MobiLoRA: Accelerating LoRA-based LLM Inference on Mobile Devices via Context-aware KV Cache Optimization

Anonymous ACL submission

Abstract

Deploying large language models (LLMs) with low-rank adaptation (LoRA) on mobile devices is promising due to their capability to complete diverse domain-specific tasks while ensuring privacy and accessibility. In this paper, we introduce MobiLoRA to accelerate LoRA-based LLM inference on mobile devices. MobiLoRA focuses on optimizing the key-value (KV) caches due to the limited computing and memory resources of mobile devices. The key insight of MobiLoRA lies in the utilization of two *contexts* for on-device LoRA serving: semantic-level contexts, such as prompts with shared prefixes, and system-level contexts, such as the application status (e.g., foreground or killed) of LLM requests. Specifically, for semantic-level contexts, MobiLoRA proposes similarity-aware delta encoding, which leverages token-wise similarity in KV caches across LoRA adapters for efficient storage and reuse. Furthermore, MobiLoRA advocates context-aware KV cache management to optimize cache eviction considering the system-level contexts. We implement MobiLoRA and compare it with state-of-the-art LLM serving frameworks using real-world mobile device traces. Results show that MobiLoRA accelerates LoRA-based LLM inference by 18.1%~80.5% on mobile devices.

1 Introduction

Deploying pre-trained large language models (LLMs) directly on mobile devices (e.g., smartphones) is crucial considering data privacy and service accessibility (Yi et al., 2023a; Kong et al., 2024b). To specialize pre-trained models for diverse domain-specific demands on the device, *low-rank adaptation* (LoRA) (Hu et al., 2022; Dettmers et al., 2024) is a widely used parameter-efficient fine-tuning technique. LoRA retains the base model parameters and introduces plug-and-play *adapters* to Transformer layers for fine-tuning, typically with a size of tens of megabytes. Major mobile device vendors such as Apple and Google heavily rely on LoRA adapters for their on-device intelligent services (Gunter et al., 2024; Android Developers, 2023).

Given the promise of LoRA-based LLMs, serving a set of adapters with the base model efficiently attracts much attention. Researchers make efforts to serve numerous LoRA adapters in the datacenter (Wu et al., 2024a; Sheng et al., 2024; Chen et al., 2024; Kong et al., 2024a). Focusing on throughput, these works merge multiple LoRA adapters with the base model and leverage handcrafted CUDA kernels to support the batched inference of the fused model. In contrast, for on-device LLM inference, latency metrics such as time-to-first-token (i.e., TTFT) are critical since real-time interaction and handling of individual requests are common on mobile devices. Other mainstream LLM serving frameworks (Kwon et al., 2023; Gao et al., 2024b; Liu et al., 2024; Lin et al., 2024) focus on storing the intermediate states, i.e., key-value (KV) cache, to avoid repetitive computation across requests with shared prefixes. The limited computing and memory resources on mobile devices underscore the importance of reusing KV cache. However, existing approaches are not directly applicable to LoRA-based LLMs because KV cache for different adapters is not reusable even when requests are identical. Hence, serving LoRA-based LLMs efficiently is challenging on mobile devices.

Fortunately, two unique opportunities on mobile devices are underexploited for accelerating LoRA-based LLM inference, namely *semantic-level* and *system-level contexts*.

(1) *Reusing semantic-level contexts.* During daily usage of mobile devices, requesting different LoRA adapters with the same semantic-level contexts, i.e., prompts and user inputs, is common (Hong et al., 2023; Wu et al., 2024b). For instance, users first use a proofreading adapter to refine the text when writing an email and then con-

dense the same paragraph with a summarization adapter. Although the KV caches are different for the same tokens on different adapters, our preliminary experiments show that they exhibit high token-wise similarity. This similarity facilitates the efficient encoding and reusing of the KV caches for LoRA adapters under semantic-level contexts.

(2) *Exploiting system-level contexts.* Besides the semantic-level contexts, another unique characteristic of on-device LLM inference is the easy access to system-level contexts such as application status that queries the LLM, e.g., foreground active or killed. Utilizing this kind of context brings a broader optimization space for efficient LLM serving on mobile devices. For example, when a user kills an application, KV caches associated with that application's queries are probably not reused. It is prudent to evict the cache and free up space for other active applications.

Leveraging the above opportunities, we propose **MobiLoRA** to accelerate the inference of LoRA-based LLMs on mobile devices. Considering the limited resources on mobile devices, we introduce a new attention mechanism, `CtxAttention`, to enhance the reusability of the KV cache via on-device contexts. Based on `CtxAttention`, for semantic-level contexts, MobiLoRA proposes *similarity-aware delta encoding* for the KV cache of shared prefixes on different LoRA adapters, facilitating its efficient storage and reuse. For system-level contexts, MobiLoRA leverages a *context-aware KV cache management* to optimize the preservation and eviction of the KV cache. Beyond the widely-used management based on least recently used (LRU) (Zheng et al., 2023; Kwon et al., 2023), MobiLoRA involves the application status that queries the LLM when deciding KV cache eviction. We implement MobiLoRA on top of the state-of-the-art LLM serving system, SGLang (Zheng et al., 2023), and extensively evaluate the framework based on real-world mobile application usage traces. Results show that MobiLoRA accelerates the on-device LoRA-based LLM inference by 18.1%~80.5% in terms of time-to-first-token (TTFT). This paper makes the following contributions:

- To the best of our knowledge, this is the first work to optimize the KV cache of LoRA-based LLM on mobile devices. This optimization is motivated by our observation of utilizing semantic- and system-level contexts for inference efficiency improvement.

