APLoRA: Efficient Multi-Task Fine-Tuning for Long-Context LLMs

1st Hongbo Chen, 2nd Yi Gong, 3rd Yueming Chen,
4th Ruiming Wen, 5th Zhaokun Wang, 6th Xunlei Chen*, 7th Wenhong Tian*
School of Information and Software Engineering
University of Electronic Science and Technology of China
Chengdu, China
Emails: xunlei@std.uestc.edu.cn, tian_wenhong@uestc.edu.cn

***Abstract*—Current large language models (LLMs) deliver strong task generalization, yet adapting them to long-context scenarios is hindered by the quadratic complexity of self-attention and hardware memory constraints. We target single-device fine-tuning and deployment, where extending context length from 4k to 12–20k tokens is difficult but critical for document-centric applications in resource-limited environments. We first present Aggregated Shifted Sparse (ASS) attention, a grouped sparse mechanism that grants global visibility to the first k sink tokens and leverages shifted local windows to retain pre-trained attention patterns. It lowers attention costs, maintains perplexity comparable to dense attention on long sequences, and enables extrapolation beyond training windows via position interpolation. Based on this, we propose APLoRA, a high-throughput multi-task LoRA fine-tuning framework for long-context LLMs. It integrates NF4 quantization, shared-base-model batch fusion, and an adaptive scheduler with Padding-Aware Sequence Alignment (PASA) to optimize memory, padding overhead, and task turnaround time under fixed hardware budgets. Evaluations on Llama2-7B and Llama3-8B across PG19, ProofPile, LongBench, and an internal long-context corpus demonstrate that our method stably extends context length from 4k to 12–20k tokens on a single Ascend 910B NPU. Compared with dense-attention and sequential LoRA baselines, APLoRA boosts effective token throughput by up to 23.9% and cuts end-to-end task time by 19%, while achieving comparable or better performance in perplexity and downstream accuracy than existing long-context adaptation methods.**

***Index Terms*—Long-Context LLMs, Resource Constrained Environments, Multi Task LoRA Fine-Tuning, Padding-Aware Sequence Alignment (PASA), Training Throughput.**

I. Introduction

LLMs have demonstrated remarkable capabilities in downstream task processing [1]–[4] and content generation [5] through unprecedented model scale expansion [6] and diversification of pretraining data growth [7], [8]. Among these, long-context modeling capability has become a key metric for evaluating the performance of LLMs [9], [10], and is crucial for tasks such as multi-document retrieval, question answering, and document summarization [11]. Despite significant progress, LLMs are fundamentally limited by the quadratic computational complexity of self-attention and fixed context windows [12], leading to inefficiency in processing long sequences [13]. For instance, documents in specialized domains often exceed 15k tokens, and multi-document QA tasks commonly involve contexts over 50k tokens. In practical single-device settings, extending the window from 4k to 12–20k tokens is already challenging. The fixed window hinders LLMs from capturing long-range dependencies [14], resulting in computational bottlenecks and degraded performance in modeling distant relationships.

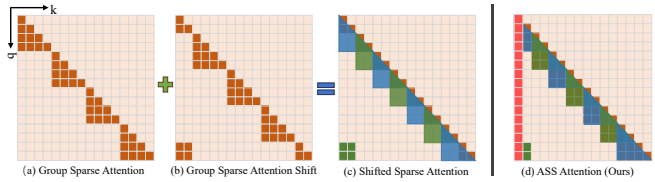

Fig. 1: (a)-(c) show existing attention mechanisms, we proposed ASS Attention (d). This approach grants the first k sink tokens global visibility as attention aggregation points, enabling all subsequent tokens to directly attend to them.

Recent research has explored two directions to mitigate this issue, context extension and parameter-efficient fine-tuning (PEFT) [15]. Context extension includes rotary position embeddings, position interpolation, and sparse attention mechanisms [13], [16], [17], which improve extrapolation but often require costly multi-device retraining. Besides, LoRA [18] and QLoRA [19] significantly reduce resource requirements by introducing low-rank adaptation and quantization. However, these methods still struggle with scalability in resource-constrained environments, where both memory consumption and throughput remain critical challenges [20], [21]. In parallel, several studies have explored frameworks to improve fine-tuning throughput. Systems such as Punica [22] and ASPEN [23] leverage parameter sharing and custom kernels to support multi task LoRA training. However, these approaches often lack integration with diverse frameworks such as MindSpore [24], and cannot fully exploit hardware software co-optimization on Ascend-class NPUs [25].

