through pairwise comparisons using large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 139–151, 2024.

[18] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[19] Darioush Kevian, Usman Syed, Xingang Guo, Aaron Havens, Geir Dullerud, Peter Seiler, Lianhui Qin, and Bin Hu. 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.

[20] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[21] Meta. Meta llama 3. `https://ai.meta.com/blog/meta-llama-3/`, 2023. Accessed: 2024-05-28.

[22] Lu Yin, Ajay Jaiswal, Shiwei Liu, Souvik Kundu, and Zhangyang Wang. Pruning small pre-trained weights irreversibly and monotonically impairs "difficult" downstream tasks in llms, 2024.

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

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

[25] Robert Lakatos, Peter Pollner, Andras Hajdu, and Tamas Joo. 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.

[26] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[27] Ariel N Lee, Cole J Hunter, and Nataniel Ruiz. Platypus: Quick, cheap, and powerful refinement of llms. *arXiv preprint arXiv:2308.07317*, 2023.

[28] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[29] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.

[30] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[31] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[32] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[33] Marco Bellagente, Jonathan Tow, Dakota Mahan, Duy Phung, Maksym Zhuravinskyi, Reshinth Adithyan, James Baicoianu, Ben Brooks, Nathan Cooper, Ashish Datta, et al. Stable lm 2 1.6 b technical report. *arXiv preprint arXiv:2402.17834*, 2024.

[34] Babak Rokh, Ali Azarpeyvand, and Alireza Khanteymoori. A comprehensive survey on model quantization for deep neural networks. *arXiv preprint arXiv:2205.07877*, 2022.

[35] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[36] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[37] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

[38] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[39] Migel Tissera. Synthia-70b-v1.2: Synthetic intelligent agent. `https://huggingface.co/migtissera/Synthia-13B`, 2023.

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

[41] Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, 2023.

[42] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023.

[43] Yeow Chong Goh, Xin Qing Cai, Walter Theseira, Giovanni Ko, and Khiam Aik Khor. Evaluating human versus machine learning performance in classifying research abstracts. *Scientometrics*, 125(2):1197–1212, Jul 2020.

[44] Divakar Yadav, Jalpa Desai, and Arun Kumar Yadav. Automatic text summarization methods: A comprehensive review, 2022.

[45] Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. Lamp: When large language models meet personalization. *arXiv preprint arXiv:2304.11406*, 2023.

[46] Migel Tissera. Synthia-70b: Synthetic intelligent agent, 2023.

[47] Guan Wang, Sijie Cheng, Qiying Yu, and Changling Liu. OpenLLMs: Less is More for Open-source Models, 7 2023.

[48] Hugging Face. Sentence transformers: all-minilm-l6-v2. `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`.

[49] Ruiyang Qin, Yuting Hu, Zheyu Yan, Jinjun Xiong, Ahmed Abbasi, and Yiyu Shi. Fl-nas: Towards fairness of nas for resource constrained devices via large language models. *arXiv preprint arXiv:2402.06696*, 2024.

[50] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *ArXiv*, abs/1705.08741, 2017.

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

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

[53] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

[54] Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023.

[55] Xuan Shen, Peiyan Dong, Lei Lu, Zhenglun Kong, Zhengang Li, Ming Lin, Chao Wu, and Yanzhi Wang. Agile-quant: Activation-guided quantization for faster inference of llms on the edge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18944–18951, 2024.

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

[57] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

[58] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

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

[60] Letian Peng, Zilong Wang, Feng Yao, Zihan Wang, and Jingbo Shang. Metaie: Distilling a meta model from llm for all kinds of information extraction tasks. *arXiv preprint arXiv:2404.00457*, 2024.

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

[62] 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. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022. Survey Certification.

[63] Haowei Lin, Baizhou Huang, Haotian Ye, Qinyu Chen, Zihao Wang, Sujian Li, Jianzhu Ma, Xiaojun Wan, James Zou, and Yitao Liang. Selecting large language model to fine-tune via rectified scaling law. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024.

[64] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[65] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[66] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[67] Zhenyu Pan, Haozheng Luo, Manling Li, and Han Liu. Conv-coa: Improving open-domain question answering in large language models via conversational chain-of-action. *arXiv preprint arXiv:2405.17822*, 2024.

[68] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. Instruction tuning for large language models: A survey, 2024.

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

[70] Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, XIAOPENG ZHANG, and Qi Tian. QA-loRA: Quantization-aware low-rank adaptation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[71] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[72] Sara Babakniya, Ahmed Roushdy Elkordy, Yahya H. Ezzeldin, Qingfeng Liu, Kee-Bong Song, Mostafa El-Khamy, and Salman Avestimehr. Slora: Federated parameter efficient fine-tuning of language models, 2023.

[73] Julian Coda-Forno, Marcel Binz, Zeynep Akata, Matthew Botvinick, Jane X Wang, and Eric Schulz. Meta-in-context learning in large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[74] Zishan Guo, Renren Jin, Chuang Liu, Yufei Huang, Dan Shi, Supryadi, Linhao Yu, Yan Liu, Jiaxuan Li, Bojian Xiong, and Deyi Xiong. Evaluating large language models: A comprehensive survey, 2023.

[75] 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. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.*, 15(3), mar 2024.

[76] Fangyun Wei, Xi Chen, and Lin Luo. Rethinking generative large language model evaluation for semantic comprehension, 2024.

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

[78] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

[79] Qiwei Peng, Yekun Chai, and Xuhong Li. Humaneval-xl: A multilingual code generation benchmark for cross-lingual natural language generalization. *arXiv preprint arXiv:2402.16694*, 2024.

[80] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[81] Shiyao Li, Xuefei Ning, Luning Wang, Tengxuan Liu, Xiangsheng Shi, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. Evaluating quantized large language models, 2024.

[82] George Pu, Anirudh Jain, Jihan Yin, and Russell Kaplan. Empirical analysis of the strengths and weaknesses of peft techniques for llms. *arXiv preprint arXiv:2304.14999*, 2023.

Appendix

A Supplements to Evaluations and Experiments

A.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. 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.

Edge Devices for Experiments. In this paper we selected some common edge devices with different RAM capacities and computation densities for evaluation as shown in Table 2. RAM is a critical resource for enabling Edge LLM learning. The model weights must be loaded into RAM so that they can be updated and optimized to adapt to user needs. In other words, when choosing to fine-tune an Edge LLM, we should consider not only the size of its weights but also the additional RAM usage due to parameter optimization and gradient calculation. Conversely, when opting to use RAG to facilitate Edge LLM learning, the consideration shifts to only the weights size and data embedding size, as RAG requires storing all data embeddings in RAM to perform MIPS. To eliminate the impact of pure hardware factors, we scale all the profiled training times as if a device were running at 1.5 TFLOPS (corresponding to Apple A14 GPU), i.e., the training time of a model is device-agnostic.

Device	RAM (GB)	FLOPS (T)
iPhone 12 (A14)	4	1.50
iPhone 15 (A16)	6	3.58
NVIDIA Jetson Orin Nano	8	0.50
Samsung Galaxy S24 Ultra	10	3.5
NVIDIA Jetson Orin NX	16	3.76

Table 2: Profile of five common edge devices evaluated in this paper

A.2 Datasets

Background. 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.

Personalization Dataset. We have selected the LaMP datasets [45] to evaluate our Edge LLM. In LaMP, each user has history data as shown in Table 3. 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 4. 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.

A.3 Selected LLMs and Profiling

In the previous section, we have explained the edge device constraints. In Table 2, 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 [29] and Llama [31]. To satisfy

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"

Table 3: One sample of user history data in each dataset

6G RAM, we pick four models from the family of StableLM [33], Pythia [29], and Gemma [32]. To satisfy 8G RAM, we pick six models from the family of Phi [42], Pythia, Llama [31], Mistral [38], Synthia [46], and OpenChat [47].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 5. For each model we make profiling to find its peak RAM usage, showing as "Training Peak RAM" on Table 5. [1] The model with smaller training peak RAM can also run on larger RAMs.

[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 5 but we still use the model for some of the experiments.

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."

Table 4: One sample of query data in each dataset

A.4 Experimental Settings

For each LaMP dataset, we analyze 100 users, each containing up to 1000 documents in their user history data. Each document encapsulates a single piece of user information, as illustrated in Table 3. 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 max inner product search (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* [48]. We select the highest-ranked data as the output in MIPS, setting the top_k parameter to 3.

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

Table 5: Selected models with their initial model weights size and the training peak RAM. For the data size per hour, it indicates the estimated amount of data each model can learn or train from given one hour.

Parameter-Efficient Fine-Tuning (PEFT). We choose LoRA [26] as the PEFT implementation in our work. For LoRA, we enable it to fine-tune parameters for *query*, *key*, and *value* layers. We set the dropout rate for LoRA as *0.1*. The initialized learning rate is *5e-4*, along with a linear learning rate scheduler to optimize the learning rate. We use *adamW* as the optimizer in LoRA. Additionally, we set the fine-tuning task type as *CAUSAL_LM*. For the rank and alpha, since they directly relate to the size of trainable parameters, we set wide ranges. For rank, it can increment from *8* to *256*. For alpha, it can increment from *8* to *32*. Such ranks allow us to explore different trainable parameter sizes.

A.5 Model Compression

The LLMs were originally designed and trained for cluster computing [49]. 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 [50]. 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.

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) [51]. Several implementations of PTQ for LLMs exist, including SmoothQuant [52], AWQ [53], LLM-QAT [54], and GPTQ [40].

For experimental consistency, in our experiments, we selected the widely-used GPTQ, which has been applied to almost all LLMs and has proven effective. Furthermore, GPTQ has been specifically adapted for Edge LLMs [55, 56].

Pruning Different from quantization, pruning lightens the model by trimming off certain weights without lowering the model performance. Compared to 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 [57], LLM-Pruner [58], and Sheared-LLaMA [41], 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 [42]. This approach represents a novel departure from traditional large language models (LLMs), allowing SLMs to acquire dense knowledge from LLMs while maintaining a smaller size. Although other works such as Distilling Step-by-Step [59], MetaIE [60], and MLFS [61] 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.

