Author Feedback

Dear Reviewer,

We appreciate the time you have dedicated to reviewing our paper. According to your suggestions, we have carefully revised the paper, including more precise language, a more comprehensive explanation of the methods, and a more detailed analysis of the experimental results, leading to an overall improvement in the paper. We hope your concerns can be addressed point-to-point as follows:

Q1 The deployment on edge devices.

According to your suggestion, we deployed EfficientLLM on non-GPU devices. We deploys using 1, 2, 4, and 8 Intel Xeon @ 2.90GHz CPUs respectively. Compared with SoTA edge models, EfficientLLM achieves both higher inference speed (ms/token) and average zero-shot accuracy.

Table 1. Inference latency (in milliseconds) and accuracy on different models with varying numbers of CPU cores.

Model	1 CPU	2 CPUs	4 CPUs	8 CPUs	Acc. (%)
MobileLLM-350M	132.37	81.26	53.99	41.95	51.30
Qwen2.5-0.5B	54.51	30.08	19.87	13.54	53.20
EfficientLLM-457M	15.93	9.38	5.67	4.07	57.35
OLMo-1B	352.92	186.39	101.81	65.12	55.66
Llama3.2-1B	51.45	30.17	16.59	10.33	56.69
EfficientLLM-1B	49.77	25.52	16.00	10.32	60.75

Q2 The scalability on large sized models (more than 7B parameters) has not been shown in the evaluation.

This work focuses on the pretraining of edge language models, which aims to train smaller-scale language models. Prior works in this field include MobileLLM (100/300M, ICML 2024), PanGu-Pi-Pro (1.5B, ICML 2024), and ShearedLlama (1.3B/2.7B, ICLR 2024). Given that edge devices typically have 8–16GB of DRAM, language models larger than 7B fall outside the primary focus of this research area. Therefore, we conduct comparative experiments on models ranging from 100M to 1B, as well as 1.3B and 2.7B. Additionally, to explore the potential for scaling to larger models, we prune Llama2-13B using the same pruning metric as LLM-Pruner.

Table 2. Performance comparison in Llama2-13B.

Model	Ratio	ARC-C	ARC-E	BoolQ	HS	OBQA	PIQA	WG	Avg.
LLM-Pruner	30%	26.79	26.30	37.83	25.72	24.00	52.18	48.46	34.47
EfficientLLM	30%	32.25	62.96	60.40	59.37	36.80	71.33	60.93	54.86
LLM-Pruner	50%	28.33	26.64	38.53	26.42	27.20	52.61	47.51	35.18
EfficientLLM	50%	31.06	62.08	59.76	56.29	37.20	71.44	60.46	54.04

Q3 It is uncler how the 2nd-order hessian translate to practical computational gains.

As mentioned in lines 259-274 (left) and lines 222-241 (right), we employed the second-order Hessian matrix in two instances, both of which are negligible for the following reasons:

1.1) During the saliency detection phase, we use for second-order saliency calculation. Similar to the approach in LLM-Pruner/Wanda, we only consider the diagonal of the Hessian matrix. Therefore, the additional computation is simply element-wise and derived from the square of the gradients. Compared to a linear layer, this results in minimal increase in overall computation time.

1.2) We only perform saliency calculations on the output layer group, which further reduces the computation by more than 2 times.

2) In second-order parameter updating, we use a method similar to SparseGPT/OBC, where the additional computation required is $H = XX^T$. However, the Hessian matrix is computed once every N updates, with N=9 in our experiments. Thus, compared to calculating a single linear layer, the average increase in computation per step is only 1/9 of the total.

Based on these methods, feel free to apply second-order Hessian metrics in large-scale training.

Q4 While the method claims to be architecture agnostic, most of the architecture evaluated on follow similar architectures.

Thank you for the advice, where ”architecture agnostic” refers to ”pruning target agnostic”. We propose an automatic design of model architecture through pruning during the pretraining phase. Given T pruning iterations, the search space for the entire model architecture is 3^T. This is the first time we have implemented automatic design for modern pretrained models, which is why we refer to it as ”architecture agnostic” (which can be understood as network architecture search). We will express this more precisely in the next version.

Q5 In Figure 4, what is the setup, how many prompts are evaluated, over how many rounds and users/judges?

As cited in the alpaca eval, we apply the standard alpaca-eval pipeline: including the 805 prompts, the GPT4-o judgements as the evaluator for one round. And we evaluate the win rates as the final results.

Q6 The overall cost of the proposed technique during pretraining has not been measured.

Table 3. Comparison of pre-training and pruning costs in GPU hours. PT indicates pretraining.