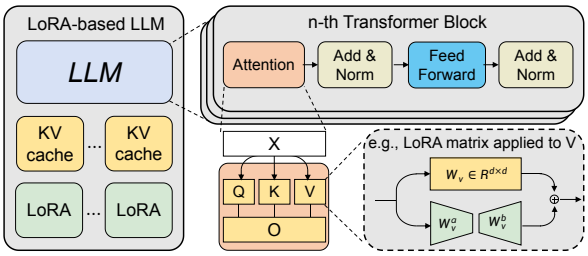

Figure 1: On-device deployment of LoRA-based LLM, exemplified using `FinGPT` adapter.

- Based on our observations, we propose a similarity-aware delta KV cache encoding used by different LoRA adapters and a context-aware KV cache management strategy for efficient on-device KV cache reuse.
- We implement MobiLoRA and extensively evaluate its performance. Results show that our approach enhances the efficiency of on-device natural language processing tasks.

2 Background and Motivation

In this section, we first introduce the on-device LoRA-based LLM and the difficulty of reusing the KV cache with LoRA. We then present the observations that guide the design of MobiLoRA.

2.1 Basics of On-device LoRA Serving

LoRA-based LLM for mobile devices. LoRA-based fine tuning of LLM is now widely adopted on mobile devices. Major mobile device manufacturers leverage LoRA in their on-device intelligent services, such as Apple Intelligence (Apple, 2024) and Android AICore (Android Developers, 2023).

To meet diverse daily demands with constrained resources, on-device LoRA-based services generally follow a single-model, multiple-adapters approach, illustrated in Fig. 1. The LoRA adapters are selectively activated according to the requests. For example, Apple Intelligence leverages an LLM with 3B parameters as the base model and offers various LoRA adapters for different scenarios, such as writing tools that include text proofreading and rewriting, notification prioritizing, and smart reply.

Challenging to reuse LoRA-based KV cache. Due to the autoregressive nature of LLM inference, the generation of each token uses the hidden state of all its preceding tokens. Storing these hidden states, referred to as key-value (KV) cache, for future token generation is able to avoid repeated computation. Therefore, utilizing the KV cache has become a popular technique for accelerating LLM inference. Various KV cache frameworks are

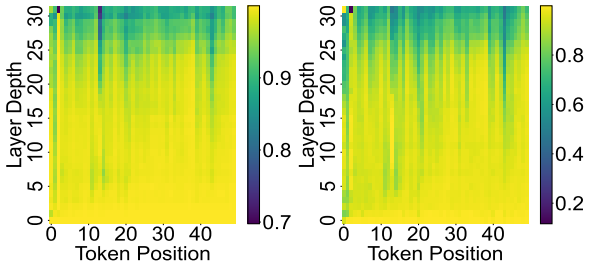

Figure 2: Key (left) and value (right) cache similarity of based pretrained model and LoRA fine-tuned model.

proposed such as vLLM (Kwon et al., 2023) and SGLang (Zheng et al., 2023). Apart from using the KV cache within a single request, these frameworks also investigate the potential to reuse the cache in multi-turn conversations or across different requests. The rationale behind this opportunity is KV cache can be reused between the prompts that share the same prefix.

However, for LoRA-based LLMs, direct KV cache reuse across LoRA adapters is impractical. We exemplify with `FinGPT` LoRA adapter fine-tuned on Llama2-7b based model. `FinGPT` applies low-rank matrices to the calculation of Q, K, and V of each Transformer layer. Fig. 1 shows the example of the calculation of V. LoRA converts the original calculation $V = xW_v$ to $V' = xW_v + xW_v^a W_v^b$, where x is the attention input, $W_v \in \mathbb{R}^{h \times d}$ is the projection matrix of V. $W_v^a \in \mathbb{R}^{h \times r}$, $W_v^b \in \mathbb{R}^{r \times d}$ are low-rank matrices with rank r. Similarly, we have the attention score with LoRA:

$$Attn_{LoRA} = \text{softmax}(\frac{Q'K'^T}{\sqrt{d_k}}V'), \qquad (1)$$

where Q' and K' are the updated values of Q and K because of LoRA, d_k is the hidden dimension. The KV cache with LoRA (K', V') is different from the ones without LoRA (K, V), and the non-linear softmax operation further leads to the reuse of the KV cache impractical. To make things worse, due to the autoregressive generation, the discrepancy between with and without LoRA propagates through deeper layers and subsequent tokens.

2.2 Opportunities of Exploiting Contexts

Semantic-level context. During the daily use of mobile devices, the semantic-level context, i.e., similar or even identical prompts, sent to different LoRA adapters are common. To investigate how to efficiently reuse the semantic-level context, we conduct a preliminary experiment. We feed the same prompt excerpted from ShareGPT (ShareGPT, 2023) to the base Llama-7B model and `FinGPT` LoRA fine-tuned model. We then compare the similarity of the KV cache of the two models, illustrated in Fig. 2. We have the following observations.

Obs. #1: KV cache similarity exists among different LoRAs with the same prompt. We observe a maximum 97% and 95% similarity in key and value cache. With this similarity, there exists an opportunity to store only incremental differences of the KV cache from different LoRAs to reduce the memory consumption, which is especially limited on mobile devices.

