

# EfficientLLM: Scalable Pruning-Aware Pretraining for Architecture-Agnostic Edge Language Models

Xingrun Xing<sup>1,2</sup> Zheng Liu<sup>2</sup> Shitao Xiao<sup>2</sup> Boyan Gao<sup>3</sup> Yiming Liang<sup>1,2</sup> Wanpeng Zhang<sup>4</sup> Haokun Lin<sup>1</sup>  
Guoqi Li<sup>1</sup> Jiajun Zhang<sup>1</sup>

## Abstract

Modern large language models (LLMs) driven by scaling laws, achieve intelligence emergency in large model sizes. Recently, the increasing concerns about cloud costs, latency and privacy make it an urgent requirement to develop compact edge language models. Distinguished from direct pretraining that bounded by the scaling law, this work proposes the pruning-aware pretraining, focusing on retaining performance of much larger optimized models. It features following characteristics: 1) Data-scalable: we introduce minimal parameter groups in LLM and continuously optimize structural pruning, extending post-training pruning methods like LLM-Pruner and SparseGPT into the pretraining phase. 2) Architecture-agnostic: the LLM architecture is auto-designed using saliency-driven pruning, which is the first time to exceed SoTA human-designed LLMs in modern pretraining. We reveal that it achieves top-quality edge language models, termed EfficientLLM, by scaling up LLM compression and extending its boundary. EfficientLLM significantly outperforms SoTA baselines with  $100M \sim 1B$  parameters, such as MobileLLM, SmolLM, Qwen2.5-0.5B, OLMo-1B, Llama3.2-1B in common sense benchmarks. As the first attempt, EfficientLLM bridges the performance gap between traditional LLM compression and direct pretraining methods, and we will fully open source at <https://github.com/Xingrun-Xing2/EfficientLLM>.

## 1. Introduction

Large Language Models (LLMs) have become a central component of modern AI systems (Achiam et al., 2023; Guo et al., 2025) and are increasingly transforming daily life, particularly in mobile edge applications. However, typical LLMs (Touvron et al., 2023a), with 7 billion to 1 trillion parameters, require on-cloud deployment and continuous internet connectivity for interface. This places significant challenges in terms of latency, data-security and cloud-costs. In fact, fully using LLMs for mobile edge applications can be impractical, which requires approximately one million H100 GPUs (Liu et al., 2024). As a result, developing edge language models on resource-constrained devices becomes a recent tendency. For instance, MobileLLM (Liu et al., 2024) focuses on sub-one billion model sizes, which would fit in the DRAM of smartphones without excessive consumption.

Direct pretraining is dominant in recent tiny language model pretraining. Some practices such as MobileLLM and PanGu- $\pi$ -Pro (Tang et al., 2024) design deep-and-thin architectures for model efficiency. Other practices such as TinyLlama (Zhang et al., 2024a) and Qwen2.5-0.5B (Yang et al., 2024b) focus on scaling up pretraining data to 3T and 17T tokens. Based on best architectures and sufficient data, modern tiny models (Yang et al., 2024b; Groeneveld et al., 2024) are encouraging to touch the performance boundary. However, their overall performance appears to somewhat locked by the parameter scaling law (Kaplan et al., 2020): given limited model size, simply scaling up pretraining data is inefficient. More importantly, the intelligence emergency (Brown et al., 2020) is only observed on larger model sizes, meaning tiny models may never achieve this by direct pretraining alone. What is the next to train more efficient edge models remains an open challenge.

In parallel, LLM compression (Ashkboos et al., 2024a; Gu et al., 2024; Ashkboos et al., 2024b) focuses on retaining the performance of larger and stronger models while reducing computational cost. Despite its protential efficiency, existing methods (Sreenivas et al., 2024; Frantar & Alistarh, 2023; Xiao et al., 2023) compress LLM only using a small calibration dataset in post-training, which often results in significant performance degradation, making them unsuitable

<sup>1</sup>Institute of Automation, Chinese Academy of Sciences

<sup>2</sup>Beijing Academy of Artificial Intelligence <sup>3</sup>University of Oxford <sup>4</sup>Beijing University. Correspondence to: Zheng Liu <zhengliu1026@gmail.com>, Jiajun Zhang <jjzhang@nlpr.ia.ac.cn>.