This study proposes a long-context adaptation strategy that integrates Aggregated Shifted Sparse (ASS) attention with position interpolation to extend the context length of large language models from 4k to 12–20k tokens under fixed hardware budgets. ASS attention reduces the effective computation

* Corresponding Author.

of self-attention while preserving pre-trained attention patterns and maintaining stable performance on long sequences (shows in Fig.1). Building on this, we design APLoRA, a high throughput multi task LoRA framework that employs parameter-shared batch fusion and adaptive scheduling to minimize padding overhead and improve utilization on a single device. This study aims to provide a unified and practical solution for scalable, high throughput long-context fine-tuning of large language models in resource constrained environments.

In summary, our work makes the following contributions:

(I) We propose a long-context extension strategy based on ASS attention. This approach reduces computational complexity by maintaining global visibility for the first k tokens, while enhancing the model's understanding and generation capabilities in long-context scenarios.

(II) We design the APLoRA framework, which incorporates an adaptive scheduling algorithm with PASA to reduce computational and padding overhead effectively, and enables efficient multi-task parallel training with dynamic resource allocation on shared hardware.

(III) Experiments on multiple models and datasets with Ascend 910B show that our approach achieves competitive and SOTA long-context performance with significantly lower computational and memory costs.

II. Related Work

The core challenges of long-context LLMs lie in breaking the fixed context window limit and reducing fine-tuning resource consumption [26]–[28]. Which mainly focuses on context length extension, PEFT, and high throughput fine-tuning frameworks [29]. In terms of context length extension, technical routes are divided into extrapolation and interpolation. StreamingLLM and attention-sink methods achieve zero-shot or fine-tuning based sequence length expansion [30], [31], but they primarily target decoding-time efficiency rather than training-time sparse attention design or single-device long-context fine-tuning. [32], [33]. Interpolation techniques, represented by Position Interpolation (PI), adapt to the pre-trained window by scaling positional indices, which is simple to implement but requires fine-tuning to adapt to new position distributions [17], [34]. LongLoRA combines shifted sparse attention with LoRA to extend context, but it relies on multi-GPU resources and has limited ability to capture long-range dependencies [35], [36].

In the field of PEFT, LoRA injects low-rank matrices for efficient fine-tuning [37]. Subsequent improvements such as QLoRA introduce 4-bit quantization to further reduce memory overhead [19], while DyLoRA and AdaLoRA dynamically adjust rank values to optimize performance [38], [39]. However, these methods still face bottlenecks of memory redundancy and computational inefficiency in long sequence scenarios. Regarding quantization techniques, NF4 fractional quantization is optimized for the normal distribution characteristics of neural network weights, resulting in smaller precision loss compared to uniform quantization schemes such as INT4 [19]. Double Quantization (DQ) can further compress the memory footprint of quantization constants, making it attractive for single device long-context fine-tuning [40].

High throughput fine-tuning frameworks focus on multi-task parallel optimization. Punica improves GPU parallel efficiency through custom CUDA kernels [**?**], and ASPEN enables multi-task training on a single GPU via model sharing and adaptive scheduling [23]. However, both lack native support for heterogeneous training frameworks such as MindSpore and NPU backends, and do not fully integrate padding-aware batching and early-stopping strategies, leaving room for improvement in end-to-end resource utilization and task throughput.

III. APLoRA: High throughput Framework

This section presents ASS attention and APLoRA, a high-throughput fine-tuning framework whose core component is the PASA scheduler for batch selection. While ASS attention can be applied to the various self-attention modules [41], we focus on its integration into a practical, multi-task LoRA fine-tuning pipeline for long-context LLMs under single-device, resource-constrained settings[1].

A. Preprocessing and ASS Attention

We consider a set of multi-task fine-tuning requests $Q_{\text{job}} = \{J_1, J_2, \ldots, J_n\}$. Each task J_i is specified by its LoRA rank r_i, learning rate η_i, batch size $B_{t,i}$, and priority p_i, which serve as inputs to the subsequent scheduling and resource allocation modules.

NF4 quantization. We apply NormalFloat-4 (NF4) quantization with double quantization to compress the pre-trained weights W_0, reducing memory usage while keeping the dequantization overhead small:

$$\begin{aligned} W_0^{\text{NF4}} &= \text{QuantNF4}(W_0^{\text{BF16}}, c_1^{\text{FP32}}, c_2^{\text{FP8}}), \\ W_0^{\text{BF16}} &= \text{DoubleDeQuant}(c_1^{\text{FP32}}, c_2^{\text{FP8}}, W_0^{\text{NF4}}), \end{aligned} \tag{1}$$

where c_1^{FP32} and c_2^{FP8} are quantization constants. At run time, W_0^{NF4} is dequantized on the fly to BF16 for compute, and remains frozen during LoRA fine-tuning.