Obs. #2: Token-wise decreasing pattern of similarity is observed. Specifically, the similarity is more pronounced in the shallow Transformer layers, and it decreases as the layer goes deeper. The rationale behind this pattern is as the layer goes deeper, more LoRA outputs are merged with the base model's output, leading to more different KV tensors. How to exploit this observation to enhance the KV cache encoding efficiency requires substantial design of MobiLoRA.

System-level contexts. KV cache management, e.g., eviction, is necessary since the limited resource on mobile devices prevents the cache grow continuously. However, commonly used LRU-based eviction falls short in certain scenarios. For example, when an application is terminated by users, the KV caches of the LLM requests sent by the terminated application are typically no longer needed and should be evicted first. This system-level context is hardly accessible for serving frameworks in datacenters since their requests are initiated by external applications. In terms of on-device serving, MobiLoRA has easy access to this context, leading to a new horizon for KV cache management. Hence, we have the following observation:

Obs. #3: Leveraging the system-level contexts is beneficial to the efficient KV cache management.

3 MobiLoRA Design

Guided by the opportunities mentioned above, we design MobiLoRA. Fig. 3 shows the overall system architecture. The core of MobiLoRA is a new attention mechanism, `CtxAttention`, which facilitates the exploitation of contexts to manage LoRA KV cache. Based on `CtxAttention`, our system proposes a similarity-aware KV cache delta encoding mechanism for efficient LoRA KV cache storage with semantic-level contexts and a context-aware KV cache management policy with the consideration of system-level contexts.

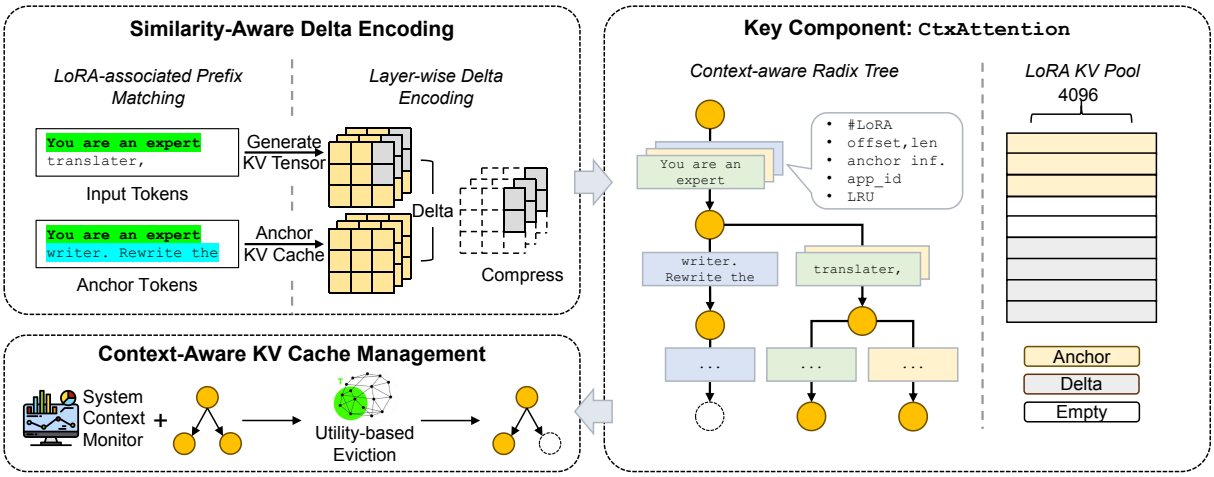

Figure 3: MobiLoRA architecture overview.

3.1 `CtxAttention` for LoRA-based LLM

To leverage the potential of on-device contexts, we introduce a new attention mechanism, `CtxAttention`, to facilitate the context-aware KV cache reuse for LoRA-based LLMs. As shown in Fig. 3, `CtxAttention` includes a context-aware radix tree and a LoRA KV pool. Inspired by RadixAttention, `CtxAttention` leverages a radix tree to map the cached token sequences to their KV cache tensors and further enhances it with the following two aspects.

To reuse cross-LoRA semantic-level contexts, `CtxAttention` extends the radix tree to store multiple mapping information (`offset` and `len` to KV pool) for different LoRA instances (`#LoRA`) at the same edge of the tree. In the LoRA KV pool, the KV cache tensors of the first recorded LoRA request are stored in their original form, referred to as the anchor tensor. Subsequent KV cache tensors of other adapters are encoded as the difference from the anchor KV, known as delta tensors, to improve storage efficiency. This anchor information is also stored in the context-aware radix tree. With the above context information, the attention score with LoRA in Eq. 1 can be transformed as follows with the reuse of anchor KV cache:

$$Attn_{LoRA} = \text{softmax}[\frac{Q(K_A \star K_\Delta)^T}{\sqrt{d_k}}(V_A \star V_\Delta)], \quad (2)$$

where K_A and V_A are the anchor key and value tensors, i.e., existing KV cache. K_Δ and V_Δ are the delta tensors. Operator $\star$ denotes the decoding of the delta tensor with its anchor tensor.

To exploit on-device system-level contexts, `CtxAttention` additionally record the application id (`app_id`) besides LRU information. These data help MobiLoRA to improve the KV cache management with the understanding of the application state on mobile devices.

3.2 Similarity-aware Delta KV Encoding

Based on the aforementioned token-wise similarity (Obs. #1) among different LoRA adapters, MobiLoRA leverages a *delta encoding* method to efficiently store and reuse the KV cache. The encoding process includes the following two steps. First, LoRA-associated prefix matching determines which input tokens should be encoded with delta. Then, a layer-wise delta encoding calculates the delta considering the KV cache similarity.