A.6 Task Difficulty

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

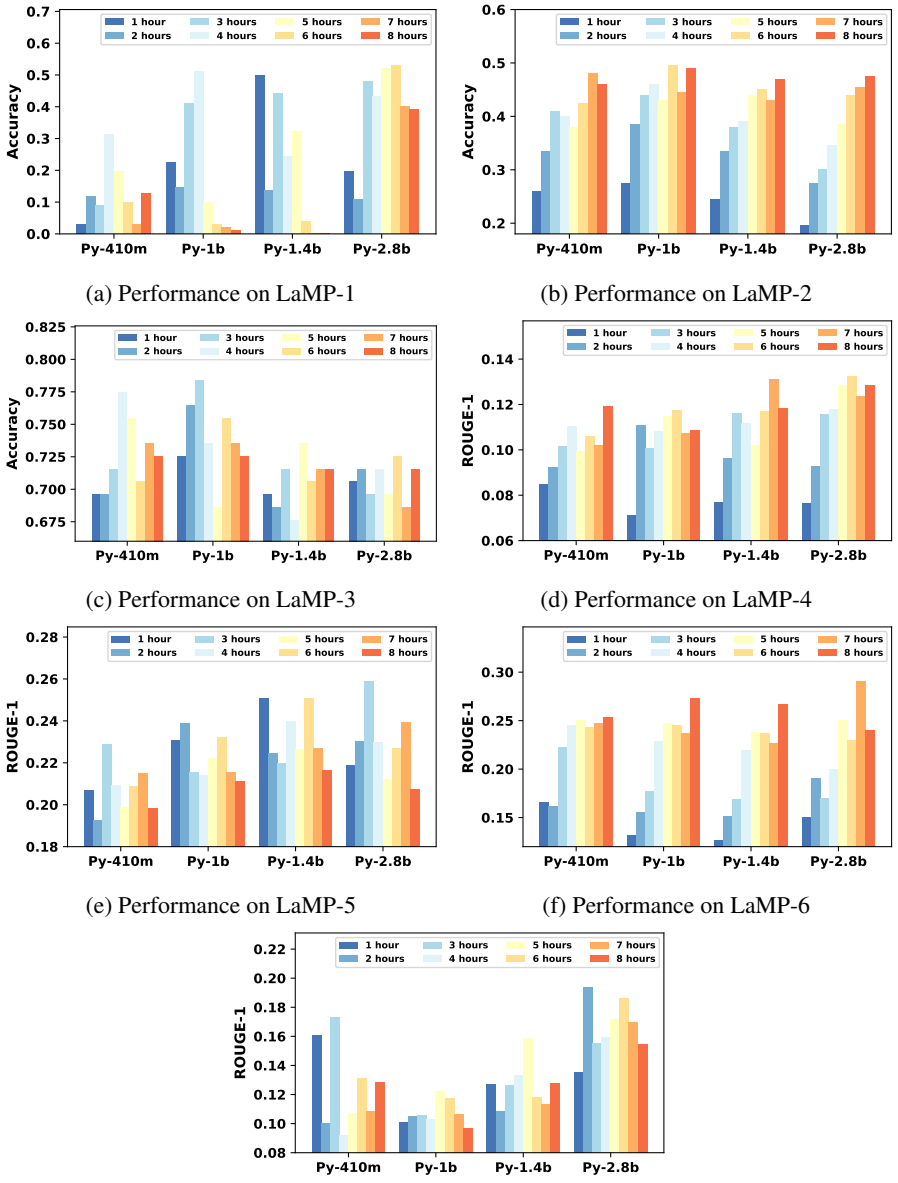
Table 6: Difficulty (weighted average) comparisons of different datasets. The mean of four cloud-based LLMs' zero-shot performance can be used as an evaluator of task difficulty.

B Results of Edge LLM Learning

B.1 Experimental Results for PEFT

B.1.1 Experiments: model size and training time

The following results shown in Figure 8 and Figure 9 correspond to the question "*Under the constraint of edge devices, does longer training time always result in better learning outcomes?*" in experiment analysis section 3.3.



(a) Performance on LaMP-1

(b) Performance on LaMP-2

(c) Performance on LaMP-3

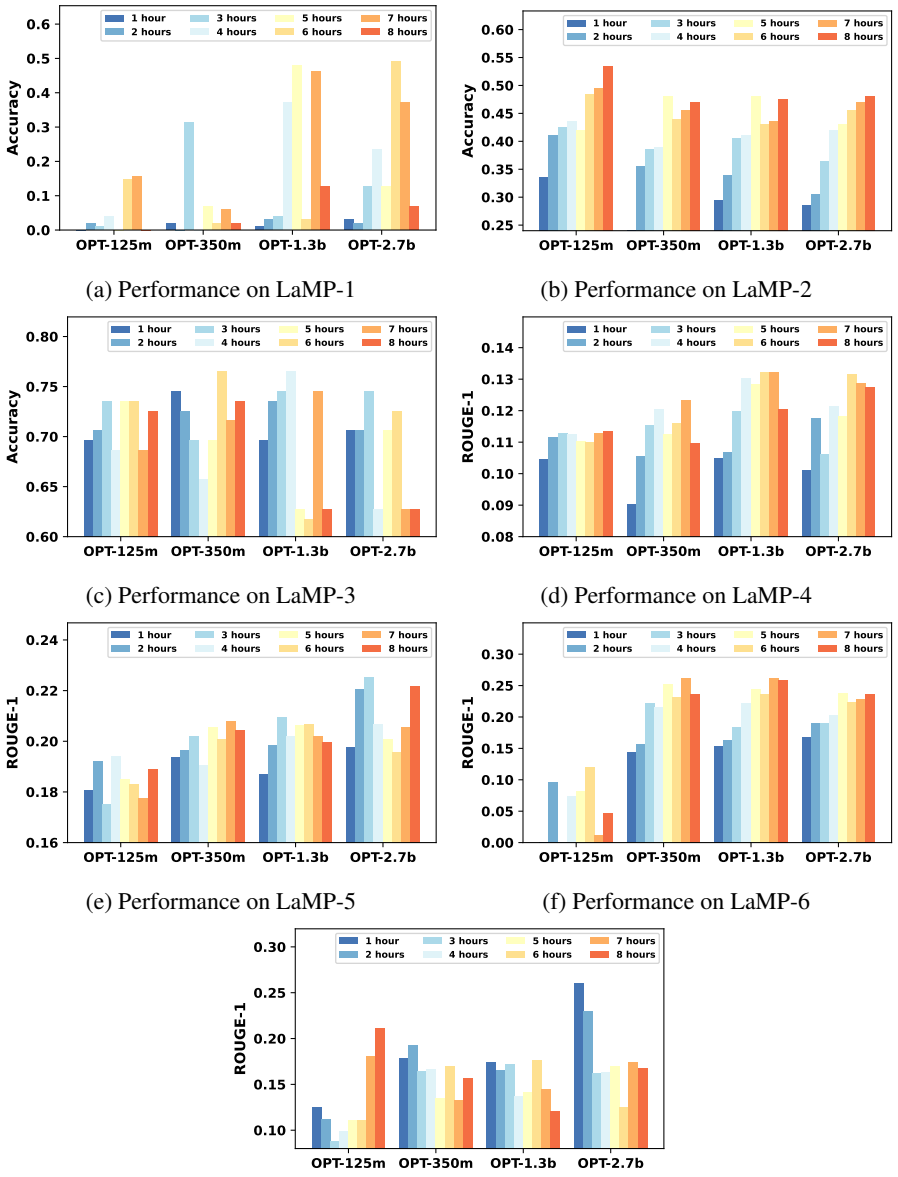
(d) Performance on LaMP-4

(e) Performance on LaMP-5

(f) Performance on LaMP-6

(g) Performance on LaMP-7

Figure 8: 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 5 Experiments are based on the default 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

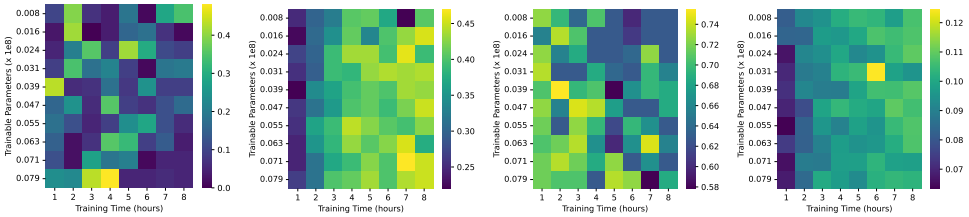
(f) Performance on LaMP-6

(g) Performance on LaMP-7

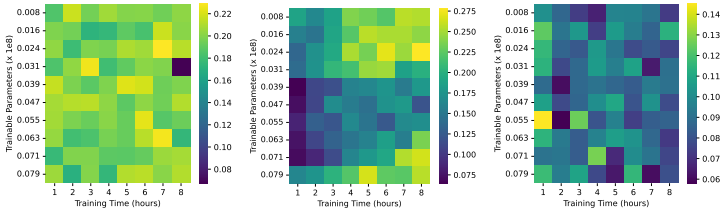
Figure 9: Performance comparison of selected LLMs on different RAM sizes, including OPT-125m on RAM 4G, OPT-350m on RAM 6G, OPT-1.3b on RAM 8G, and OPT-2.7b on RAM 16G. Examine their performance across datasets with different difficulty levels. Legend 1 to 8 represent the number of hours of training, where the detailed data size per unit hour can be found in Table 5 Experiments are based on the default settings with *rank* = 8 and *alpha* = 8.

B.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 the question "*Does larger portion of trainable parameters conduct better performance?*" in experiment analysis section 3.2.

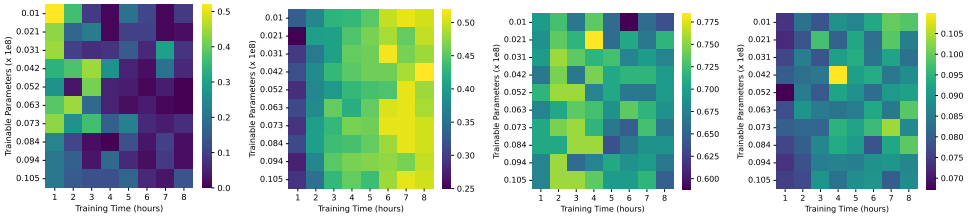


(a) LaMP-1 (b) LaMP-2 (c) LaMP-3 (d) LaMP-4

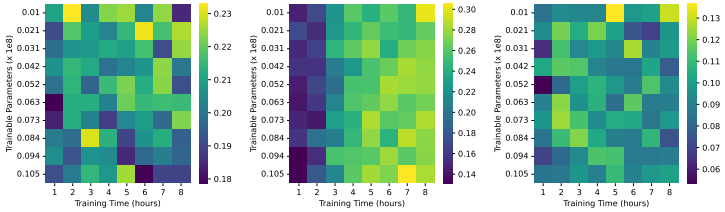


(e) LaMP-5 (f) LaMP-6 (g) LaMP-7

Figure 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.