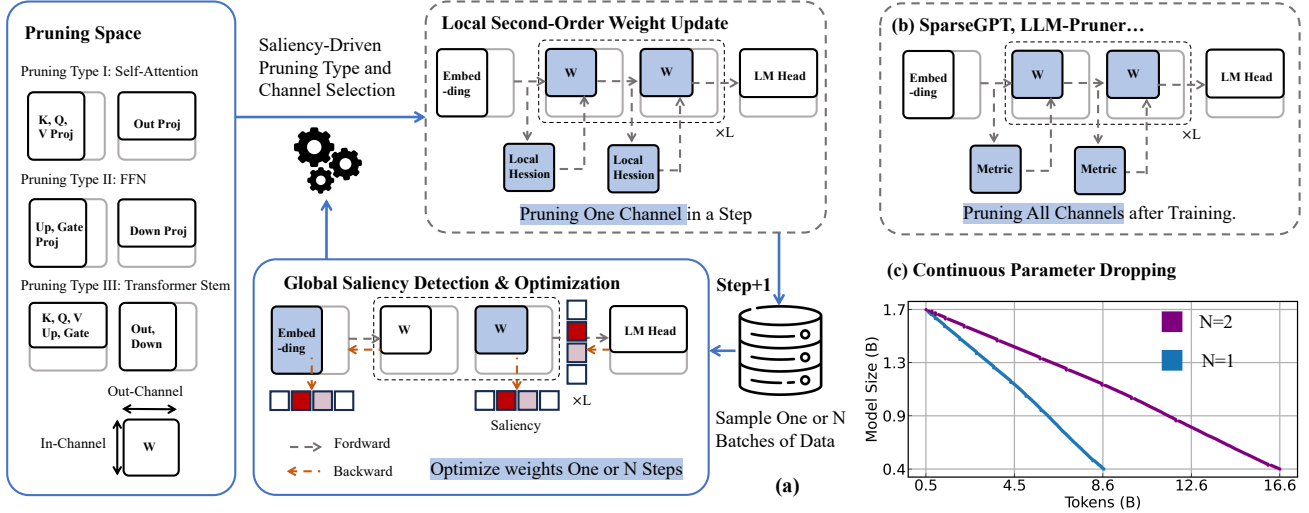


Figure 1. An overview of pruning-aware pretraining. (a) Training loop includes the joint saliency detection and weight optimizing, pruning type selection from pruning space, and second-order weight updating. (b) Traditional post-training pruning can be embedded in the training loop to scale up. (c) Continuous model size compression in pretraining.

for top-quality edge language models. Recently, ShearedLlama (Xia et al., 2023) initializes from an optimized LLM, improving training efficiency. However, the constrained optimization (Platt & Barr, 1987) hinders scaling up pruning stage and the performance gap to direct pretraining still remains. This work extends the performance boundary of traditional LLM compression by scaling up training data, which is underexplored but essential in this field.

This work proposes the pruning-aware pretraining to extend the efficiency boundary of edge language models. A family of top-efficiency edge language models in  $100M \sim 1B$  sizes are pretrained, named EfficientLLM. As shown in Fig. 1, we formulate pruning-aware pretraining as a bi-level optimization problem, and decouple the LLM pruning at every pretraining step. Driven by saliency, the overall architecture can be auto-designed (Zoph et al., 2018; Yu et al., 2020) according to predefined pruning space step by step. Compared with direct pretraining, pruning-aware pretraining leverages the performance of much larger optimized models, which direct pretraining smaller models never achieves. Compared with post-training pruning, it scales up the pruning stage with pretraining data. As shown in Fig. 2, pruning-aware pretraining scales up vanilla LLM-Pruner, achieving more than a 10% increase in accuracy. This work advances both edge language models and LLM compression:

- We propose a family of SoTA edge language models, EfficientLLM, in  $100M \sim 1B$  sizes. EfficientLLM exceeds traditional parameter scaling law and extends efficiency boundary of tiny models.
- We propose the pruning-aware pretraining, promoting LLM compression to the era of pretraining. By scaling up training data, vanilla LLM-Pruner significantly

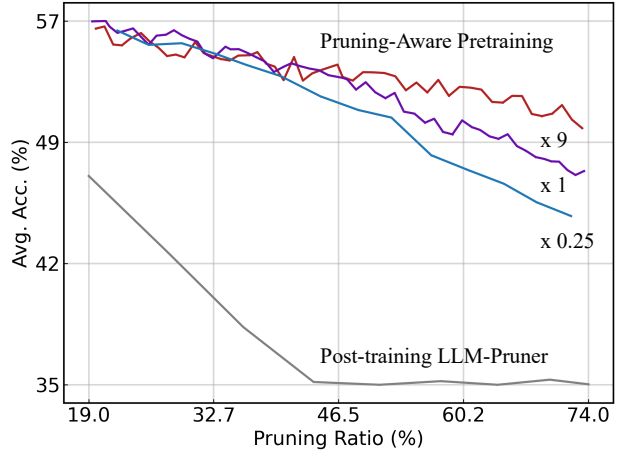


Figure 2. Performance of Pruning-Aware Pretraining. By scaling up LLM-Pruner in pretraining, performance of the source model is retained even if the pruning rate more than 70%.

exceeds SoTA methods without bells and whistles.

- We explore the auto-designed architectures in modern pretraining for the first time. Saliency-driven architectures are competitive with human best practices.

## 2. Preliminary and Related Works

**Edge Language Models.** Modern large language models are powered by the scaling law (Kaplan et al., 2020): larger models achieve higher data efficiency, making optimal training favor large models with moderate data. Towards accurate compact models, a lot of efforts explore the optimal training recipes: 1) data scale. OLMo-1B (Groeneweld et al., 2024), TinyLlama-1.1B (Zhang et al., 2024a),

Qwen2.5-0.5B (Yang et al., 2024b) pretrain on 2T, 3T, and 17T tokens respectively, which is significantly larger than the optimal data sizes according to scaling law. 2) Architectures. MobileLLM (Liu et al., 2024) shows that the deep-and-thin network and layer sharing achieve additional performance gains. However, previous works are bounded by the scaling law, and can be data-inefficient in pretraining. More recently, Llama3.2 (Dubey et al., 2024) and MiniTron (Sreenivas et al., 2024) introduce distillation and pruning for data-efficient training. There are mainly 2 drawbacks which addressed in this work: 1) the LLM pruning itself does not scale up. MiniTron only uses a small calibration dataset for pruning and only scales up recovery training, while this work scales up pruning itself to retain more performance. 2) the distillation (Ko et al., 2024) in pretraining is not training-efficient. Teacher models are typical 7B sized LLM (Touvron et al., 2023b) with more than  $\times 50$  FLOPs than a sub-billion edge model, which we delate in EfficientLLM.

**LLM Pruning** (Dong et al., 2024; Zhang et al., 2024b; Zhao et al., 2024; Bhaskar et al., 2024). We mainly focus on structural pruning (Chen et al., 2023; Choukse et al., 2018) for hardware friendly edge models. The most widely used LLM pruning is based on the Taylor expansion (LeCun et al., 1989; Hassibi et al., 1993; van der Ouderaa et al., 2023), as shown in Table 1. By calibration, typical SparseGPT (Frantar & Alistarh, 2023) and Wanda (Sun et al., 2023) can only applied in simi-structured pruning; LLM-Pruner (Ma et al., 2023) only achieves 20% pruning ratio with reasonable accuracy. Even if pruning with finetuning, LoraPrune (Zhang et al., 2023a) can only prune in 50% ratio. So there is an urgent requirement to scale up LLM pruning in pretraining. Another line of works learn to initialize from source model such as ShearedLlama (Xia et al., 2023) and NutePrune (Li et al., 2024) with less than 0.5B tokens. However the constraint optimization is biased in large-scale pretraining and this work scales up Taylor expansion metrics.