LoRA-associated prefix matching. With the `CtxAttention` mechanism, when a new request arrives, MobiLoRA compares the prefix of the prompt in the radix tree to find a matched prefix. If matches and the input LoRA is different from the existing KV cache, the similarity-aware delta encoding is triggered. If there is no matched prefix, MobiLoRA will store the KV cache of the input as the anchor tensor and create a new edge in the radix tree with the inputs.

Layer-wise delta encoding. Aligned to various quantization schemes of LLMs, KV caches have different data types such as 8-bit integer (`INT8`) and 16-bit floating-point (`FP16`). For the integer KV cache, calculating the delta of tensors with high similarity decreases the absolute value of the tensor, making it possible to use fewer bits to represent it. Therefore, MobiLoRA directly leverages arithmetic coding (Liang et al., 2018) for encoding.

However, encoding the floating-point KV cache, which is more common in real-world deployment, faces non-trivial challenges. Despite the relatively

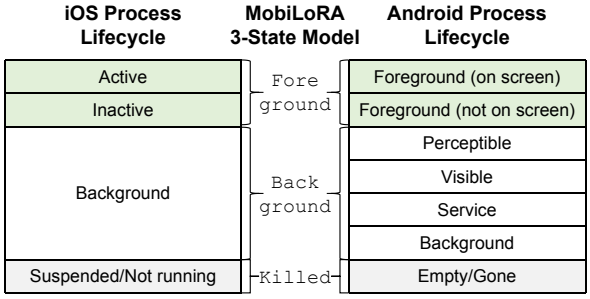

Figure 4: Application state classification of major mobile OSes and the three-state model of MobiLoRA.

small absolute value of the floating-point delta, the strong randomness of the ending mantissa bits in its representation makes it difficult to achieve a high lossless compression ratio. Considering the limited computation and storage capacity of mobile devices, we encode the floating-point delta by an error-bounded quantization for a high compression ratio. Take the key cache encoding as an example. We use K_I to denote the input key cache tensor being encoded with an existing K_A. Following the idea of `sz` compression, we calculate an error-bounded delta quantization between K_I and K_A:

$$K_\Delta = \lfloor \frac{K_I - K_A}{2\log(1+\epsilon)} + 0.5 \rfloor, \tag{3}$$

where K_Δ is the resulting error-bounded and quantized representation of the tensor delta. The selection of error-bound parameter ϵ (e.g., 1E-4, 1E-5) is the key to balancing the encoding precision and compression ratio. Driven by the insight of decreasing pattern of similarity (Obs. #2), we apply more relaxed error bounds for deeper layers. Specifically, for each token, we continuously monitor its KV cache similarity against the anchor tensor. We split the layers into multiple similarity groups, i.e., layers with high similarity (>97.5% in our current implementation), medium similarity, and moderate similarity (<85%). We then apply different ϵ to each group: 1E-4 for high similarity, 1E-3 for medium similarity, and 1E-2 for moderate similarity. The encoding of the value cache is similar.

It is worth noting that these error-bound parameters are not empirically set. Parameters should be adjusted for specific base model architectures, taking into account the number and dimension of attention heads.

3.3 Context-aware KV Cache Management

We first demonstrate how system-level contexts are recorded and propose a utility-based KV cache eviction mechanism instead of LRU for efficient on-device LoRA inference.

Algorithm 1 Context-aware KV Management

Input: Prefix tree nodes $n \in \mathbb{N}_t$; Input KV cache n_{in}; KV cache size function size(); Utility function $U()$; Memory budget $\mathbb{M}$;
Output: $\mathbb{N}_t$ for every time step t
Initialize: $\mathbb{N}_0 = \emptyset$, $U(\mathbb{N}_0) = 0$, $\mathbb{N}_{\text{evict}} = \emptyset$
for each time step t **do**
  Update $U(n)$ for each $n \in \mathbb{N}_{t-1}$
  if size$(\mathbb{N}_{t-1})$ + size$(n_{\text{in}}) \leq \mathbb{M}$ **then**
    $\mathbb{N}_t \leftarrow \mathbb{N}_{t-1} \cup \{n_{\text{in}}\}$
  else
    while size$(\mathbb{N}_{evict})$ < size(n_{in}) **do**
      $\mathbb{N}'_t \leftarrow \mathbb{N}_{t-1} \cup \{n_{\text{in}}\}$
      $n_{\text{evict}} \leftarrow \arg\min_{n_i \in \mathbb{N}'_t} U(n_i \mid \mathbb{N}'_t \setminus n_i)$
      $\mathbb{N}_t \leftarrow \mathbb{N}'_t \setminus \{n_{\text{evict}}\}$
      if $n_{\text{evict}} \neq n_{\text{in}}$ **then**
        $\mathbb{N}_{\text{evict}} \leftarrow \mathbb{N}_{\text{evict}} + n_{\text{evict}}$
      end if
    end while
  end if
end for

System-level context representation. As shown in Fig. 4, the state transitions during the application process lifecycle on different mobile operating systems (OSes) are different (Zheng et al., 2024; Lee et al., 2016). By design, MobiLoRA is supposed to serve as an OS-agnostic middleware for LLM serving. Hence, we map both lifecycle models to a general three-state model, i.e., *foreground*, *background*, and *killed*. We implement a lightweight state monitor as a plug-in to various mobile systems that tracks the state transitions of all applications associated with KV caches.