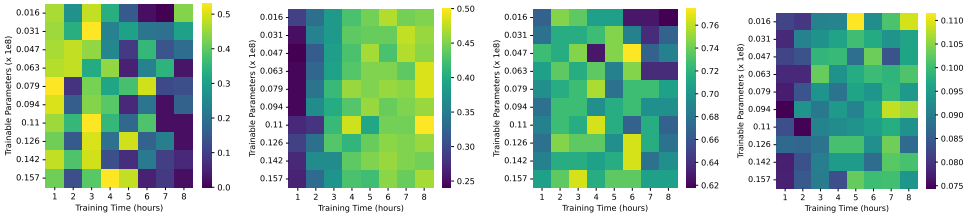


(a) LaMP-1 (b) LaMP-2 (c) LaMP-3 (d) LaMP-4

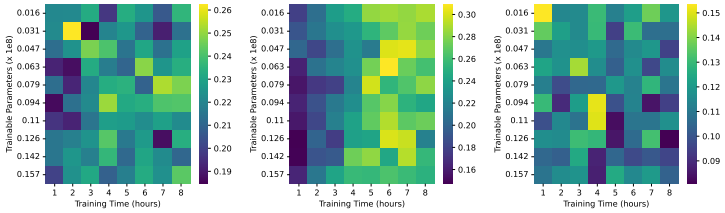


(e) LaMP-5 (f) LaMP-6 (g) LaMP-7

Figure 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.

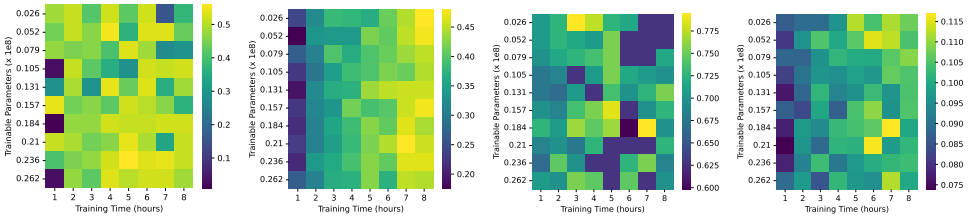


(a) LaMP-1 (b) LaMP-2 (c) LaMP-3 (d) LaMP-4

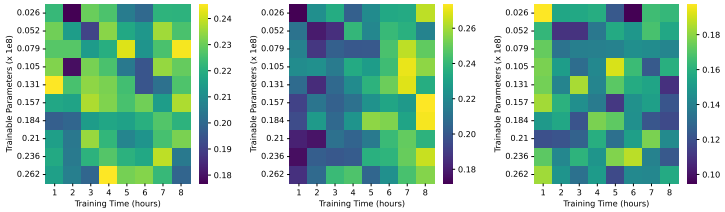


(e) LaMP-5 (f) LaMP-6 (g) LaMP-7

Figure 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

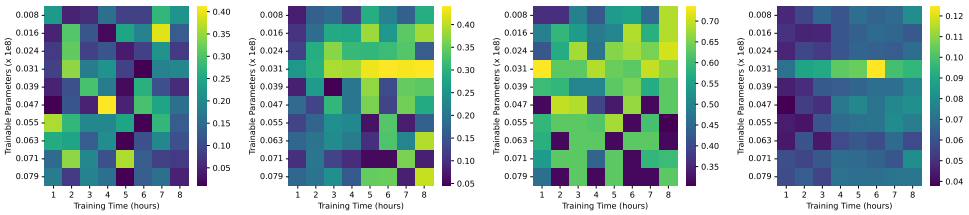


(e) LaMP-5 (f) LaMP-6 (g) LaMP-7

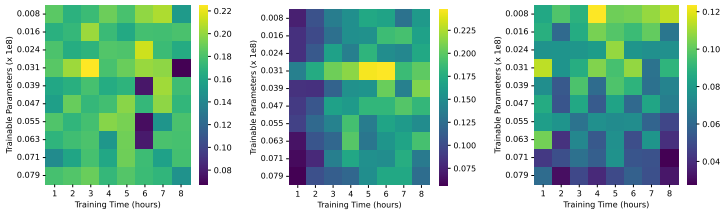
Figure 13: Performance heatmaps for Pythia-2.8b on dataset LaMP-1 to LaMP-7, given *alpha* = 32 and *rank* = 8, 16, 24, 32, 40, 48, 56, 72, 80. In each heatmap, we use the number of trainable parameters of different *rank* values as y-axis , and the training hours as the x-axis.

B.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 the question "*Does larger portion of trainable parameters conduct better performance?*" in experiment analysis section 3.2.

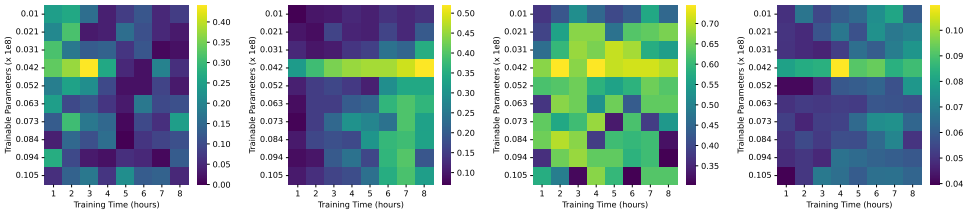


(a) LaMP-1 (b) LaMP-2 (c) LaMP-3 (d) LaMP-4

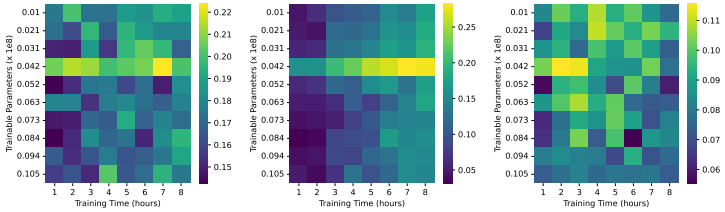


(e) LaMP-5 (f) LaMP-6 (g) LaMP-7

Figure 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.

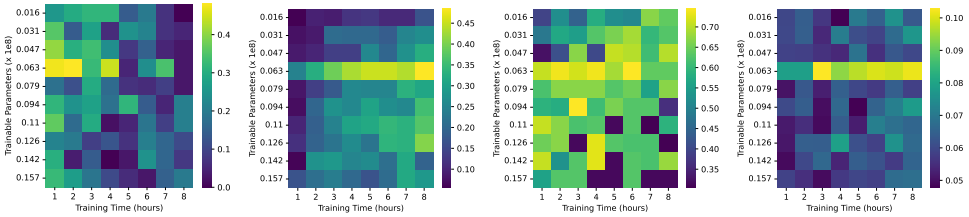


(a) LaMP-1 (b) LaMP-2 (c) LaMP-3 (d) LaMP-4

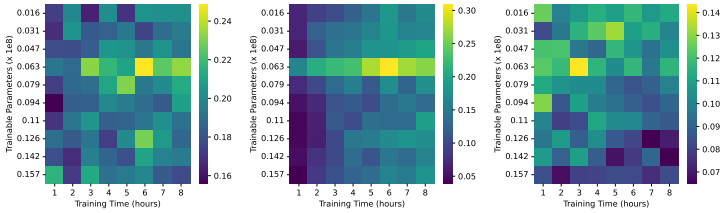


(e) LaMP-5 (f) LaMP-6 (g) LaMP-7

Figure 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.