### 3. Pruning-Aware Pretraining

According to scaling laws, both the scale of training data and the number of parameters are fundamental to the emergence of intelligence in modern LLMs. Direct pretraining of smaller models is inefficient and lacks generalization ability. Model compression methods, although based on pretrained large models, fail to meet the data scale requirements and suffer from significant performance drop.

The principle of this work is to bridge the gap between direct pretraining and LLM compression by considering both the source model scale and pretraining data scale. In practice, pruning-aware pretraining continuously drops insignificant parameters and scales up pruning in pretraining.

**Problem Formulation.** Finding a sub-network from a pre-

Table 1. Comparison between Taylor-expansion based LLM pruning. “PT”, “FT”, “CB” denote pruning in the pretraining, finetuning, calibration stage respectively, and “G”, “U”, “Ratio” denote global pruning, weight update, pruning ratio.

Method	PT	FT	CB	G	U	Ratio
Magnitude	–	–	–	✗	✗	–
SparseGPT			✓	✗	✓	2:4
Wanda			✓	✗	✗	2:4
LLM-Pruner			✓	✓	✗	20%
LoraPrune		✓		✓	✗	50%
EfficientLLM-A	✓			✓	✗	>70%
EfficientLLM-B	✓			✓	✓	>70%

trained LLM is non-trivial. Given an optimized LLM, post-training LLM pruning focuses on finding optimal channels in each layer towards a target architecture. However, for edge language models, it is still challenging to define the efficient target architecture from its source model. For instance, MobileLLM shows the deeper architecture is better than the wider for sub-billion LLMs by human design and practice. This best practice can be sub-optimal in the context of the given source model, because each source model has distinguished salient pruning target. We formulate the architecture-agnostic pruning problem as:

$$\min_{a \in \mathcal{A}} \min_{c \in \mathcal{C}} \min_w \mathcal{L}_{pretrain}(a, c, w | \mathcal{M}), \quad (1)$$

where  $\mathcal{A}$  and  $\mathcal{C}$  are sub-architectures and sub-channels sampled from the source model  $\mathcal{M}$ . We jointly optimize pre-training loss through three factors: 1) the sub-architecture (Cai et al., 2019), 2) the sub-channels, and 3) the model weights. We outline the pruning-aware pretraining in Fig. 1 and detail each part in the following subsections.

#### 3.1. Defining Minimal Pruning Group

To achieve architecture-agnostic pruning, we first define the minimal parameter groups as the minimal architectures of the network, which should be flexible enough to construct any shape transformer models. Given an optimized large model  $\mathcal{M}$ , the pruned model  $\mathcal{M}^*$  can be represented as:

$$\mathcal{M}^* = \mathcal{M} - \sum_{t=1}^n g_t, \quad \text{s.t.} \quad \min_{g_t \in \mathcal{G}} \mathcal{L}_{pretrain}(\mathcal{M}), \quad (2)$$

where  $g_t$  is the mini-group of parameters pruned in step  $t$ , and  $\mathcal{G}$  is the pruning space formulated by defined mini-groups. According to Eq. 2, the pruning are decoupled by  $t$  steps and can be approximately solved sequentially:

$$\begin{aligned} \mathcal{M}_t &= \mathcal{M}_{t-1} - g_t^* \\ \text{s.t.} \quad g_t^* &= \operatorname{argmin}_{g_t \in \mathcal{G}} \mathcal{L}_{pretrain}(g_t | \mathcal{M}_{t-1}). \end{aligned} \quad (3)$$