Position interpolation for long-contexts. For a task J_i with input sequence length $L_{1,i} > L_0$ (the pre-trained context window), we apply position interpolation to keep the position distribution consistent with the pre-training regime:

$$f'(x_i, n_i) = f\left(x_i,\ \left(n_i \cdot \frac{L_0}{L_{1,i}}\right) \Big/ \left\lceil \frac{L_0}{L_{1,i}} \right\rceil\right), \tag{2}$$

where $n_i \in [0, L_{1,i})$ is the original positional index, $f(\cdot)$ is the RoPE encoding function, and $f'(\cdot)$ is the interpolated encoding. This interpolation avoids aggressive extrapolation while enabling contexts up to 12–20k tokens.

Aggregated Shifted Sparse (ASS) attention. Standard self-attention suffers from $O(L^2)$ complexity in both time and memory. To reduce its effective cost under fixed hardware budgets, we replace the dense attention layer with ASS attention, a grouped sparse pattern that combines local windows with a small set of globally visible "sink" tokens, inspired by

[1] APLoRA implements on MindSpore/Ascend, but it is framework-agnostic.

attention-sink ideas such as StreamingLLM but tailored to our long-context, single-device setting.

Concretely, the input sequence of length L is partitioned into G groups of equal size $M = L/G$. Each token attends densely within its group, yielding an intra-group complexity of $O(L^2/G)$. To enable cross-group information flow, we introduce a cyclic shift between adjacent groups with offset $M/2$, so that each token can also attend to a subset of tokens from neighboring groups. Finally, we designate the first k tokens ($k = M/2$ in our implementation) as global sink tokens, which remain visible to all positions and act as aggregation points for long-range dependencies.

This design preserves the dominant attention patterns learned during pre-training, through the global sink tokens while sparsifying long-range connections.

B. PASA for Multi-Task Parallel Training

To enable efficient multi-task LoRA fine-tuning, APLoRA performs batch fusion over multiple tasks while preserving the accuracy of the NF4-quantized pre-trained weights. At each scheduling step, a set of tasks is selected, and the shared W_0^{NF4} is dynamically dequantized to W_0^{BF16} for the forward and backward passes. For each selected task J_i, input sequences x_i with length $L_{1,i} > L_0$ are first processed by position interpolation (Eq. 2). Then, PASA pads and fuses sequences $\{x_1, \ldots, x_k\}$ into a single batch of maximum length $L_{\max}$:

$$X_f = \text{Fusion}(x_1, \ldots, x_k) \in \mathbb{R}^{k \times L_{\max} \times d}, \tag{3}$$

where k is the number of parallel tasks in the fused batch, d is the hidden dimension, and $L_{\max}$ is chosen adaptively by PASA based on the candidate sequence lengths (Section III-C).

Forward and backward with shared base model. During the forward pass, we reuse the shared base model W_0^{BF16} for all tasks and apply task-specific LoRA adapters (A_i, B_i) with scaling factor s_i:

$$H = W_0^{\text{BF16}} X_f + \text{Concat}\big(s_i x_i B_i A_i\big)_{i=1}^{k}, \tag{4}$$

where $H \in \mathbb{R}^{k \times L_{\max} \times d}$ is the fused hidden representation. In the backward pass, we freeze W_0^{NF4} and only update the LoRA parameters using the cross-entropy loss $\mathcal{L}$:

$$\frac{\partial \mathcal{L}}{\partial B_i} = \frac{\partial \mathcal{L}}{\partial H}(X_f^T A_i^T s_i), \quad \frac{\partial \mathcal{L}}{\partial A_i} = \frac{\partial \mathcal{L}}{\partial H}(s_i B_i^T X_f^T). \tag{5}$$

Padding-Aware Sequence Alignment (PASA). To quantify padding overhead, we define the padding rate $\delta = \xi_p/\xi$, where ξ_p is the number of padded tokens and ξ is the total number of tokens in the fused batch. The effective token throughput is $T_e = (1-\delta)\,(\sum \text{token})/t$, where t is the wall-clock time of a training step. PASA aims to minimize δ by selecting and grouping tasks with similar sequence lengths. Given a set of candidate sequence lengths $\{a_1, \ldots, a_n\}$ (decreasing order) and a candidate window size m, the total padding required to fuse the m longest sequences into a batch of length a_n is:

$$\text{pad}_m = m a_n - (S_n - S_{n-m}), \tag{6}$$

where S_k is the prefix sum of $\{a_i\}$. PASA evaluates different values of m within the scheduler's top-k candidate set and selects the subset with the smallest pad_m for fusion, subject to memory constraint. This heuristic substantially reduces redundant padding and improves the throughput.