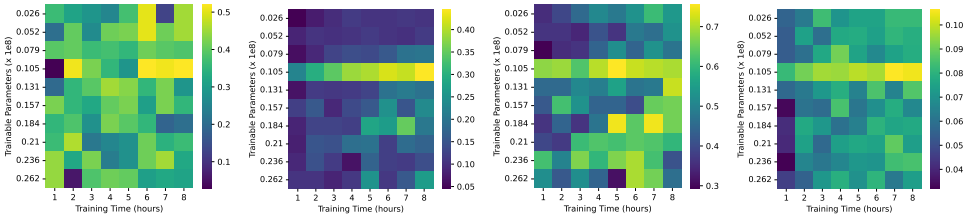


(a) LaMP-1 (b) LaMP-2 (c) LaMP-3 (d) LaMP-4

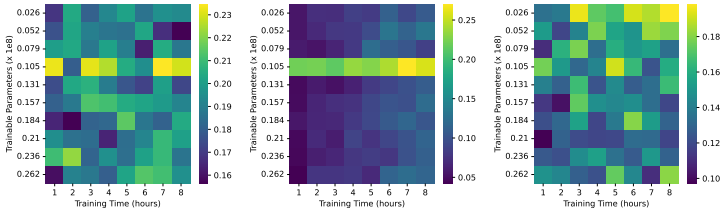


(e) LaMP-5 (f) LaMP-6 (g) LaMP-7

Figure 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

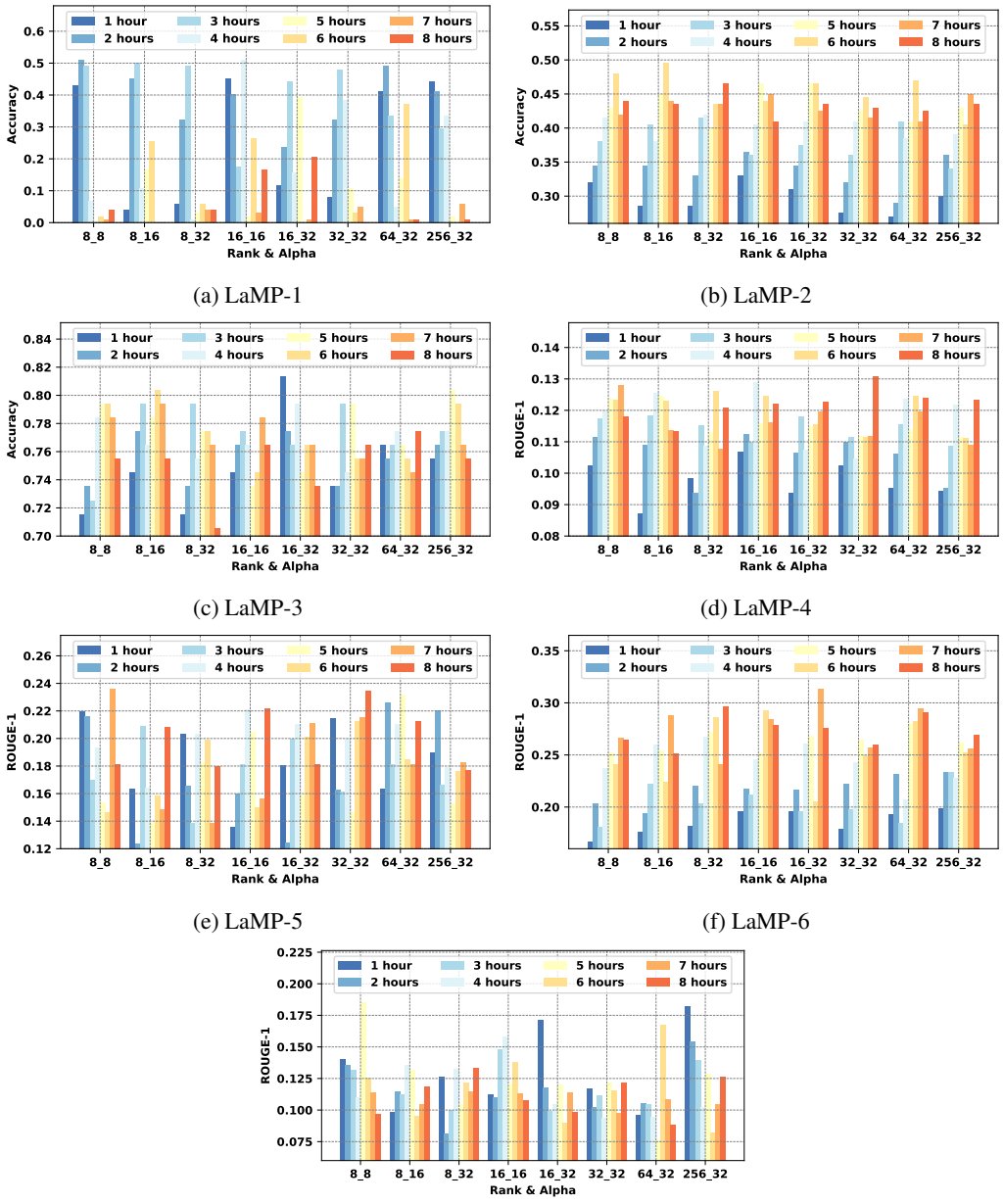


(e) LaMP-5 (f) LaMP-6 (g) LaMP-7

Figure 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.

B.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 the question "*Does larger portion of trainable parameters conduct better performance?*" in experiment analysis section 3.2.



(a) LaMP-1

(b) LaMP-2

(c) LaMP-3

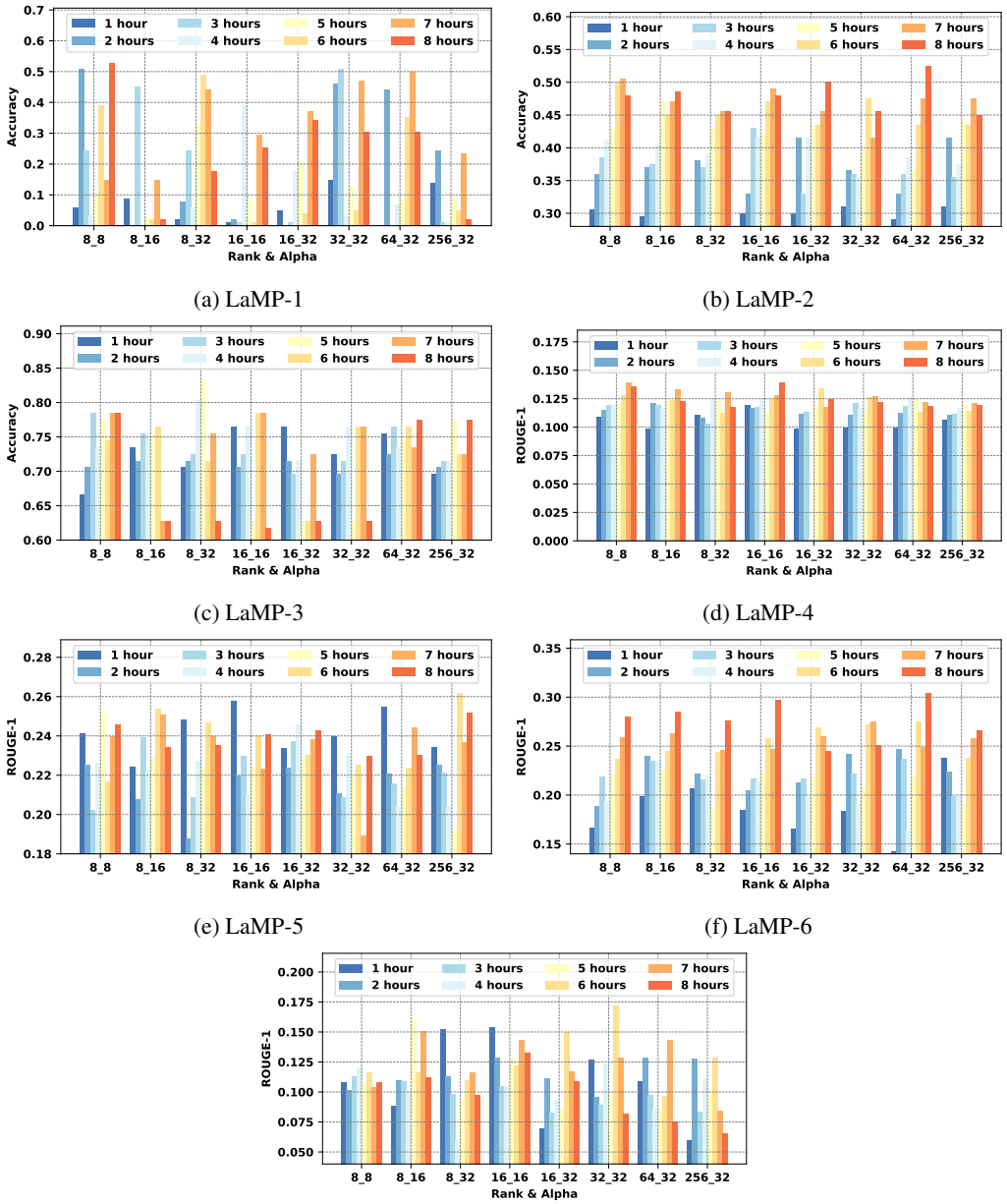
(d) LaMP-4

(e) LaMP-5

(f) LaMP-6

(g) LaMP-7

Figure 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 settings.



(a) LaMP-1

(b) LaMP-2

(c) LaMP-3

(d) LaMP-4

(e) LaMP-5

(f) LaMP-6

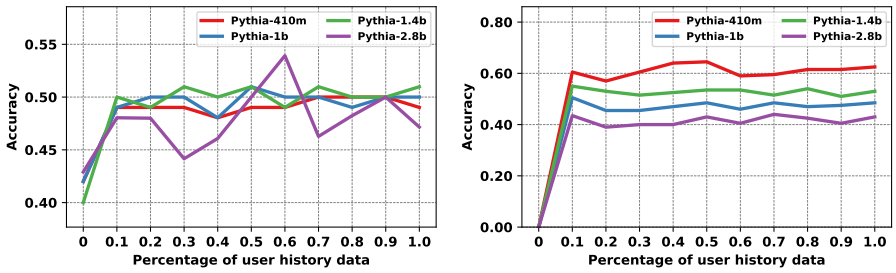
(g) LaMP-7

Figure 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 settings.

B.2 Experimental Results for RAG

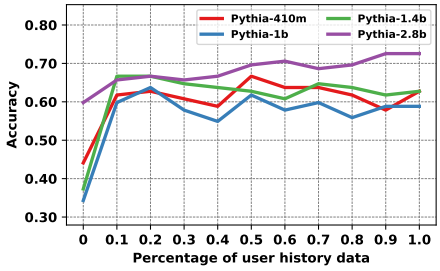
B.2.1 Experiments: RAG performance on different sizes of user history data

Below, we present the experiments for the impact of the amount of user history data on RAG performance. These experiments can correspond to the question "What is the threshold of the user history data volume to ensure a good RAG performance?" in the section 3.4