Model	PT Tokens	GPU Hours	Pruning	GPU Hours	Continued PT	GPU Hours	Total Hours	Acc. (%)
Qwen2.5-0.5B	17T	199467	–	–	–	–	199467	53.20
EfficientLLM-457M	–	–	72.1B	2166	50B	1018	3184	54.77
EfficientLLM-457M	–	–	72.1B	2166	500B	10178	12344	57.35

Thank you for the suggestion. We evaluate pretraining speed under the same environment, as shown in the table. With pruning-aware pretraining, training EfficientLLM-457M requires x16 to x62 times fewer GPU hours compared to Qwen2.5-0.5B, while achieving higher accuracy.

Q7 The evaluation lacks detailed ablation studies on saliency metrics.

Pruning-Aware Pretraining is a general method to scale up previous pruning metrics. As mentioned in line 382(left) 370(right), we evaluate different metrics including LLM-Pruner, SparseGPT/OBC. We also add more ablation studies as follows:

Table 4. Ablation studies on LLM-Pruner, SparseGPT, and Wanda metrics.

Model	#Ratio	Calibration	Pretrain	Δ Acc.
LLM-Pruner	10%	56.63%	58.12%	1.49%
	36%	38.71%	55.37%	16.66%
	51%	35.20%	52.98%	17.78%
	74%	35.51%	48.19%	12.68%
OBC (SparseGPT)	10%	57.04%	58.21%	1.17%
	36%	40.24%	55.68%	15.44%
	51%	36.68%	53.00%	16.32%
	74%	35.53%	48.95%	13.42%
OBD (Wanda)	10%	57.07%	58.69%	1.62%
	36%	37.04%	55.63%	18.59%
	51%	35.04%	52.91%	17.87%
	74%	35.12%	48.35%	13.23%

As shown in the Table, the general improvements are observed in most popular LLM pruning metrics. We finally select LLM-Pruner to scale up because it is more general in this field.

Q8 How would this technique behave on reasoning models?

Before our submission (Jan. 30th), apart from DeepSeek-R1 (Jan. 22nd), there were not many reliable open-source reasoning models. Therefore, we discuss this as the future work. We note that DeepSeek-R1 suggests that for models smaller than 32B, direct reinforcement learning performs worse than distillation from larger models. This aligns with our conclusion: we find that pruning larger models enables better pretraining of small models for edge deployment. Combining distillation with pruning-aware pretraining to train reasoning models is a promising direction for further exploration.

Q9 How does the proposed method work with quantized networks? Could you further prune based on data from the downstream task?

Thank you for your suggestion. Further compression through pruning on downstream tasks or quantization is entirely feasible. Since bit-width still contains redundancy, we applied OmniQuant to perform 8-bit weight and activation quantization (8W8A) on EfficientLLM. The experimental results show that 8-bit quantization only causes minor disturbances to the results, further proving the effectiveness of the model acceleration.

Table 5. Performance comparison of models with different bit-widths.

Model	#Bits	ARC-C	ARC-E	BoolQ	HS	OBQA	PIQA	WG	Avg.
Qwen2.5-0.5B	bf16	32.17	64.44	61.99	52.09	35.20	70.29	56.20	53.20
EfficientLLM-A-457M	bf16	**38.40**	72.10	62.42	56.84	**40.40**	**73.83**	57.46	57.35
EfficientLLM-A-457M-Quant	8W8A	38.23	**72.14**	**63.18**	**56.90**	40.00	73.67	**57.93**	**57.44**
Llama3.2-1B	bf16	31.48	65.28	63.88	63.69	37.40	74.59	60.54	56.69
EfficientLLM-A-1B	bf16	**42.24**	**73.48**	**67.09**	**64.09**	**41.80**	**75.41**	**61.17**	**60.75**
EfficientLLM-A-1B-Quant	8W8A	41.64	73.32	66.51	63.98	42.40	75.24	60.30	60.48

Q10 Is it possible to dynamically move from one size of pruned model to another, or there needs to be retraining?

Thank you for the interesting idea. Retraining is optional; however, if we aim to achieve the highest possible accuracy, retraining is necessary. Since the goal of this paper is to explore the performance boundary of edge language models, we conduct retraining at a larger scale. As shown in Table 2, even with a small amount of retraining (50B), EfficientLLM-A-1B outperforms Llama-3.2-1B, which was pretrained on more than 9T tokens.

Q11 L132: What is a ”sub-architecture”?

We draw inspiration from neural architecture search. To achieve an automatically designed model, we first assume a supernet that encompasses all possible architectures. In the pruning problem, we treat the source model as the supernet, where a sub-architecture represents a sampled architecture within the supernet. Our ultimate pruning objective is to obtain the optimal ”sub-architecture”.

Q12 When does the pruning stage end during pretraining? How do users reach a predefined parameter count?

Pruning-aware pretraining only requires setting the target model size, allowing the model to automatically iterate its structure and stop once the target size is reached. The target size can be flexibly determined based on the user’s edge device requirements.