C. Early Stopping and Adaptive Scheduling

To avoid out-of-memory (OOM) errors under fixed hardware budgets, APLoRA first predicts the memory usage of each task and then performs memory-constrained scheduling. For a task with batch size B_t and sequence length L_n, we use a simple quadratic regression model:

$$\begin{cases} OM = \beta_0 + \beta_1 B_t L_n + \beta_2 B_t L_n^2, \\ \displaystyle\sum_{J_i \in X} OM_i \le M_{\text{mem}}, \end{cases} \tag{7}$$

where OM denotes the estimated memory consumption (GB), β_0 is the base memory, and β_1, β_2 capture activation and attention-related growth. X is the set of tasks scheduled in the same batch, and M_{mem} is the device memory limit. The coefficients $\beta_0, \beta_1, \beta_2$ are fitted once from a small number of profiling runs and then refined online using memory usage.

Early stopping prediction. To avoid wasting resources on stalled tasks, we apply an early-stopping rule based on smoothed loss changes:

$$\frac{1}{w}\sum_{t=0}^{w-1} |\mathcal{L}_{\tau+t} - \mathcal{L}_{\tau+t+1}| < \epsilon, \tag{8}$$

where w is the observation window, $\mathcal{L}_t$ is the validation loss at step t, and ϵ is a small threshold. In addition, a task is terminated if $\text{Acc}_t < 0.95 \times \text{Acc}_{\max}$ for a fixed number of evaluations or if the loss diverges (NaN).

Scheduling policy. Given the Q_{job}, M_{mem}, and a scheduler top$k$, we proceed as follows. Tasks are first sorted by priority p_i and submission time. The top-k candidates C_{job} are then selected, and for each $J_i \in C_{\text{job}}$ we estimate its memory demand OM_i and expected early-stopping iteration $T_{\text{stop},i}$, based on recent learning (Eq. 8).

Next, we rerank C_{job} by $T_{\text{stop},i}$ and pad_m obtained from PASA, and incrementally construct the scheduled set R such that $\text{Mem}_{\text{used}} + OM_i \le M_{\text{mem}}$ for all $J_i \in R$. The selected set R is then submitted to the batch-fusion module for joint training. During training, we continuously log actual memory usage OM_{real}, loss $\mathcal{L}$, and effective throughput T_e to refine the memory model parameters $(\beta_0, \beta_1, \beta_2)$ and the early-stopping threshold ϵ. Asynchronous task submissions trigger re-scheduling events that add new jobs into the queue and repeat the above procedure. Upon meeting the early-stopping criterion or reaching the maximum iteration budget, the optimized LoRA matrices A_i^*, B_i^* (BF16) for task J_i are saved. The corresponding fine-tuned weight is obtained by:

$$W_{\text{final},i} = W_0^{\text{BF16}} + s_i B_i^* A_i^*. \tag{9}$$

After serialization, memory associated with A_i^* and B_i^* is released and is made available for subsequent tasks in Q_{job}.

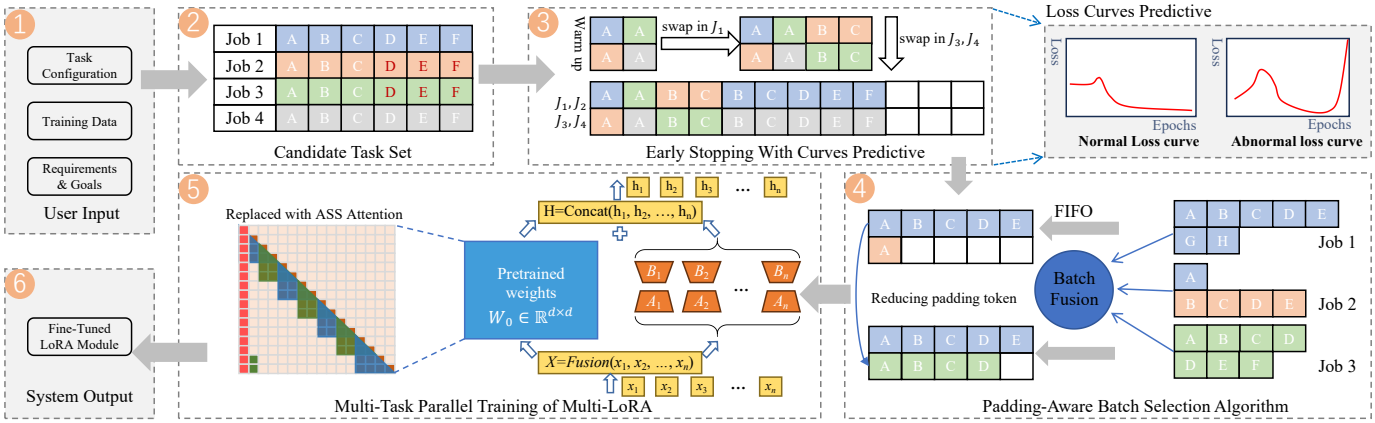

Fig. 2: Overview of the APLoRA framework, which combines the pre-trained model sharing mechanism and adaptive scheduling strategy to schedule computing resources for multiple LoRA tasks dynamically. It improves fine-tuning throughput and resource utilization while maintaining the performance of the fine-tuned model.