Utility-based KV cache eviction. Evicting stale KV caches when the cache pool is full is a critical management consideration in MobiLoRA. Taking advantage of the system-level contexts, MobiLoRA ranks the KV caches by their *utility* for future reusing. We define the utility of a KV cache node n in the `CtxAttention` radix tree with three parts: the application state score $\mathbb{S}(a)$ of application a associated with n, the LRU score $\mathbb{T}(n)$, and the length of the KV cache $\mathbb{L}(n)$. Specifically, we have the following formulation:

$$U(n) = \lambda_s \phi_s \left(\sum_{a \in A_n} \mathbb{S}(a) \right) + \lambda_t \phi_t \left(\mathbb{T}(n) \right) + \lambda_l \phi_l \left(\mathbb{L}(n) \right). \tag{4}$$

Here, A_n denotes the application set associated with node n. λ_s, λ_t, and λ_l are hyperparameters that controls the focus of the three scores. ϕ_s, ϕ_t, and ϕ_l are monotone, non-negative functions, which ensures the $U(n)$ to be submodular (Kumari et al., 2024; Bilmes, 2022).

During cache eviction, nodes with the lowest $U(n)$ are purged first, ensuring foreground app caches persist under memory pressure while obsolete entries are reclaimed proactively. This mechanism bridges system-level contexts with KV cache management, leading to optimized user-perceived responsiveness and memory efficiency. Since $U(n)$ is a submodular function, according to the properties of submodular functions (Bilmes and Bai, 2017), we leverage a greedy algorithm to obtain a suboptimal solution within a factor of $(1 - 1/e)$ in a finite number of steps. Hence, we have the context-aware KV management algorithm in Alg. 1. For each time step, MobiLoRA updates the utility of each node in the prefix tree (line 5). Then, MobiLoRA determines if it needs to evict the KV cache according to the memory budget (line 6). If the memory budget is not reached, the input KV cache is directly stored in the KV pool (line 7). Otherwise, MobiLoRA leverages a greedy algorithm to select the stale cache nodes with low utility scores with Eq. 4 and evict it (lines 8-16).

4 Evaluation

In this section, we first introduce the implementation details and the experiment setup of MobiLoRA. Then, we show the performance of MobiLoRA in the following aspects: the end-to-end performance to reduce the time-to-first-token latency, the generation quality with our delta encoding, and the ablation study.

4.1 Experiment Setup

We implement MobiLoRA on the state-of-the-art LLM serving framework, SGLang (Zheng et al., 2023). The similarity-aware delta KV encoding is implemented using the open-source data compression library, `sz`. We conduct evaluations on the widely-used mobile development platform, NVIDIA AGX Orin, under the experiment setups specified below.

Scenarios. We use Llama2-7B (Touvron et al., 2023) as the base model, and we obtain ten real-world open-source LoRA adapters in our evaluation. To evaluate the MobiLoRA's performance in various configurations, we select five evaluation scenarios with different numbers of LoRA adapters, memory budgets for the KV cache pool, and the max input length of each request. The detailed configurations are depicted in Tab. 1. All the models and adapters are collected from HuggingFace.

Table 1: Evaluation scenario configurations.

Scenarios	S1	S2	S3	S4	S5
# LoRA adapter	5	5	5	10	10
Memory budget (GB)	2.0	4.0	4.0	2.0	4.0
Max input len. (token)	1024	1024	2048	1024	2048

Tasks and workloads. We mainly consider two natural language processing (NLP) tasks that are popular on mobile devices. (1) *Conversation* task represents the LLM-empowered chatbots on mobile devices similar to Apple Siri and Samsung Bixby. We use ShareGPT (ShareGPT, 2023) dataset for this task. (2) *Writing* task is another popular LLM-base tasks that heavily rely on LoRA adapters, such as the writing tools of Apple Intelligence. We use Xsum (Narayan et al., 2018) dataset for this task.

Due to the absence of a real-world LoRA request trace dataset for mobile devices, we synthesize workload traces using the China-telecom dataset (Yu et al., 2018) for application usage traces for the above two NLP tasks. We tokenize each request to simulate arrival patterns with different adapter distributions. The adapters in the dataset follow the Pareto distribution, representing concentrated usage of frequently used apps. All datasets are downloaded from their public websites and conform to their intended usage.

Baselines. We use various state-of-the-art LLM serving engines as comparison baselines. (1) *Huggingface PEFT* (Mangrulkar et al., 2022), which is the default inference engine for HuggingFace. (2) *vLLM* (Kwon et al., 2023), which introduces PagedAttention for efficient KV cache memory allocation. (3) *S-LoRA* (Wang et al., 2020; Zheng et al., 2023), which is built on SGLang and enhances the LoRA serving ability.

For vLLM, we control the GPU memory preallocated for the KV cache by setting the `gpu_memory_utilization` parameter in the engine, ensuring it aligns with the specified memory budget. Similarly, we achieve the same objective by configuring an analogous `past_key_values`, which regulates the length of key-value pairs transmitted to the forward computation. As for S-LoRA,

Table 2: Time-to-first-token performance comparison of MobiLoRA under different scenarios (unit: second).