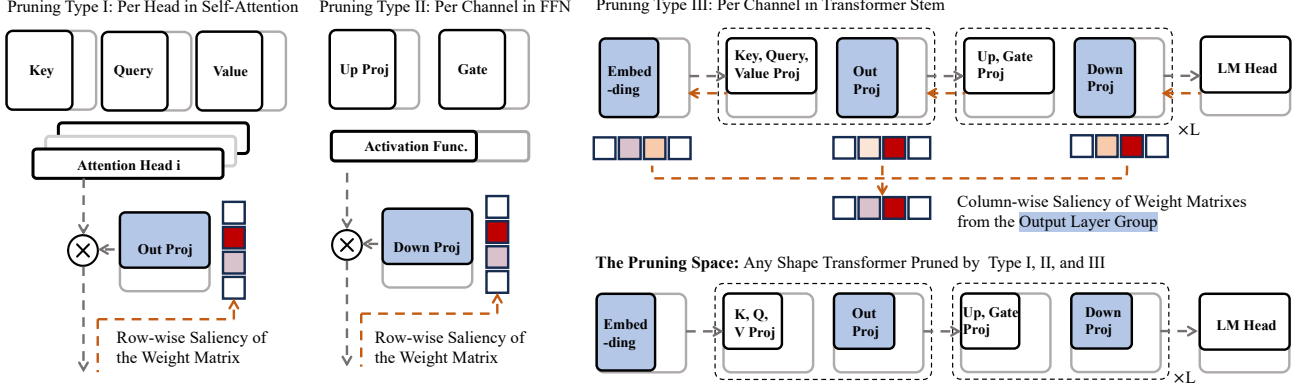


Figure 3. Three basic pruning types in the pruning space. We plot all the weight metrics with shape  $[D_{input}, D_{output}]$ . In backpropagation (in orange), the saliency of the output layer group (in blue) is calculated according to Eq. 9.

We first assume an optimal  $g_t^*$  in each pruning step with respect to the pretraining loss, and solve how to acquire  $g_t^*$  in the next subsection. In each pruning step, an optimal mini-group of parameters are selected and dropped from the pretraining LLM  $\mathcal{M}_{t-1}$ , allowing the model  $\mathcal{M}$  to adaptively reduce the number of parameters until a specific computation budget is met.

Towards fully structured pruning, two crucial constraints are considered in the mini-group design: (i) coupled architectures in different layers and (ii) consistent shape (Xia et al., 2023) in different blocks, where the former is introduced by LLM-Pruner to confirm all relevant parameters are pruned at the same time, and the later provides further speeding up in the system level. The difference from LLM-Pruner pruning space is that we split the pruned parameters into minimal groups and are able to adaptively combine during pretraining considering training dynamics, instead of the handcraft target in LLM-Pruner. Additionally, we couple channels in all blocks by saliency to satisfy constraint (ii).

For simplify, we indicate the **input layer group** as *query, key, value projections* in attention blocks; and *up, gate projections* in feed forward blocks. We indicate the **output layer group** as the *output projections* in attention blocks; and *down projections* in feed forward blocks. As shown in Fig. 3, three basic pruning types are defined:

(i) Per-head pruning in self-attention blocks: when an attention head is pruned, all the corresponding output channels in the input layer group and input channels in the output layer group are pruned at the same time. Additionally, we select the mini-group  $\mathcal{G}_{\text{attn}}^{(\ell)}$  in each block with the minimal saliency in step  $t$ , and merge  $\mathcal{G}_{\text{attn}}^{(\ell)}$  in all blocks as  $\mathcal{G}_{\text{attn}}$ :

$$\mathcal{G}_{\text{attn}} = \{W_{:,ij}^{(k,\ell)}, W_{:,ij}^{(q,\ell)}, W_{:,ij}^{(v,\ell)}, W_{ij,:}^{(o,\ell)}, \ell = 1, 2, \dots, n\}, \quad (4)$$

where  $W_{:,ij}$  and  $W_{ij,:}$  are column-wise and row-wise pruned, and  $ij$  corresponds to channels of an attention head.