Dataset	**PG19**			**ProofPile**			
Model / Context length	8192	16384	20480	8192	16384	20480	Average
Llama2-7B	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$
Llama2-7B + PI	8.30	9.50	10.80	3.05	3.45	3.90	6.50
MPT-7B-8K	7.66	7.79	7.95	2.67	2.73	2.79	**5.27**
LongLoRA-7B-8K	7.90	8.15	8.43	2.75	2.84	2.95	5.50
APLoRA-7B-8K	7.81	7.96	8.18	2.72	2.78	2.87	5.39
Model / Task	Single-Doc. QA	Multi-Doc. QA	Abs. Gen.	Few-Shot Learning	Code Comple.	Compre. Tasks	Average
Llama2-7B-chat	24.9	22.6	24.7	60.0	48.1	5.9	31.0
LongChat-v1.5-7B	28.7	20.6	26.7	60.0	54.1	15.8	34.3
Vicuna-v1.5-7B	28.0	18.6	26.0	**66.2**	47.3	5.5	31.9
LongLoRA-7B-12K	28.7	28.1	27.8	63.7	56.0	16.7	36.8
Llama-3.1-8B-32K	**38.5**	**36.0**	30.5	65.0	58.5	**30.0**	**43.3**
APLoRA-7B-12K	35.9	32.7	**31.1**	65.4	**59.2**	22.3	38.8

TABLE I: Comprehensive evaluation of long-context language modeling and downstream tasks. **Top:** Perplexity (PPL, lower is better) on PG19 and ProofPile at context lengths 8K, 16K, and 20K. "Llama2-7B + PI" denotes the base model with position interpolation but *without* long-context fine-tuning. **Bottom:** Accuracy on six LongBench sub-tasks, including Single-Document QA, Multi-Document QA, Abstractive Generation, Few-Shot Learning, Code Completion, and Comprehensive Tasks, after supervised fine-tuning Llama2-7B to a 12,288-token context length on the LongML dataset to obtain APLoRA-7B-12K. We additionally report an open-source Llama-3.1-8B-32K as a frontier long-context baseline.

IV. Experiment

At the algorithmic level, we validate the effectiveness of APLoRA through long sequence language modeling and long-context benchmarking experiments. Moreover, we evaluate the throughput performance of the MAPLoRA framework. The results demonstrate the advantages of our approach in extending the context length of large language models.

A. Experimental Setting

Model and Dataset The APLoRA-Llama2-7B-8K model is obtained by fine-tuning 1000 steps on the RedPajama dataset [42] and evaluated on PG19 [43] and ProofPile [44]. We compare with Llama2-7B [45], Llama3-8B [46], MPT-7B [47], LongLoRA-7B [35], and an open-source 32K-context model. For long-context benchmarks, we train APLoRA-Llama2-7B-12K with ASS attention and position interpolation on our in-house LongML corpus (technical, legal, and scientific documents, 4K–20K tokens) and evaluate on LongBench [48], as well as perplexity and key-retrieval at 16K/20K against base models using only position interpolation.

Configuration We use the AdamW optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.95$) with a learning rate of 2×10^{-4}, linear warm-up over 20 steps, and no weight decay. Training is performed with a batch size of 1 and gradient accumulation over 16 steps. During inference, model weights are quantized to 4-bit precision for sequences longer than 12,288 tokens; otherwise, float16 precision is used. All experiments are conducted on an Ascend NPU 910B with 64GB HBM. We report effective token throughput (useful tokens per second) and padding ratio.

B. Results and Discussion

Main Results. Table I and Table II together show that APLoRA-7B-8K achieves stable and robust long-context language modeling. Compared with the base Llama2-7B equipped only with position interpolation, APLoRA-7B-8K not only attains lower perplexity within the 8K training window, but also exhibits much milder degradation when extrapolating to 16K and 20K on both PG19 and ProofPile. This indicates that our fine-tuning procedure learns genuinely length-generalizable representations rather than merely overfitting to the training window, while retaining competitive performance

Model	PG19 PPL			ProofPile PPL		
	8K	16K	20K	8K	16K	20K
Llama2-7B + PI	8.30	9.50	10.80	3.05	3.45	3.90
APLoRA-7B-8K	7.81	7.96	8.18	2.72	2.78	2.87
%Δ (16K vs 8K)	–	+2.0	+4.7	–	+2.2	+5.5
%Gain vs PI @20K	–	24.3	24.3	–	26.4	26.4

TABLE II: Perplexity extrapolation beyond the 8K training window. APLoRA-7B-8K shows only a small degradation when moving from 8K to 16K/20K, whereas Llama2-7B + PI suffers a much larger increase in PPL.

