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 Unfair comparisons in Table 2.

This paper primarily explores the performance boundary of edge language models. Therefore, directly comparing our results with SoTA industrial models in Table 2 serves as a crucial support for our focus. Since most industrial models do not open source their training details, we are unable to conduct controlled experiments directly in Table 2. Instead, relevant comparisons are presented in Section 4.3 Ablation Studies and Section 4.4 Comparisons with LLM Pruning.

To evaluate pruning-aware pretraining, we conduct two experiments:

1) Because of time, we conduct small scale experiments. As suggested, we compare pretraining Llama-1.3B from scratch using 20B tokens from RedPajama and pruning-aware pretraining with total 5.7B tokens from the same dataset. Notice that, pruning-aware pretraining is trained from existing pretrained Llama2-7B. The pretraining stage is speeding up based on pruning.

Table 1. Comparison between training from scratch and pruning-aware pretraining.

Model	Tokens	ARC-C	ARC-E	BoolQ	HS	OBQA	PIQA	WG	Avg.
From Scratch	20B	24.66	46.80	56.73	34.61	30.40	63.87	50.67	43.96
Pruning-Aware Pretraining	5.7B	28.58	56.90	62.42	49.81	32.20	68.93	55.49	50.62

2) The large-scale evaluation is demonstrated in the comparison with SmolLM-135M in Table 2. Both SmolLM-135M and EfficientLLM-134M are trained using the same pretraining data, schedule, and a similar number of tokens. Notably, SmolLM-135M undergoes pretraining with 256B tokens, which has already reached its saturation accuracy. According to their report, the performance boundary is approached at around 300B tokens. With pruning-aware pretraining, EfficientLLM-A-134M extends this performance boundary using the same dataset.

Table 2. Performance comparison between SmolLM-135M and EfficientLLM-134M.

Model	Total Tokens	ARC-C	ARC-E	BoolQ	HS	OBQA	PIQA	WG	Avg.
SmolLM-135M	600B	29.35	61.32	59.85	42.67	34.40	68.55	52.96	49.87
EfficientLLM-134M	550.3B	30.97	62.88	60.40	43.81	33.60	68.82	53.28	50.54

Q2 Unclear boundaries of pretraining and continued pretraining.

Thank you for this advice. We mainly follow the definition from ShearedLlama (ICLR 2024), one of the most influential papers in this field:

1) Pruning Stage: Given an existing pretrained LLM as the source model for pruning, both ShearedLlama and EfficientLLM start from the source LLM and prune it down to a much smaller target edge model.

2) Continued Pretraining After Pruning: ShearedLlama introduced this stage, highlighting that continued pretraining is crucial for restoring model accuracy. In this work, we adopt ShearedLlama's setup. However, unlike ShearedLlama, which struggles with scaling up pruning stage (limited to 0.4B parameters for pruning), we propose an efficient scaling method during the pruning stage.

We conduct pruning during the pruning stage and refer to our scaled-up pruning stage as pruning-aware pretraining.

To remain consistent with ShearedLlama, we use a 50B token setup for continued pretraining. Since our main goal is to explore the performance limits of edge-side models, we further scale up continued pretraining to 500B and 320B tokens. As stated in line 321 of the main text, even without this scaling, we can still achieve state-of-the-art (SoTA) performance.

As mentioned in Appendix A.2 and B.1, the settings for continued pretraining, model architectures, and pretraining token counts are detailed in Table 5 and Table 7, due to space constraints in the main paper. The pretraining corpus is described in lines 305–315 (left column) and is the same as that used for SmolLM. We will add more background on ShearedLlama in the next version.

Q3 What is the reason of adding OpenWebMath?

Please refer to the SmolLM report: https://huggingface.co/blog/smollm (in the ”Training” section), where OpenWebMath constitutes 5.5% of the training data. Our training data is largely consistent with that of SmolLM.

Q4 Comparison with training from scratch.