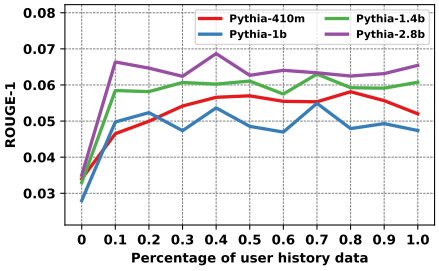


(a) LaMP-1

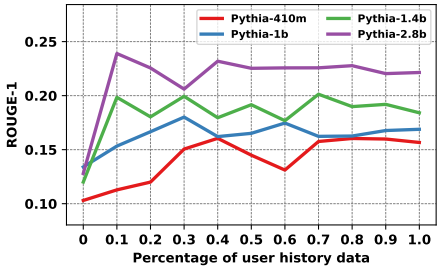
(b) LaMP-2



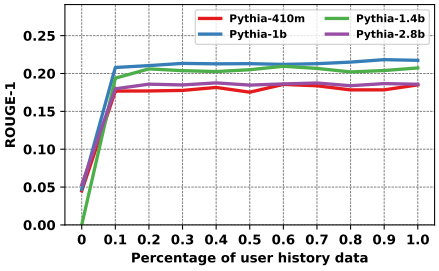
(c) LaMP-3



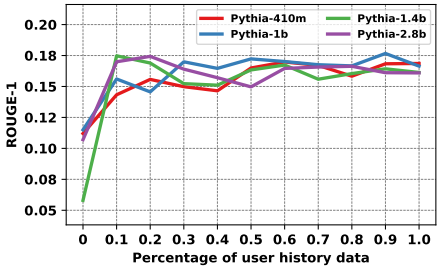
(d) LaMP-4



(e) LaMP-5

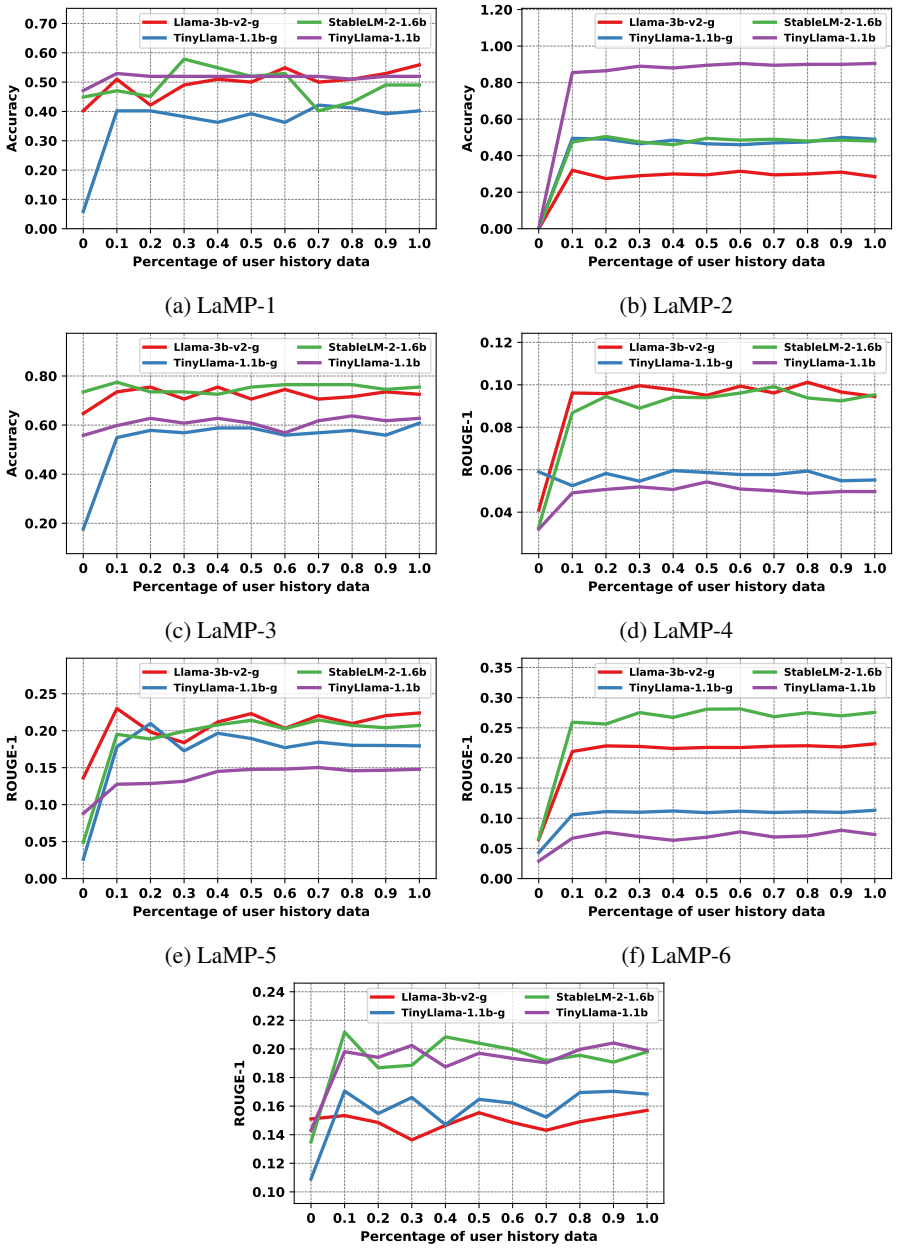


(f) LaMP-6



(g) LaMP-7

Figure 20: Performance improvement brought by RAG on four Pythia models of different sizes across different sizes of user history data on all seven datasets.



(a) LaMP-1

(b) LaMP-2

(c) LaMP-3

(d) LaMP-4

(e) LaMP-5

(f) LaMP-6

(g) LaMP-7

Figure 21: 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-3b quantized, StableLM-2-1.6B, TinyLlama-1.1B quantized, and TinyLlama-1.1B.

B.2.2 Experiments: Examine RAG and PEFT

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	0.380	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	0.000	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	0.250
Llama-v3-8b	0.324	0.000	0.130	0.000	0.627	0.000	0.065	0.000	0.180	0.000	0.172	0.000	0.189	0.000
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	0.000	0.105	0.000	0.278	0.000	0.275	0.249	0.157
Llama-v3-8b-G	0.539	0.000	0.140	0.000	0.461	0.000	0.073	0.000	0.229	0.000	0.174	0.000	0.277	0.000
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

Table 7: Performance comparisons of PEFT and RAG for selected LLMs. 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 8.

[1] G represents GPTQ, the quantization technique

[2] P represents Pruned

[3] S represents Sheared

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	0.000	0.441	0.034	0.103	0.045	0.112
Pythia-1b	0.420	0.000	0.343	0.028	0.134	0.047	0.115
Pythia-1.4b	0.400	0.000	0.373	0.033	0.120	0.000	0.058
Pythia-2.8b	0.429	0.000	0.598	0.035	0.128	0.053	0.107
opt-125m	0.039	0.000	0.304	0.018	0.098	0.034	0.096
opt-350m	0.429	0.000	0.255	0.021	0.106	0.056	0.116
opt-1.3b	0.429	0.000	0.284	0.033	0.130	0.041	0.102
opt-2.7b	0.429	0.000	0.451	0.025	0.113	0.051	0.152
TinyLlama-1.1b	0.471	0.000	0.588	0.032	0.088	0.029	0.143
Llama-v2-3b	0.469	0.000	0.461	0.026	0.136	0.053	0.150
Llama-v3-8b	0.000	0.000	0.000	0.000	0.000	0.000	0.000
StableLM-2-1.6b	0.449	0.000	0.735	0.033	0.138	0.066	0.135
StableLM-3b	0.429	0.000	0.441	0.040	0.177	0.067	0.257
Gemma-2b	0.078	0.000	0.696	0.025	0.145	0.077	0.130
Phi-1.5	0.500	0.000	0.382	0.023	0.094	0.056	0.062
Phi-2	0.429	0.010	0.471	0.064	0.096	0.059	0.259
Phi-3	0.441	0.000	0.549	0.049	0.108	0.104	0.098
Llama-v2-3b-G[1]	0.402	0.000	0.647	0.041	0.173	0.065	0.158
StableLM-3b-G	0.490	0.005	0.559	0.049	0.225	0.098	0.166
TinyLlama-1.1B-G	0.059	0.000	0.176	0.026	0.069	0.043	0.109
Gemma-2b-G	0.402	0.000	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	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Mistral-7b-G	0.429	0.000	0.343	0.034	0.210	0.079	0.166
OpenChat-3.5-G	0.480	0.000	0.735	0.052	0.182	0.134	0.202
Synthia-7b-G	0.500	0.000	0.373	0.033	0.120	0.000	0.271
Gemma-7b-G	0.402	0.000	0.676	0.026	0.114	0.051	0.112
Llama-13b-G	0.429	0.000	0.696	0.027	0.252	0.068	0.174
S^3-Llama-1.3b-P^2	0.294	0.000	0.255	0.028	0.111	0.038	0.176
S-Llama-1.3B	0.429	0.000	0.206	0.030	0.107	0.029	0.122
S-Llama-2.7B-P	0.010	0.000	0.422	0.027	0.128	0.043	0.124
S-Llama-2.7B	0.429	0.000	0.549	0.029	0.174	0.074	0.118

Table 8: Performance comparisons of zero-shot learning for all selected LLMs.

C Results of Edge LLM Design

C.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 9 and Table 10. Then, we examine the quantized Llama family models, showing results in Figure 22. Additionally, we examine the quantized models with similar size but different architectures, showing results in Figure 23.

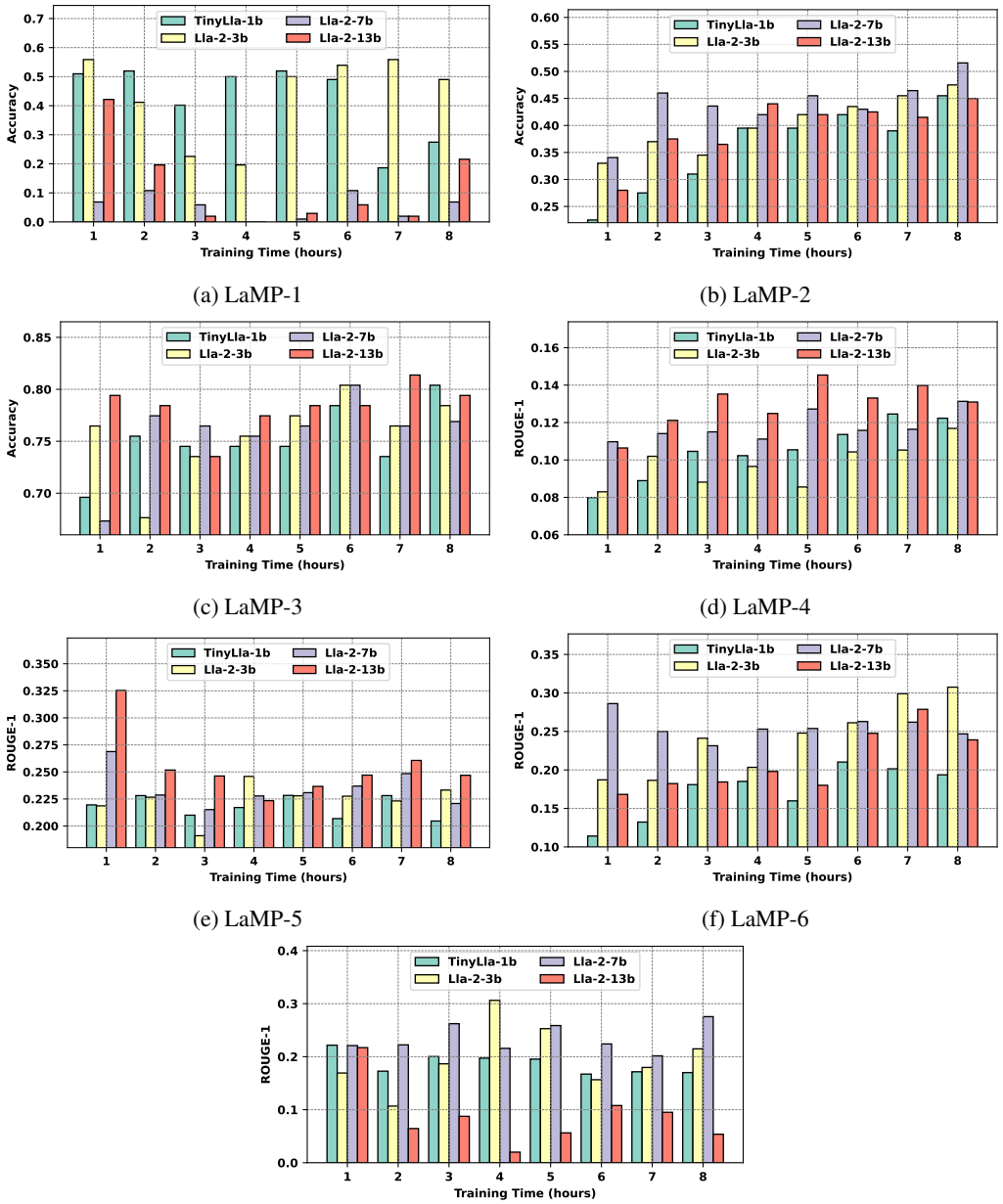
Gemma-GPTQ	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-6		LaMP-7	
	2b	7b	2b	7b	2b	7b	2b	7b	2b	7b	2b	7b	2b	7b
R(8) A(8)	0.520	0.529	0.490	0.420	0.814	0.775	0.124	0.105	0.240	0.213	0.201	0.244	0.227	0.030
R(8) A(16)	0.408	0.451	0.455	0.440	0.794	0.784	0.122	0.099	0.240	0.184	0.220	0.220	0.289	0.032
R(8) 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 9: 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
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

Table 10: Performance comparisons between two quantized models with different normalized data sizes under the LoRA setting: *rank* = 8 and *alpha* = 32.