Attention Mechanism	Method		Target Context Length		
	Shift	Convergence	4096	8192	16384
Dense	-	-	7.99	8.00	8.00
s^2	✓	×	8.09	8.07	8.08
ASS	✓	✓	8.03	8.03	8.02

TABLE III: Effectiveness at Different Target Context Lengths. Models are trained on RedPajama to different tokens without quantization.

against strong long-context baselines such as MPT-7B-8K and LongLoRA-7B-8K, all under a single Ascend 910B budget with only 1,000 update steps.

On downstream evaluation, APLoRA-7B-12K consistently improves over Llama2-based chat models and LongLoRA-7B-12K across all LongBench sub-tasks in Table I, and significantly narrows the gap to a larger open-source 32K-context model (Llama-3.1-8B-32K). In particular, APLoRA-7B-12K delivers strong gains on abstractive generation and code-related tasks, suggesting better utilization of long-range evidence and richer long-context reasoning. Given that these improvements are obtained with a single-device fine-tuning setup (28 hours, 300M tokens), APLoRA offers a practical and efficient solution for deploying domain-specific long-context models in real-world scenarios such as legal and industrial document understanding.

Ablation Study on Attention Mechanisms. In Table III, we compare the PPL of dense attention, s^2-attention, and ASS attention at different target context lengths. Results show that dense attention maintains nearly constant perplexity but incurs high computational cost, while s^2-attention reduces complexity at the expense of slightly higher perplexity. In contrast, ASS attention closely matches the perplexity of dense attention and remains stable as the target context length increases, while retaining lower computational complexity, demonstrating its effectiveness for long-context modeling and suitability for practical deployment.

Ablation on LoRA Rank and Quantization Precision. Table IV shows that PPL generally decreases as the LoRA rank increases, and when the rank reaches 32 or 64, APLoRA matches its best PPL and remains competitive with LongLoRA while approaching MPT-7B-8K, indicating that higher ranks enable richer feature learning at little extra cost. The quantization ablation further reveals that NF4 and NF4+DQ incur almost no degradation compared to the BF16 baseline, while INT4 leads to a noticeably larger loss, demonstrating that NF4-based quantization can preserve APLoRA's long-context modeling capability nearly losslessly at extremely low bit-width and with additional memory savings from dual quantization.

LoRA Rank Ablation		Quantification Performance		
APLoRA(rank)	PPL	QS	PPL	Loss
8	7.91	BF16(Baseline)	7.80	–
16	7.83	INT4	7.99	4.4%
32	7.81	Float4(E2M1)	7.93	1.7%
64	7.81	Float4(E3M0)	7.88	1.0%
LongLoRA	7.90	NF4	7.81	0.1%
MPT-7B-8K	7.66	NF4+DQ	7.81	0.1%

TABLE IV: The Influence of relative performance degradation of different LoRA ranks in fine-tuning Llama2-7B on the PG19 dataset. And the modeling performance of APLoRA-7B-8K/12K under various quantization strategies (QS) compared to the BF16 baseline.

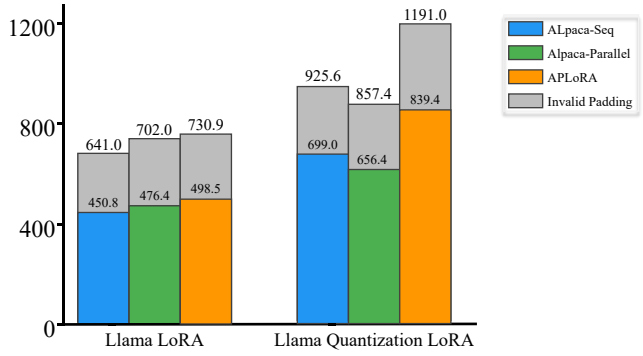

Fig. 3: Throughput evaluation. We use a Multi Task setup with four independently trained LoRA models (batch size $B = 4$) to measure effective vs. padded token throughput. Alpaca-LoRA (single-task baseline), Alpaca-Seq (sequential execution), and Alpaca-Parallel (concurrent via model copies) serve as baselines.