(ii) Per-channel pruning in feed-forward blocks: when a

channel is pruned in the intermediate activation, the coupled channels include one output channel in the input layer group, and one input channel in the output layer group in a FFN block. Across different blocks, we also couple the minimal-saliency groups  $\mathcal{G}_{\text{ffn}}^{(\ell)}$  and merge to  $\mathcal{G}_{\text{ffn}}$ :

$$\mathcal{G}_{\text{ffn}} = \{W_{:,i}^{(up,\ell)}, W_{:,i}^{(q,\ell)}, W_{i,:}^{(down,\ell)}, \ell = 1, 2, \dots, n\}. \quad (5)$$

(iii) Per-channel pruning in the transformer stem: when a channel of the transformer stem is pruned, as shown in Fig.3, one coupled channel in the token embedding, one input channel in input layer group and one output channel in output layer group for every block, one input channel of the LM head projection is correspondingly pruned at the same time. We donate the stem mini-group as  $\mathcal{G}_{\text{stem}}$ :

$$\begin{aligned} \mathcal{G}_{\text{stem}} = & \{W_{i,:}^{(k,\ell)}, W_{i,:}^{(q,\ell)}, W_{i,:}^{(v,\ell)}, W_{:,i}^{(o,\ell)}\}, \dots \\ & \cup \{W_{i,:}^{(up,\ell)}, W_{i,:}^{(gate,\ell)}, W_{:,i}^{(down,\ell)}\}, \dots \\ & \cup \{\mathbf{w}_i^{(emb)}, W_{i,:}^{(head)}\}, \quad \ell = 1, 2, \dots, n \end{aligned} \quad (6)$$

where the  $i$  should be the same in every blocks, while  $ij$  needn't the same across blocks in Eq.4,5.

Given a transformer with hidden size  $m$ , head number  $h$ , intermediate size  $n$ , and  $l$  layers, the original pruning space is  $h^{(\ell)} \times n^{(\ell)} \times m$ . Notice that, the mini-groups are dynamically grouped by saliency in each pruning step, and we only choose among the 3 types to prune in step  $t$ . By coupling the parameters into mini-groups, the choice space is reduced to 3 in each step of Eq.3, and the final pruning space is  $3^t$ .

### 3.2. Optimizing Mini-Groups by Saliency

Based on the mini-groups, Eq.1 becomes a bi-level optimization problem of the mini-groups  $g$  and weights  $w$ :

$$\begin{aligned} \min_{g \in \mathcal{G}} \quad & \mathcal{L}_{\text{pretrain}}(g, w^* | \mathcal{M}), \\ \text{s.t.} \quad & w^* = \underset{w}{\operatorname{argmin}} \mathcal{L}_{\text{pretrain}}(w, g^* | \mathcal{M}), \end{aligned} \quad (7)$$



where the outer optimization could be solved by Eq.3, and the inner optimization could be directly solved by gradient descent. In pretraining, gradient descent and mini-group optimization (Eq.3) alternate. We refer to the alternating approach of one step of gradient descent followed by one step of pruning as pruning-aware pretraining  $\times 1$ .

In the mini-group optimization step, Taylor expansion evaluates the optimal mini-group  $g_t^*$  in Eq.3. For an optimized model, loss of any weight  $\mathbf{w}$  can be approximated by a second-order Taylor expansion around its optimal value  $\mathbf{w}^*$ :

$$\mathcal{L}(\mathbf{w}) \simeq \mathcal{L}(\mathbf{w}^*) + \delta \mathbf{w}^\top \nabla \mathcal{L}(\mathbf{w}^*) + \frac{1}{2} \delta \mathbf{w}^\top \mathbf{H}_{\mathcal{L}}(\mathbf{w}^*) \delta \mathbf{w} \quad (8)$$