[1] G represents GPTQ, the quantization technique



(a) LaMP-1

(b) LaMP-2

(c) LaMP-3

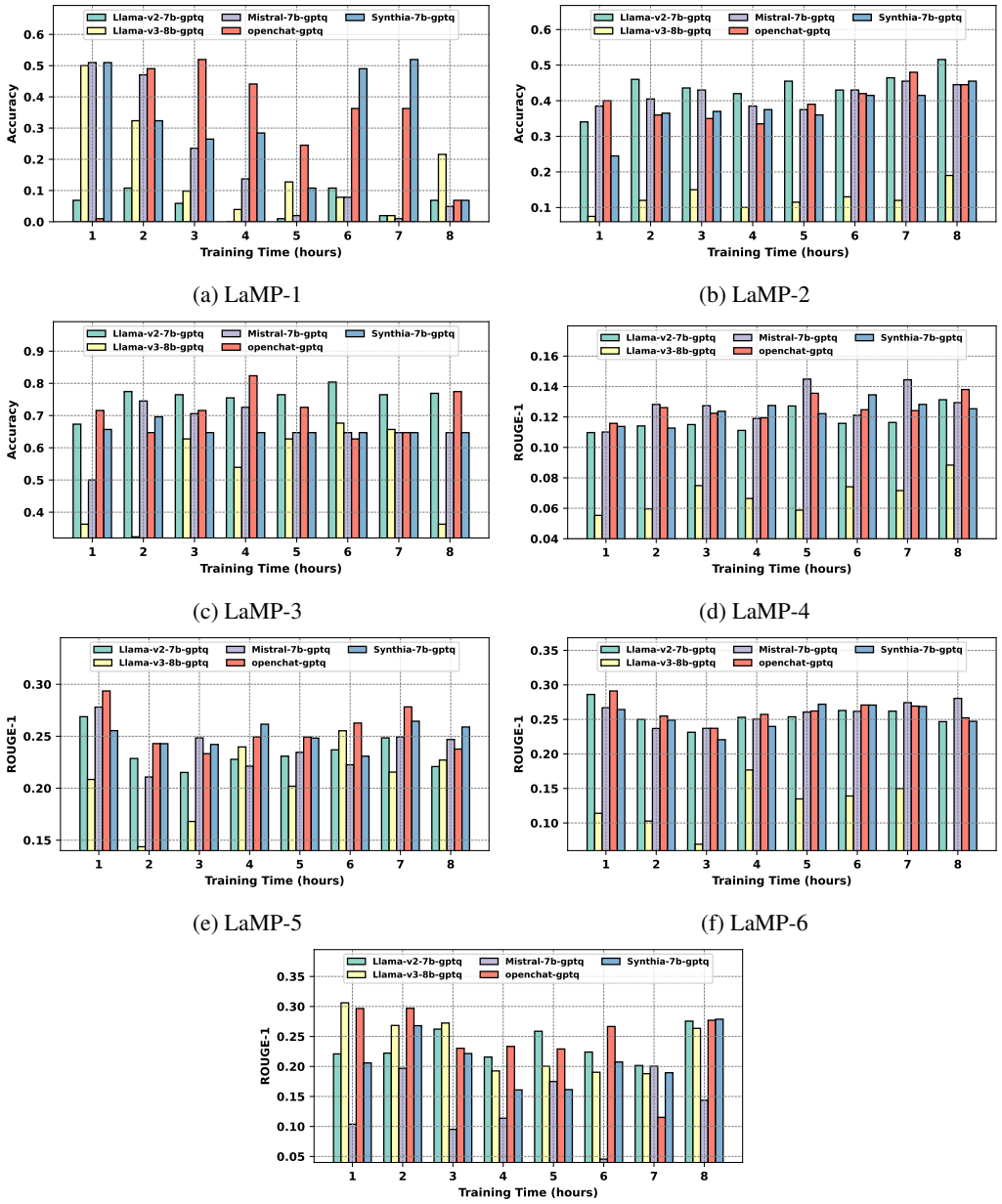
(d) LaMP-4

(e) LaMP-5

(f) LaMP-6

(g) LaMP-7

Figure 22: 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

Figure 23: 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.

C.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 the question "*Quantization, pruning, and knowledge distillation: what are their pros and cons? can we identify the best way to compress the model?*" in section 3.5 and the question "*Do we need to compress the model or can we simply use their original version? What factors can impact our decision?*" in section 3.6.

We first examine Sheared-Llama-1.3b-Prune and Sheared-Llama-1.3b across different data size and different lora setting. The results can be seen in Table 11 and Table 12. Then, we examine Sheared-Llama-1.3b-Prune vs Sheared-Llama-1.3b vs TinyLlama-1.1b. The results can be seen in Figure 24. 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 13 and Table 14. 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 25.

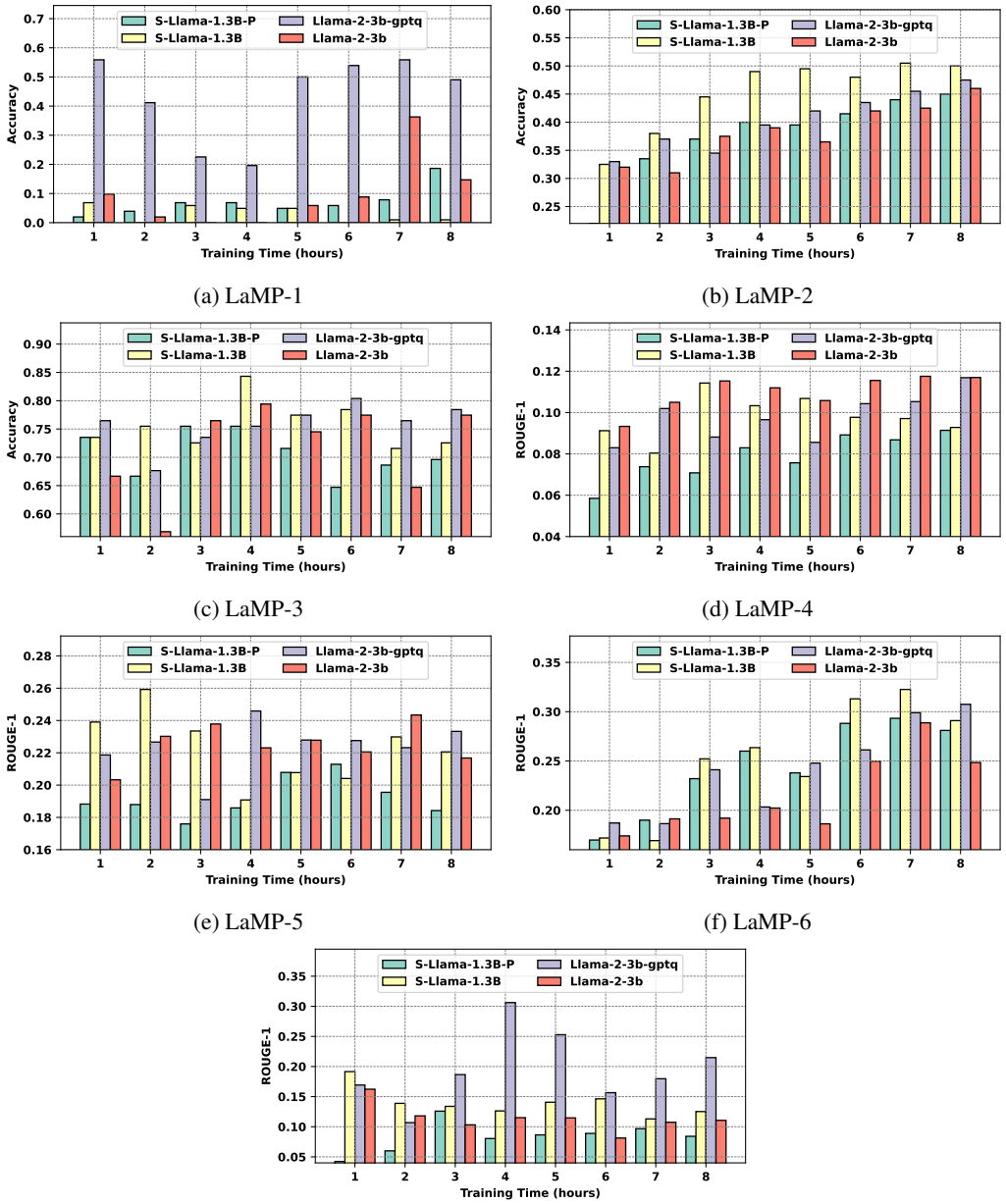
Llama-1.1b	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-6		LaMP-7	
	-P[2]	+P	-P	+P	-P	+P	-P	+P	-P	+P	-P	+P	-P	+P
R(8) A(8)	0.461	0.069	0.510	0.410	0.784	0.686	0.113	0.090	0.205	0.215	0.247	0.268	0.126	0.096
R(8) A(16)	0.245	0.118	0.480	0.445	0.765	0.676	0.099	0.097	0.193	0.188	0.298	0.297	0.129	0.071
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 11: 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	+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

Table 12: 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.

[1]P represents Pruning



(a) LaMP-1

(b) LaMP-2

(c) LaMP-3

(d) LaMP-4

(e) LaMP-5

(f) LaMP-6

(g) LaMP-7

Figure 24: 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.