Framework Performance Evaluation. As shown in Figure 3, APLoRA attains the highest effective token throughput among all baselines in both the Llama LoRA and Llama Quantization LoRA settings. By fusing batches and scheduling multiple independently trained LoRA models within a single backbone, APLoRA reduces kernel launch and padding overhead, substantially narrowing the gap between padded and effective throughput. Compared to Alpaca-Seq and Alpaca-Parallel, it consistently improves effective throughput while preserving model accuracy, demonstrating its efficiency for Multi Task workloads under shared-resource deployment.

V. CONCLUSION

This paper proposes a novel high throughput fine-tuning framework for long-context LLMs. APLoRA integrates ASS attention with a padding-aware scheduling strategy to reduce quadratic complexity while preserving pre-trained patterns. The framework enables efficient Multi Task LoRA training with dynamic resource allocation, achieving significant gains in throughput and memory efficiency. Extensive experiments demonstrate that APLoRA achieves state-of-the-art performance across long-context benchmarks while substantially lowering computational cost. Overall, APLoRA provides a scalable and practical solution for long-context.

ACKNOWLEDGMENT

This research is supported by Chengdu Science and Technology Bureau Project (ID 2024-YF09-00041-SN) and Huawei Funding (ID H04W241592; partially supported by UESTC Kunpeng & Ascend Center of Cultivation).

REFERENCES

[1] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and et al., "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[2] X. Chen, J. Guo, Y. Li, and et al., "Alter: Asymmetric lora for token-entropy-guided unlearning of llms," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 40, no. 42, 2026, pp. 35 366–35 374.

[3] Q. Gong, X. Yang, and X. Tang, "Orthogonal soft pruning for efficient class unlearning," *arXiv preprint arXiv:2506.19891*, 2025.

[4] Z. Wang, J. Guo, J. Pu, and et al., "Noise-robustness through noise: A framework combining asymmetric lora with poisoning moe," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[5] S. Sudhakaran, M. González-Duque, M. Freiberger, C. Glanois, E. Najarro, and S. Risi, "Mariogpt: Open-ended text2level generation through large language models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 54 213–54 227, 2023.

[6] J. Achiam, S. Adler, S. Agarwal, and et al., "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[7] W. X. Zhao, K. Zhou, J. Li, and et al., "A survey of large language models," *arXiv preprint arXiv:2303.18223*, vol. 1, no. 2, 2023.

[8] S. Smith, M. Patwary, B. Norick, P. LeGresley *et al.*, "Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model," *arXiv preprint arXiv:2201.11990*, 2022.

[9] X. Wang, W. Zhu, M. Saxon, M. Steyvers, and W. Y. Wang, "Large language models are latent variable models: Explaining and finding good demonstrations for in-context learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 15 614–15 638, 2023.

[10] J. Ou, J. Guo, S. Jiang, and et al., "Accelerating long-context inference of large language models via dynamic attention load balancing," *Knowledge-Based Systems*, p. 115018, 2025.

[11] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, and et al., "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[12] F. D. Keles, P. M. Wijewardena, and C. Hegde, "On the computational complexity of self-attention," in *International conference on algorithmic learning theory*. PMLR, 2023, pp. 597–619.

[13] T. Dao, "Flashattention-2: Faster attention with better parallelism and work partitioning," *arXiv preprint arXiv:2307.08691*, 2023.

[14] S. An, Z. Ma, Z. Lin, N. Zheng, J.-G. Lou, and W. Chen, "Make your llm fully utilize the context," *Advances in Neural Information Processing Systems*, vol. 37, pp. 62 160–62 188, 2024.

[15] H. Liu, D. Tam, M. Muqeeth, and et al., "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1950–1965, 2022.

[16] J. Su, M. Ahmed, Y. Lu *et al.*, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127063, 2024.

[17] S. Chen, S. Wong, L. Chen, and Y. Tian, "Extending context window of large language models via positional interpolation," *arXiv preprint arXiv:2306.15595*, 2023.

[18] E. J. Hu, Y. Shen, P. Wallis *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022.

[19] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," *Advances in neural information processing systems*, vol. 36, pp. 10 088–10 115, 2023.

[20] P. Gkotsiopoulos, D. Zorbas, and C. Douligeris, "Performance determinants in lora networks: A literature review," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1721–1758, 2021.

[21] L. Zhang, L. Zhang, S. Shi, X. Chu, and B. Li, "Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning," *arXiv preprint arXiv:2308.03303*, 2023.

[22] L. Chen, Z. Ye, Y. Wu, D. Zhuo, L. Ceze, and A. Krishnamurthy, "Punica: Multi-tenant lora serving," *Proceedings of Machine Learning and Systems*, vol. 6, pp. 1–13, 2024.