In the reply to Q1, we conduct two settings to compare with training from scratch. The first one indicates that pruning-aware pretraining largely speeds up pretraining, while the latter indicates that pruning-aware pretraining can further extand the boundaries of small model performance, even compared with full training.

Q5 the reason to exclude MMLU in Table 3 and Table 4.

Although we wish to include it, most of the current LLM pruning works do not conduct MMLU evaluations. Additionally, since most LLM pruning works have not open-sourced their weights, we have excluded MMLU comparisons.

Q6 The number of total pretraining tokens are too small, less than 1T, where the models are far away from convergence.

We demonstrate the full convergence of small models from 3 aspects:

1) Both EfficientLLM and SmolLM limit the number of training tokens to 256B and undergo multiple training epochs to ensure complete convergence with 600B training tokens. SmolLM has shown that 300B tokens are sufficient to reach saturation accuracy.

2) We adopt a trapezoidal learning rate schedule, the same as SmolLM. Therefore, we only focus on the phase where the learning rate remains the same: 0–400B tokens. As shown in the figure, EfficientLLM-134M exhibits a stable during the continued pretraining phase, indicating that it has already achieved full convergence.

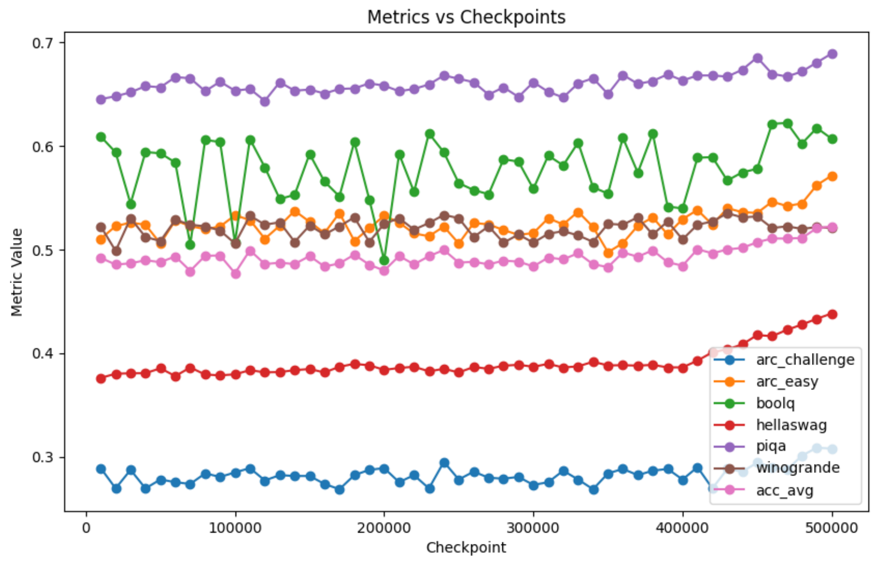

Figure 1. Performance in continued pretraining.

3) In Fig. 3 in the paper, we scale up to a maximum of 72.1B, mainly because for the total 256B pretraining tokens, we found that after reaching 72.1B, further scaling led to diminishing returns. Therefore, we selected a more cost-effective inflection point. The x-axis in Fig. 3 appears stretched, but the growth trend has plateaued.

Q7 Section 3 is unnecessarily long and unclear.

(a) Thank you for your advice, and we will add more related works, especially ShearedLlama. Equations (1) and (2) are different from iterative pruning:

1) Pruning-aware pretraining is a joint architecture auto-design and traditional pruning method. Iterative pruning can be seen as a way to implement network architecture search (NAS), and the specific difference is reflected in Equations (1) and (2).

2) For Equation (1), traditional pruning does not optimize the model structure, whereas the goal of Equation (1) includes both the selection of the architecture topology of the model and the selection of each layer's channels.

3) Equation (2), including Section 3.1, actually defines the model's search space. By using the concept of Iterative pruning in pruning, we express the essential difference between automatic model design and simple pruning.

(b) W refers to the weight of the linear layer; L indicates the number of attention blocks in the transformer; l and n refer to the l-th attention block, where the maximum number of layers is n.