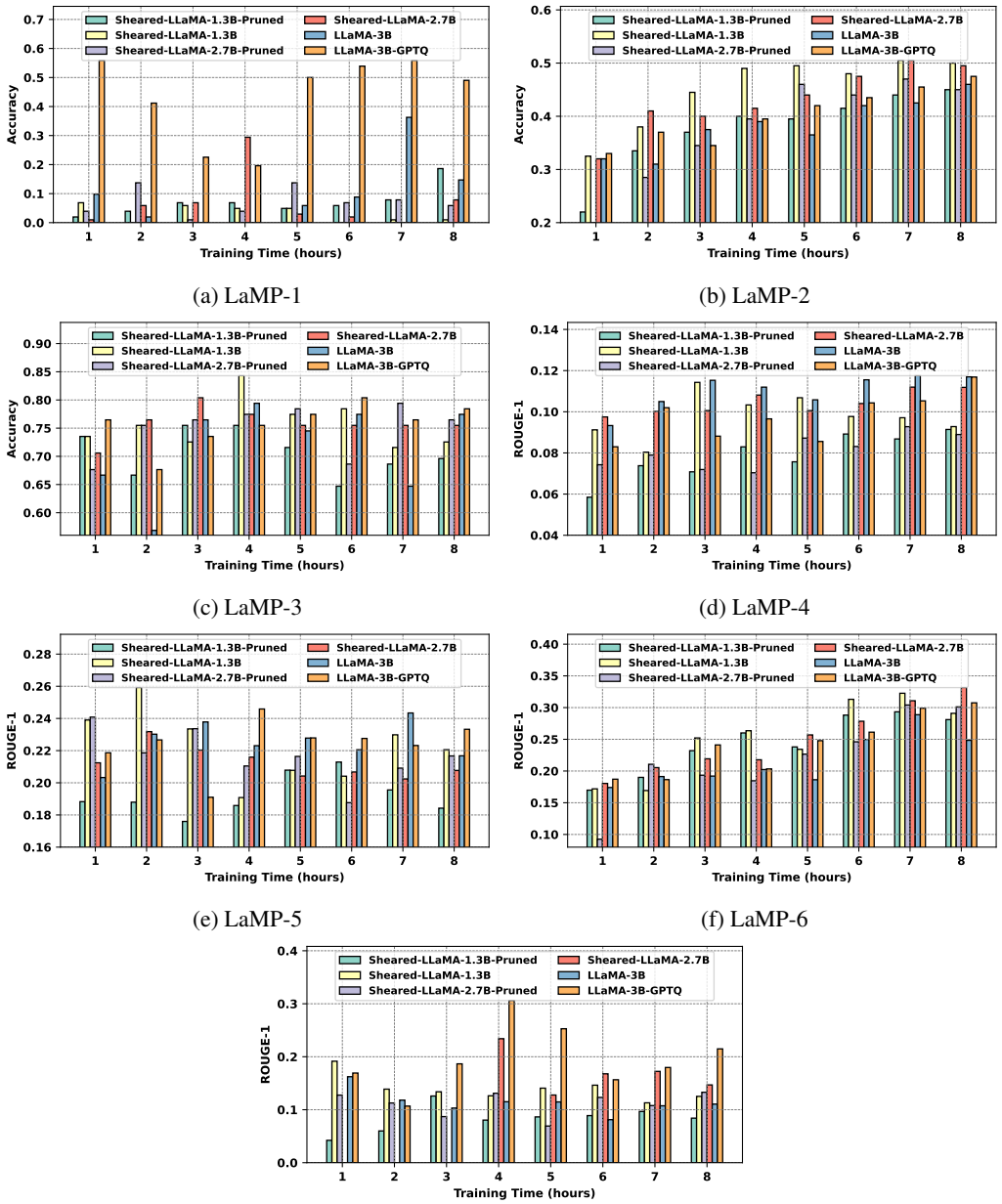
Sheared-Llama-2.7b	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-6		LaMP-7	
	-P[3]	+P	-P	+P	-P	+P	-P	+P	-P	+P	-P	+P	-P	+P
R(8) A(8)	0.020	0.167	0.480	0.455	0.735	0.725	0.112	0.100	0.213	0.221	0.278	0.319	0.001	0.124
R(8) A(16)	0.069	0.245	0.490	0.455	0.784	0.745	0.109	0.089	0.201	0.206	0.286	0.285	0.000	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 13: Performance comparisons between pruned (+P) and unpruned (-P) models with different LoRA settings under training hours of 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

Table 14: 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.

[1]P represents Pruning



(a) LaMP-1

(b) LaMP-2

(c) LaMP-3

(d) LaMP-4

(e) LaMP-5

(f) LaMP-6

(g) LaMP-7

Figure 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.

C.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 15 and Table 16. These experiments can correspond to section 3.5.

		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 15: Performance comparisons between pruned (+P) and unpruned (-P) models with 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	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

Table 16: 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.

C.4 Experiments: Compare the three compression techniques

The experiments examine how each compressed model can perform compared to 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.

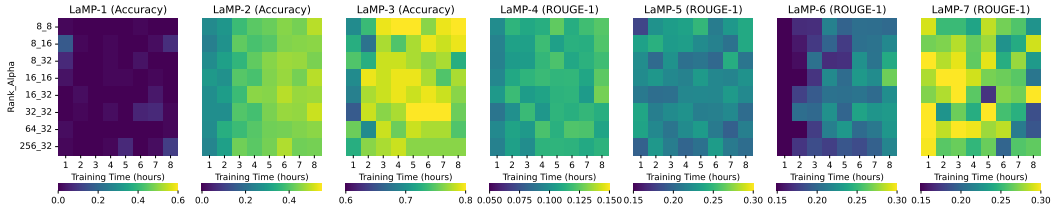


Figure 26: Performances of **Gemma-2b-GPTQ** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

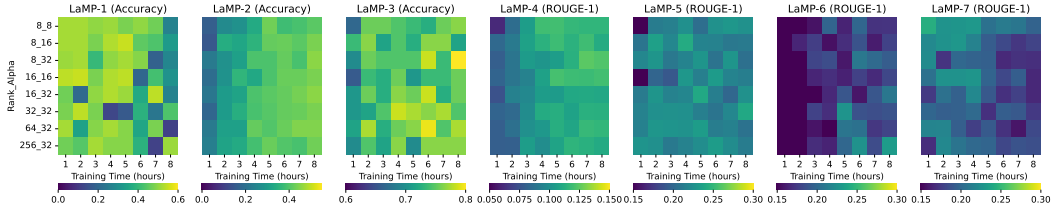


Figure 27: Performances of **TinyLlama-1.1B-GPTQ** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

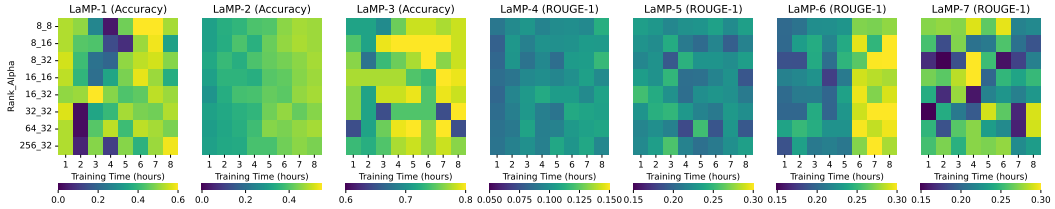


Figure 28: Performances of **Llama-2-3b-GPTQ** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

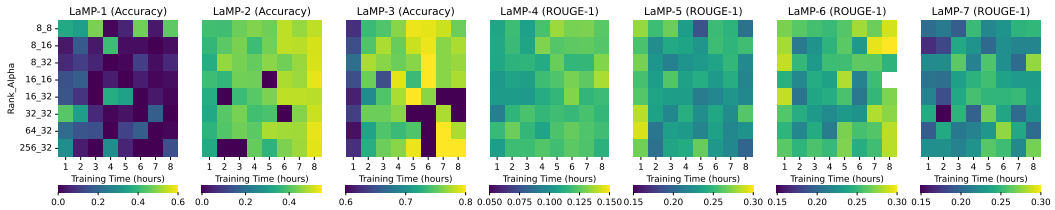


Figure 29: Performances of **Llama-2-7B-GPTQ** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

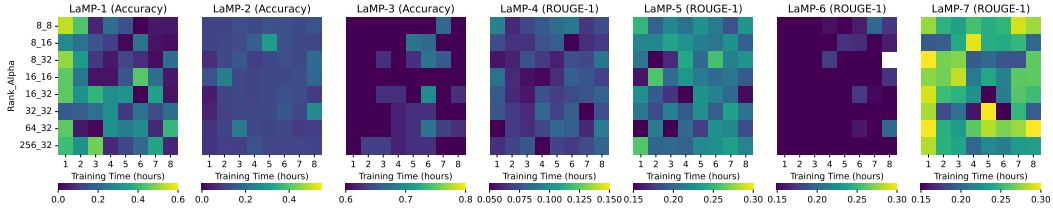


Figure 30: Performances of **Llama-3-8B-GPTQ** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

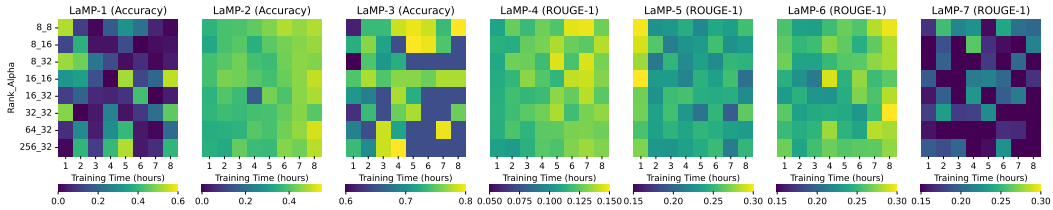


Figure 31: Performances of **Mistral-7b-GPTQ** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

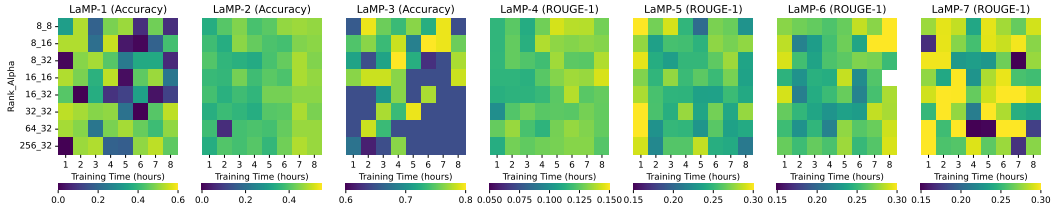


Figure 32: Performances of **OpenChat-3.5-GPTQ** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

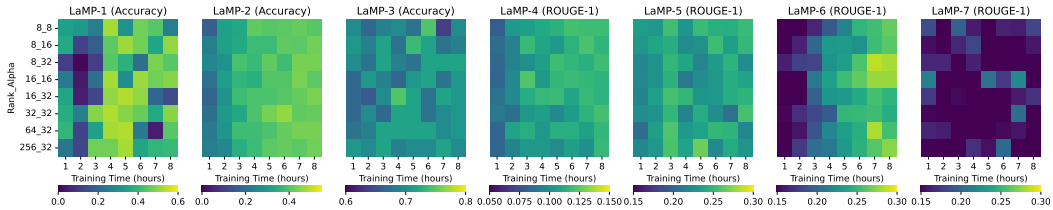


Figure 33: Phi-1.5

Figure 34: Performances of **Phi-1.5** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