[23] Z. Ye, D. Li, J. Tian *et al.*, "Aspen: High-throughput lora fine-tuning of large language models with a single gpu," *CoRR*, 2023.

[24] Z. Tong, N. Du, X. Song, and X. Wang, "Study on mindspore deep learning framework," in *2021 17th International Conference on Computational Intelligence and Security (CIS)*. IEEE, 2021, pp. 183–186.

[25] J. Ou, J. Guo, S. Jiang, and et al., "Low-rank decomposition assisted quantization and inference compensation for quality large language model inference," in *2025 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2025, pp. 1–9.

[26] B. Peng, J. Quesnelle, H. Fan, and E. Shippole, "Yarn: Efficient context window extension of large language models," *arXiv preprint arXiv:2309.00071*, 2023.

[27] Y. Mao, J. Li, F. Meng, and et al., "Lift: Improving long context understanding through long input fine-tuning," *arXiv preprint arXiv:2412.13626*, 2024.

[28] C. An, J. Zhang, M. Zhong, and et al., "Why does the effective context length of llms fall short?" *arXiv preprint arXiv:2410.18745*, 2024.

[29] S. Yu, X. Xu, K. Deng, L. Li, and L. Tian, "Tree of agents: Improving long-context capabilities of large language models through multi-perspective reasoning," *arXiv preprint arXiv:2509.06436*, 2025.

[30] J. Xiao, Y. Chen, Y. Ou, and et al., "Baichuan2-sum: Instruction finetune baichuan2-7b model for dialogue summarization," in *2024 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2024, pp. 1–8.

[31] Z. Liao, J. Wang, H. Yu, and et al., "E2llm: Encoder elongated large language models for long-context understanding and reasoning," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025, pp. 19 212–19 241.

[32] J. Ding, S. Ma, L. Dong, and et al., "Longnet: Scaling transformers to 1,000,000,000 tokens," *arXiv preprint arXiv:2307.02486*, 2023.

[33] H. Naveed, A. U. Khan, S. Qiu, and et al., "A comprehensive overview of large language models," *ACM Transactions on Intelligent Systems and Technology*, 2023.

[34] Y. Wu, Y. Gu, X. Feng, and et al., "Extending context window of large language models from a distributional perspective," *arXiv preprint arXiv:2410.01490*, 2024.

[35] Y. Chen, S. Qian, H. Tang, and et al., "Longlora: Efficient fine-tuning of long-context large language models," *arXiv preprint arXiv:2309.12307*, 2023.

[36] Z. Han, C. Gao, J. Liu, and et al., "Parameter-efficient fine-tuning for large models: A comprehensive survey," *arXiv preprint arXiv:2403.14608*, 2024.

[37] E. J. Hu, Y. Shen, P. Wallis, and et al., "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022.

[38] M. Valipour, M. Rezagholizadeh, I. Kobyzev, and A. Ghodsi, "Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation," in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, 2023, pp. 3274–3287.

[39] Q. Zhang, M. Chen, A. Bukharin, and et al., "Adalora: Adaptive budget allocation for parameter-efficient fine-tuning," *arXiv preprint arXiv:2303.10512*, 2023.

[40] D. Orr, L. Ribar, and C. Luschi, "Optimal formats for weight quantisation," *arXiv preprint arXiv:2505.12988*, 2025.

[41] B. Lin, C. Zhang, T. Peng, H. Zhao, W. Xiao, M. Sun, and et al., "Infinite-llm: Efficient llm service for long context with distattention and distributed kvcache," *arXiv preprint arXiv:2401.02669*, 2024.

[42] M. Weber, D. Fu, Q. Anthony, and et al., "Redpajama: an open dataset for training large language models," *Advances in neural information processing systems*, vol. 37, pp. 116 462–116 492, 2024.

[43] J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap, "Compressive transformers for long-range sequence modelling," *arXiv preprint arXiv:1911.05507*, 2019.

[44] Z. Azerbayev, E. Ayers, and B. Piotrowski, "Proof-pile," https://github.com/zhangir-azerbayev/proof-pile, 2022.

[45] H. Touvron, L. Martin, K. Stone, and et al., "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[46] A. Grattafiori, A. Dubey *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[47] M. Team *et al.*, "Introducing mpt-7b: A new standard for open-source, commercially usable llms, 2023," *URL www. mosaicml. com/blog/mpt-7b. Accessed*, pp. 05–05, 2023.

[48] Y. Bai, X. Lv, J. Zhang, and et al., "Longbench: A bilingual, multitask benchmark for long context understanding," *arXiv preprint arXiv:2308.14508*, 2023.