(c) As mentioned in line 229, above Equation (9), Equation (9) is "acquired by directly substituting Equation (8) into Equation (3)," where Equation (3) itself forces the parameter groups to shrink over time. Equation (9) is the result of Equation (3) for iterative pruning.

(d) Thank you for your advice. Our redefinition includes both input and output parameter groups. To simplify the description of significance calculation, the definition of parameter groups forms the overall transformer structure search space, which differs from traditional pruning. Therefore, we believe Section 3 is necessary. We will also add more comparisons with other LLM expressions as suggested.

Q8 Why would scaling up tokens during the pretraining stage be more effective?

It is necessary to scale up the pruning stage as part of the pretraining for two main reasons:

1) ShearedLlama proposed that scaling up the continued training stage can effectively improve the edge model performance. However, SheardLlama does not address the issue of how to scale up the pruning process itself. Scaling up the pruning stage helps to retain more capabilities of the source model. If further development in the pruning field is desired, addressing how to scale up the pruning stage itself is essential.

2) Another effect of directly scaling up the pruning stage is the realization of automatic model architecture design, which directly integrates NAS methods and pruning methods. Due to the enormous computational cost of pretraining, traditional automated machine learning methods struggle to obtain automatically designed model architectures through multiple attempts. This paper, by decoupling the pruning process into the pretraining stage and designing the search space, achieves automatic architecture search in LLM pretraining for the first time.

Q9 Justification to ignore layer pruning.

Thanks for this question. Recent results show that for models with fewer parameters, deeper and thinner pretraining models tend to perform better, according to MobileLLM (ICML 2024) and PanGu-Pi-Pro (ICML 2024). Therefore, to obtain deeper edge models, we ignored layer pruning when designing the pruning space.

The basic differences can refer to reply of Q7 (a). We also mention the discussion in line 243(right), the discussion section.

1) scaling up LLM pruning in pretraining. Althrough some industrial LLMs such LlaMA-3.2 and MiniTron are also pretrained from larger models, the pruning stage itself may not scale up. MiniTron only iteratively prunes 4 times by a small calibration dataset, while the pruning-aware pretraining is continuously optimized by large-scale pretraining data.

2) Target-agnostic pruning. The auto-designed architectures achieve competitive results to SoTA human designed LLMs in modern pretraining for the first time.

3) Efficient Second-Order Updating. We propose efficient Hession approximations, making the second-order updating acceptable in pretraining.

Q11 Do you have any insight of your pruning method behaviors?

Yes, we mainly have 2 insights:

1) We recently further verified that by automatically designing the model architecture through pruning (ignoring model weights), we can achieve higher performance. This marks the first successful application of neural architecture search in large-scale pretraining. As shown in the table, after obtaining the model architecture through pruning, we perform random

weight initialization. Compared to the manually designed best practices (the baseline, which scales the source model's shape directly), direct pretraining can still achieve higher accuracy.

Table 3. Comparison of Human-practice and Auto-design models.

Model	**Arc_C**	**Arc_E**	**BoolQ**	**HS**	**OBQA**	**PIQA**	**WG**	**Avg.**
Human-practice	29.10	60.04	60.64	41.98	33.20	67.30	50.91	49.02
Auto-design	29.18	60.65	60.98	41.50	33.80	67.30	53.51	**49.56**

2) As shown in Figure 7 (right) in Appendix A.1, by recording the significance, we identified the optimal pruning locations for different pruning ratios, which can be valuable for future work: when the pruning ratio exceeds 54%, transformer model dimension pruning should be applied.

Table 4. Pruning Ratio and Pruning Type

Ratio	**Pruning Type**
0% - 54%	(I) + (II)
54% - 99%	(I) + (II) + (III)

Q12 What is your initialization of your model weights?

The weights are initialized from existing pretrained models like Llama2-7B, which follows the standard ShearedLlama pipeline in this field. We would add more backgrounds.