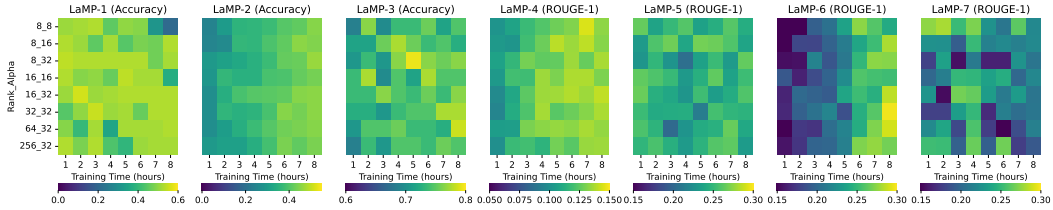


Figure 35: Performances of **Phi-2** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

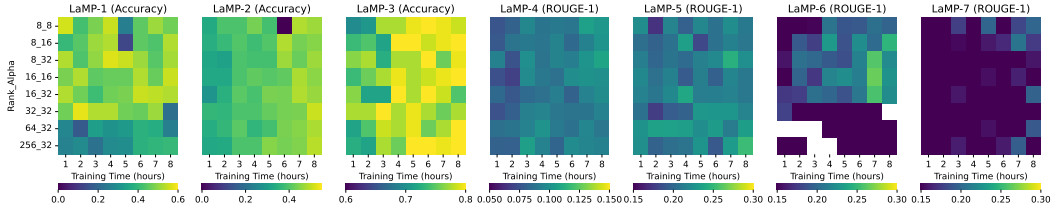


Figure 36: Performances of **Phi-3** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

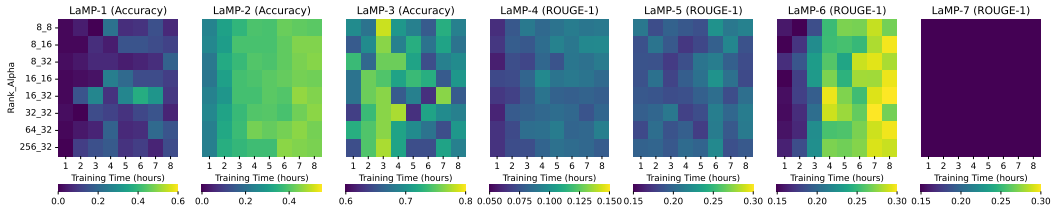


Figure 37: Performances of **Sheared-LLaMA-1.3B-Pruned** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

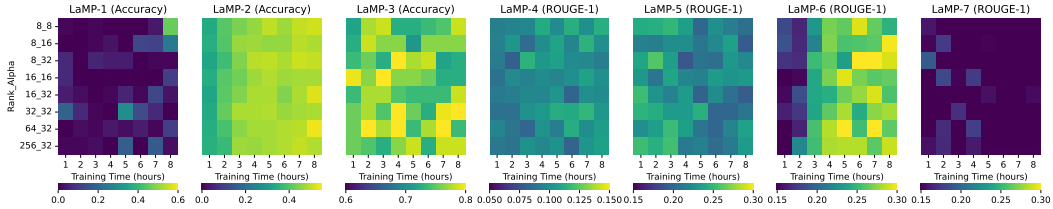


Figure 38: Performances of **Sheared-LLaMA-1.3B** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

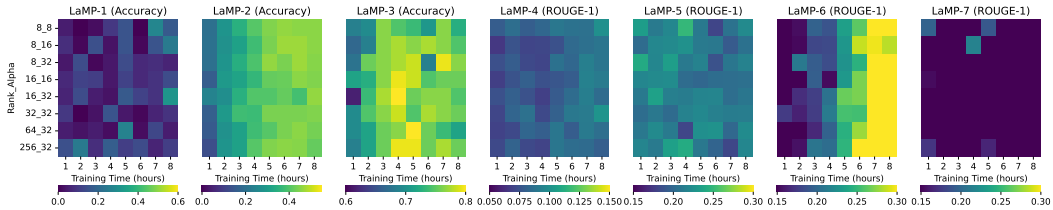


Figure 39: Performances of **Sheared-LLaMA-2.7B-Pruned** on eight commonly used combinations of *alpha* and *rank*, over eight hours training time, across seven datasets.

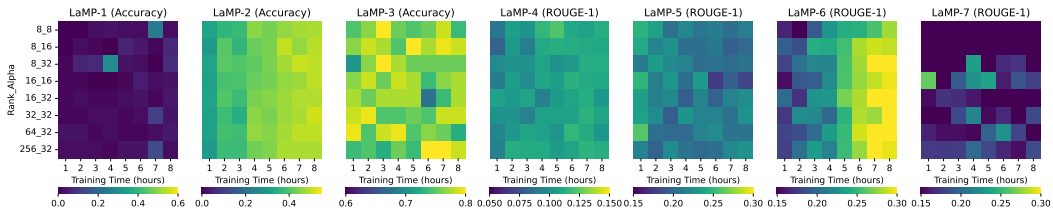


Figure 40: Sheared-LLaMA-2.7B

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

C.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. Thse experiments can correspond to section 3.6

Llama-v2-3b	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-6		LaMP-7	
	-G[1]	+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[1]	+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

Table 17: Performance comparisons between quantized (+G) and quantized (-G) models with different LoRA settings under training hours of 8.

[1]G represents GPTQ, the quantization technique

Llama-v2-3b	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-6		LaMP-7	
	-G[1]	+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[1]	+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[1]	+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[1]	+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

Table 18: 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.

[1]G represents GPTQ, the quantization technique

D Hypotheses behind findings

In this section, we provide (unproven) discussions on the intuitions and theoretical foundations behind the findings. Each subsection here corresponds to a subsection in Section 3. While these discussions shall not be treated as fully verified and correct, we feel the need to provide some thoughts on how future research could be facilitated to provide deeper investigations into those areas.

D.1 Why does RAG work the best with mediocre tasks?

We hypothesize that such behavior is because RAG depends on the semantic understanding ability of the base LLM, and it has an upper bound on improving the LLM's performance. Considering the emergent ability of LLMs [62] which states that some abilities (on solving more complex problems) only present in larger models, it makes sense for small LLMs to work with simple tasks but not difficult ones. On the other hand, the LLMs that could be deployed on the edge are not too big, which means that those 7B-sized models might not have the ability to solve too complex tasks (with RAG). Under those scenarios, fine-tuning at least teaches those models some semantic structure of the answer, which makes parameter learning become better than RAG.

D.2 Why is fixing alpha needed but not rank for LoRA fine-tuning?

We hypothesize that fixing α for those tasks is an implicit measure to avoid overfitting. Since increasing r will increase the model's adaptation towards the training dataset, fixing LoRA's effect on the original model with a constant $\frac{a}{r}$ will cause the model to adapt more towards details of the training dataset, which might cause overfitting issues. Nonetheless, we do acknowledge that the results are rather counterintuitive, and future research needs to be done on this topic, especially for resource-constrained fine-tuning.

D.3 Why does test accuracy not improve after training for some time?

We hypothesize that the interesting behavior of LLMs not improving after some training is due to two possible reasons. First, since the evaluated tasks are personalized datasets, the training data might not be very diverse, which raises the possibility of overfitting. The LLMs might learn patterns associated with specific tokens, rather than the desired features of the target population. Second, due to the resource constraints on the edge, we simulate the practical scenarios and limit the training data size to be rather small. Recent work [63] argues the phenomenon of "pre-scale law", which states that loss during LLM's fine-tuning is not linearly decreasing. At the beginning of fine-tuning, the test loss does not decrease very fast. What we observe in the experiments might correspond to the "pre-power law" and even the previous stage of "pre-power law", where LLM learns an initial adaptation towards the downstream task. In the first stage, the LLM learns the task's representation, which causes performance improvement. Then, it enters the pre-power-law stage, where performance starts to oscillate. Finally, it will reach the power-law stage, but that requires at least 10K-100K samples according to [63], which is almost impossible under the constraint of edge devices.

D.4 Why does using less history data have marginal effects on RAG performance?

Such results are not surprising, especially if we consider that personalized data might not be as diverse as general domain data. On the other hand, these results further validate our hypothesis on the learning curves of parameter learning. Firstly, the user history is not very diverse, which raises the possibility of overfitting. Secondly, by providing grounded context on the target's format and basic information, RAG lifts the LLM to the stage before "pre-power law". In this case, the intertwined performance of RAG and PEFT on some tasks, as shown in Figure 2.b, aligns with the hypothesis that PEFT will reach the "pre-power law" stage after a few hours.

D.5 What are the foundations behind each compression technique?

From the results, we see that model distillation is the most stable method to fit a large LLM on the edge, but it seldom (never) stands out as the best option. On the other hand, GPTQ is not very stable, but it can reach the best performance most of the time with some combination of LoRA parameters and training data. Shearing and pruning are just not very good compared to those two methods.

First of all, it should be evident why sheared and pruned models are not suitable for edge devices. Pruning is a technique that removes certain weights in the model, but because those models need to be fine-tuned for downstream tasks, those missing weights will inevitably be added back during fine-tuning. Thus, pruned models neither have advantages in RAM nor inherit semantic understanding ability from large models. Then, we hypothesize that the quality and memorization of pre-training datasets affect the fine-tuning of those models. It is well known that LLMs memorize part of their training datasets [22, 64], but two things remain unknown. The first one is that we do not know the quality of the memorized data (i.e. do they memorize only high-quality/significant data or anything?), and the second one is that **where** they memorize the data. It is possible that models with low-quantity pre-training data will have unstable behavior with downstream tasks, but further investigations are needed to prove this.

D.6 Why is a compressed model sometimes better?

We hypothesize that such a phenomenon shares the same intuition with the previous subsection. 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 pre-training information that is stored in the less significant bits of the LLM and makes the fine-tuning faster.

E Background and Related Works

E.1 Large Language Models Customization for Downstream Tasks

Large Language Models (LLMs), such as GPT-4 [18], BERT [65], and T5 [66], are pre-trained on vast amounts of text data [67]. 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 [68, 69]. 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 [68]. The dominant approach uses variations of low-rank approximation (LoRA) techniques [26, 70, 71, 72], 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 [73] and retrieval-augmented generation (RAG) [28]. 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.

E.2 Large Language Model Evaluations

The field of Large Language Models (LLMs) has seen a significant amount of research focusing on various aspects of model evaluation [74, 75, 76]. Numerous studies have proposed benchmarks to systematically evaluate the capabilities of LLMs across different tasks. On the benchmark side, notable benchmarks include MMLU [77], GPQA [78], HumanEval [79], and GSM-8K [80], but they are not suitable for edge LLMs due to the reasons explained in the introduction section. In this paper, we focus on the LaMP dataset [45], 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 [81]. Optimal fine-tuning techniques have been extensively studied by works like [82], which provides a framework for choosing the optimal fine-tuning techniques given the task type and data availability, and [63] which 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 being unable to provide a guidebook for edge LLMs.