		PEFT	vLLM	S-LoRA	MobiLoRA
Conversation	S1	0.554	0.533	0.282	**0.183 (↓35.1%~67.0%)**
	S2	0.561	0.486	0.241	**0.158 (↓34.4%~71.8%)**
	S3	0.678	0.959	0.399	**0.197 (↓50.6%~79.5%)**
	S4	0.685	0.543	0.586	**0.397 (↓26.9%~42.0%)**
	S5	0.586	0.959	0.648	**0.480 (↓18.1%~49.9%)**
Writing	S1	0.520	0.767	0.281	**0.174 (↓38.1%~77.3%)**
	S2	0.517	0.863	0.255	**0.168 (↓34.1%~80.5%)**
	S3	0.563	0.764	0.327	**0.207 (↓36.7%~72.9%)**
	S4	0.542	0.902	0.627	**0.342 (↓36.9%~62.1%)**
	S5	0.586	1.147	0.745	**0.392 (↓33.1%~65.8%)**

we set the size of `TokenToKVPool` to align with the budget, since the S-LoRA is already integrated in SGLang.

4.2 End-to-End Performance

Time-to-first-token performance. We compare MobiLoRA with the four aforementioned baselines. We choose TTFT as the main performance metric, it's crucial for assessing the quality of service in LLM deployment. As shown in Tab. 2, each row presents the TTFT results obtained by different serving systems under the corresponding simulated scenarios. Remarkably, MobiLoRA reduces the TTFT by at most 80.5% over the state-of-the-art baselines. The best-performed baseline is S-LoRA since it is a dedicated serving system for LoRA-based LLMs. S-LoRA efficiently serves multiple adapters simultaneously by loading them into memory, demonstrating superior performance compared to PEFT and vLLM. We adopted S-LoRA's adapter loading strategy. However, MobiLoRA surpasses S-LoRA in performance due to our ability to reuse the KV cache for each LoRA adapter. MobiLoRA only prefills the new input of the new conversation. Moreover, MobiLoRA can load and reuse the KV cache of different LoRA adapters at the cost of minimal memory usage.

We then analyze the performance of MobiLoRA under different scenarios. We can see from Tab. 2 that MobiLoRA performs better in harsh scenarios such as S1 in both conversation and writing tasks. This demonstrates the effectiveness of MobiLoRA in achieving efficient LoRA-based LLM inference on resource-constrained devices, opening up new possibilities for NLP tasks on these devices. The quantity of LoRa adapters also impacts the end-to-end performance. Comparing the performance of S1 against S4 in both tasks, we observe a significant TTFT degradation from S1 to S4. This is because having more adapters can potentially reduce the KV cache hit ratio, which in turn may limit the opportunity to reuse the KV cache, resulting in slower performance.

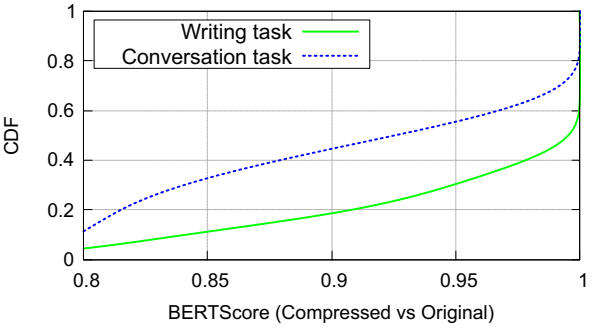

Figure 5: BERTScore performance comparison of with and without similarity-aware delta encoding.

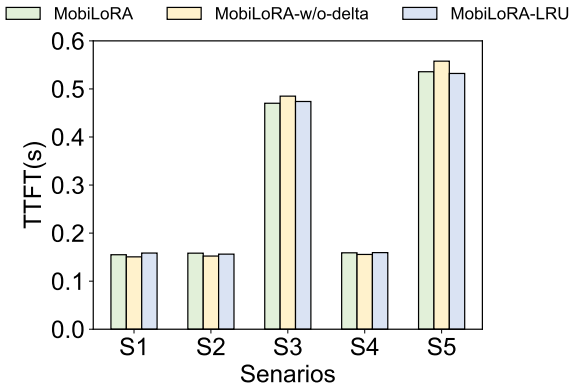

Figure 6: The TTFT of MobiLoRA and its variants on synthesized workload traces with different scenarios.

Generation quality. We then evaluate the generation quality performance with and without similarity-aware delta encoding we proposed in §3.2. We use BERTScore as the main metric and compare it for writing tasks and conversation tasks in Fig. 5. We can see from the figure that on both tasks, our similarity-aware delta encoding has little influence on the generation accuracy. We owe this merit to our layer-wise delta encoding, which selects an optimized compression error bound for different layers of the KV cache, balancing the tradeoff between generation quality and efficiency. In summary, results show a similar pattern to the synthetic workloads. This means the strong performance MobiLoRA holds for real-world workloads.

4.3 Ablation Study

We conducted a breakdown of the two design modules and performed ablation experiments. As shown in Fig.6, MobiLoRA no-diff-compress means disabling the similarity-aware KV delta en-

coding in §3.2, and MobiLoRA-LRU means to disable the context-aware KV cache management in §3.3. We can see from the figure that our approach outperforms the two variants, especially in scenarios S3 and S5, indicating that with ample memory, our method achieves higher memory utilization, allowing for the storage of more LoRA KV caches.

In extreme memory environments, the proportion of evictions triggered by our method is higher. Therefore, in scenarios S1 and S4, our performance surpasses that of the naive LRU eviction mechanism. The primary reason is that our mechanism can perceive the context of both the application layer and the system layer, enabling joint management of the KV cache under extreme memory conditions.

5 Related Work

Optimize LoRA-based LLM Serving System. There have been many efforts to accelerate the multi-tenant LoRA serving system. dLoRA (Wu et al., 2024a) use advanced dynamical batching mechanisms for efficient serving merged and unmerged inference. Punica (Chen et al., 2024) presents a new CUDA kernel design that allows batching of GPU operations for different LoRA models in a GPU claster. S-LoRA proposes a new tensor parallelism strategy to decoupled the base model and LoRA adapters, and also including unified paging strategy to manage KV caches and adapter weights uniformly. Caraserve (Li et al., 2024) employs a CPU-assisted approach and a rank-aware scheduling algorithm to mitigate the cold-start overhead and meet SLOs respectively. Pets (Zhou et al., 2022) presents a unified framework for serving multiple LoRA adapters efficiently.

KV Cache Management and Compression. KV cache is widely used for accelerating autoregressive decoding nature. Existing work explores various approaches to reducing the storage requirements of KV caches from multiple perspectives. The first one is reducing the generation of KV caches. SGLang and vLLM exploit sharing prefixes to reach that. The second one is reducing the size of KV caches. CacheGen reduces the bandwidth needed to transmit KV caches by compressing them into compact bitstreams. CacheAttention (Gao et al., 2024a) manages KV caches through hierarchical KV cache placement and an overlapping mechanism designed to reduce the overhead associated with this process. Parrot (Lin et al., 2024)

Serving LLM on Device. Mllm (Yi et al., 2023b) proposes to utilize on-device NPU for reducing prefill latency and energy consumption first. EdgeMoE (Yi et al., 2023a), an on-device MoE engine with treats memory as a cache for experts that are held in external storage. LLM in a flash (Alizadeh et al., 2024) leverages the model sparsity to accelerate the on-device LLM inference. PowerInfer (Song et al., 2023) exploits the cold-hot neurons' distribution to design a GPU-CPU hybrid inference engine. LLMCad (Xu et al., 2023) deliver LLM's scaling ability to mobile devices by redesigning speculative generation pipeline. LLMaaS (Cai et al., 2024) proposed a new paradigm of mobile AI which decoupled the memory management of application and LLM contexts.

6 Conclusion

In this paper, we present MobiLoRA, an efficient inference framework for LoRA-based LLMs on mobile devices. MobiLoRA takes advantage of the semantic- and system-level contexts to accelerate the inference. The core of MobiLoRA is a new attention mechanism referred to as `CtxAttention`, which stores the semantic- and system-level contexts for KV cache management optimization.

With `CtxAttention`, MobiLoRA proposes a similarity-aware delta KV encoding to facilitate the efficient storage and reuse of the KV cache for LoRA-based LLMs. Moreover, MobiLoRA leverages the system-level contexts, i.e., the application state of who sends the LLM request, to optimize the KV cache management. Evaluation with real-world mobile usage traces shows the effectiveness of our design. Compared with existing LoRA serving frameworks, MobiLoRA achieves 18.1%~80.5% latency improvement.

7 Limitations

This paper presents an initial trial towards the optimization of the KV cache for LoRA-based LLMs, aiming to facilitate more natural language processing tasks on mobile devices. We recognize that this initial trial has its limitations and risks.

First, although the design of MobiLoRA is not bound to specific foundation models and adapters, our current implementation does not involve more different architectures of foundation models and

other distributions of LoRA adapters. Second, our proof-of-concept implementation focuses on a specific mobile device platform with only GPU acceleration. Instead, commercial-off-the-shelf mobile devices have variant hardware configurations, such as some mobile platforms relying on the domain-specific accelerator such as neural processing unit (NPU). We identify the cooperative inference between multiple accelerators is able to further accelerate the LoRA inference as a promising problem for future exploration.

References

Keivan Alizadeh, Seyed Iman Mirzadeh, Dmitry Belenko, S. Khatamifard, Minsik Cho, Carlo C Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. 2024. LLM in a flash: Efficient Large Language Model Inference with Limited Memory. In *Proc. of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 12562–12584, Bangkok, Thailand.

Android Developers. 2023. Android AICore, a new system service for on-device foundation models. https://android-developers.googleblog.com/2023/12/a-new-foundation-for-ai-on-android.html.

Apple. 2024. Introducing Apple's On-Device and Server Foundation Models. https://machinelearning.apple.com/research/introducing-apple-foundation-models.

Jeff Bilmes. 2022. Submodularity in machine learning and artificial intelligence. *Preprint*, arXiv:2202.00132.

Jeffrey Bilmes and Wenruo Bai. 2017. Deep submodular functions. *Preprint*, arXiv:1701.08939.

Zinuo Cai, Rongbo Ma, Yicheng Fu, Weishan Zhang, Ruhui Ma, and Haibing Guan. 2024. LLMaaS: Serving Large Language Models on Trusted Serverless Computing Platforms. *IEEE Transactions on Artificial Intelligence*, pages 1–11.

Lequn Chen, Zihao Ye, Yongji Wu, Danyang Zhuo, Luis Ceze, and Arvind Krishnamurthy. 2024. Punica: Multi-Tenant LoRA Serving. *Proceedings of MLSys*, 6:1–13.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. 2024a. Cost-Efficient large language model serving for multi-turn conversations with CachedAttention. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 111–126, Santa Clara, CA. USENIX Association.

Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. 2024b. {Cost-Efficient} Large Language Model Serving for Multi-turn Conversations with {CachedAttention}. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 111–126.

Tom Gunter, Zirui Wang, Chong Wang, Ruoming Pang, Andy Narayanan, Aonan Zhang, Bowen Zhang, and et al. 2024. Apple Intelligence Foundation Language Models. *Preprint*, arXiv:2407.21075.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2023. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In *Proc. of ICLR*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. In *Proc. of International Conference on Learning Representations (ICLR)*.

Rui Kong, Qiyang Li, Xinyu Fang, Qingtian Feng, Qingfeng He, Yazhu Dong, Weijun Wang, Yuanchun Li, Linghe Kong, and Yunxin Liu. 2024a. LoRA-Switch: Boosting the Efficiency of Dynamic LLM Adapters via System-Algorithm Co-design. *Preprint*, arXiv:2405.17741.

Rui Kong, Yuanchun Li, Qingtian Feng, Weijun Wang, Xiaozhou Ye, Ye Ouyang, Linghe Kong, and Yunxin Liu. 2024b. SwapMoE: Serving Off-the-shelf MoE-based Large Language Models with Tunable Memory Budget. In *Proc. of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6710–6720, Bangkok, Thailand.

Lilly Kumari, Shengjie Wang, Tianyi Zhou, Nikhil Sarda, Anthony Rowe, and Jeff Bilmes. 2024. BumbleBee: Dynamic KV-Cache Streaming Submodular Summarization for Infinite-Context Transformers. In *Proc. of Conference on Language Modeling (COLM)*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, New York, NY, USA.

Joohyun Lee, Kyunghan Lee, Euijin Jeong, Jaemin Jo, and Ness B Shroff. 2016. Context-aware application scheduling in mobile systems: What will users do and not do next? In *Proc. of ACM UbiComp*, pages 1235–1246.

Suyi Li, Hanfeng Lu, Tianyuan Wu, Minchen Yu, Qizhen Weng, Xusheng Chen, Yizhou Shan, Binhang Yuan, and Wei Wang. 2024. Caraserve: Cpu-assisted and rank-aware lora serving for generative llm inference. *Preprint*, arXiv:2401.11240.

Xin Liang, Sheng Di, Dingwen Tao, Sihuan Li, Shaomeng Li, Hanqi Guo, Zizhong Chen, and Franck Cappello. 2018. Error-controlled lossy compression optimized for high compression ratios of scientific datasets. In *Proc. of IEEE Big Data*.

Chaofan Lin, Zhenhua Han, Chengruidong Zhang, Yuqing Yang, Fan Yang, Chen Chen, and Lili Qiu. 2024. Parrot: Efficient Serving of LLM-based Applications with Semantic Variable. *Preprint*, arXiv:2405.19888.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. 2024. CacheGen: KV Cache Compression and Streaming for Fast Large Language Model Serving. In *Proc. of ACM SIGCOMM*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745.

ShareGPT. 2023. Share Your Wildest ChatGPT Conversations with One Click. `https://sharegpt.com/`.

Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph Gonzalez, and Ion Stoica. 2024. S-LoRA: Scalable Serving of Thousands of LoRA Adapters. In *Proc. of MLSys*, volume 6, pages 296–311.

Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. 2023. PowerInfer: Fast Large Language Model Serving with a Consumer-grade GPU. In *Proc. of ACM SOSP*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Xiong Wang, Linghe Kong, Zucheng Wu, Long Cheng, Chenren Xu, and Guihai Chen. 2020. SLoRa: Towards secure LoRa communications with fine-grained physical layer features. In *Proc. of ACM SenSys*, pages 258–270.

Bingyang Wu, Ruidong Zhu, Zili Zhang, Peng Sun, Xuanzhe Liu, and Xin Jin. 2024a. {dLoRA}: Dynamically Orchestrating Requests and Adapters for {LoRA} {LLM} Serving. In *Proc. of USENIX OSDI*, pages 911–927.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2024b. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversations. In *Proc. of Conference on Language Modeling (COLM)*.

Daliang Xu, Wangsong Yin, Xin Jin, Ying Zhang, Shiyun Wei, Mengwei Xu, and Xuanzhe Liu. 2023. Llmcad: Fast and scalable on-device large language model inference. *Preprint*, arXiv:2309.04255.

Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. 2023a. EdgeMoE: Fast On-Device Inference of MoE-based Large Language Models. *Preprint*, arXiv:2308.14352.

Rongjie Yi, Xiang Li, Zhenyan Lu, Hao Zhang, Daliang Xu, Liming Yang, Weikai Xie, Chenghua Wang, Xuanzhe Liu, and Mengwei Xu. 2023b. mllm: fast and lightweight multimodal llm inference engine for mobile and edge devices.

Donghan Yu, Yong Li, Fengli Xu, Pengyu Zhang, and Vassilis Kostakos. 2018. Smartphone app usage prediction using points of interest. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):174.

Jianwei Zheng, Zhenhua Li, Feng Qian, Wei Liu, Hao Lin, Yunhao Liu, Tianyin Xu, Nan Zhang, Ju Wang, and Cang Zhang. 2024. Rethinking Process Management for Interactive Mobile Systems. In *Proc. of ACM MobiCom*, pages 215–229, New York, NY, USA.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2023. SGLang: Efficient Execution of Structured Language Model Programs.

Zhe Zhou, Xuechao Wei, Jiejing Zhang, and Guangyu Sun. 2022. {PetS}: A Unified Framework for {Parameter-Efficient} Transformers Serving. In *Proc. of USENIX ATC*, pages 489–504.