where  $\mathcal{L}$ ,  $\nabla \mathcal{L}$ ,  $\mathbf{H}_{\mathcal{L}}$  is the global loss, gradient, hessian matrix; and  $\delta \mathbf{w} = \mathbf{w} - \mathbf{w}^*$ . We substitute Eq.8 into Eq.3:

$$\begin{aligned} g_t^* &= \operatorname{argmin}_{g_t \in \mathcal{G}} \mathcal{L}_{\text{pretrain}}(g_t | \mathcal{M}_{t-1}) \\ &= \operatorname{argmin}_{g_t \in \mathcal{G}} g_t^\top \nabla \mathcal{L}(\mathcal{M}_{t-1}) + \frac{1}{2} g_t^\top \mathbf{H}_{\mathcal{L}}(\mathcal{M}_{t-1}) g_t, \end{aligned} \quad (9)$$

where we omit the first term  $\mathcal{L}(\mathbf{w}^*) = \mathcal{L}(\mathcal{M}_{t-1})$  in Eq.8, because  $\mathcal{L}(\mathcal{M}_{t-1})$  is constant for the 3 mini-groups,  $\mathcal{G} = \{\mathcal{G}_{\text{attn}}, \mathcal{G}_{\text{ffn}}, \mathcal{G}_{\text{stem}}\}$ .

As shown in Fig.3, we define the calculation of mini-group saliency based on Eq.9, and then couple the mini-groups.

*Pruning Type I:* in each attention block, we only calculate element-wise saliency matrix for the output projection. To evaluate input channels, the saliency matrix is summed row-wise. We select  $\mathcal{G}_{\text{attn}}^{(\ell)}$  with the minimal row-wise saliency in each block, and then couple to  $\mathcal{G}_{\text{attn}}$ . The group saliency  $\mathcal{S}_{\text{attn}}$  is the summation over  $\mathcal{G}_{\text{attn}}$ .

*Pruning Type II:* in each FFN, we only calculate element-wise saliency matrix for the down projection as Eq.9, and then sum row-wise.  $\mathcal{G}_{\text{ffn}}^{(\ell)}$  with the minimal summed saliency are similarly coupled as  $\mathcal{G}_{\text{ffn}}$  with saliency  $\mathcal{S}_{\text{ffn}}$ .

*Pruning Type III:* in stem, we already have all the element-wise saliency in the output layer group based on Type I, II. To evaluate input channels, saliency matrixes are summed column-wise in a layer. We then sum column-wise saliency over the output layer group and select the minimum as  $\mathcal{S}_{\text{stem}}$ . In each step, the optimization in Eq.3 is finally solved by:

$$g_t^* = \operatorname{argmin}_{g_t \in \mathcal{G}} \{\mathcal{S}_{\text{attn}}, \mathcal{S}_{\text{ffn}}, \mathcal{S}_{\text{stem}}\}. \quad (10)$$

**Efficient Saliency Calculation.** In pretraining, we speed up the saliency calculation from 2 aspects: 1) Output layer group only calculation: a neural network is a directed acyclic graph (DAG) (Liu et al., 2018). For each node in the graph, pruning all of its inputs or all of its outputs is sufficient to prune the entire network. Additionally, since the number of parameters in the input layer group of a Transformer is 2 ~ 3 times that of the output layer group, we only

calculate the saliency of the output layer group for pruning. 2) Approximate Hessian matrix: existing works such as LLM-Pruner, SparseGPT, and Wanda have proposed a series of Hessian approximations to speed up. By substituting Eq.9, this framework is general to cooperate with previous post-training pruning metrics. Without loss of generalization, we choose LLM-Pruner to scale up.

### 3.3. Second-Order Weight Updating

Existing second order pruning applies the same Hessian matrix for the pruning weight detection and the remaining weight updating. However, calculating the global Hessian matrix is impossible in modern LLMs for its  $\mathcal{O}(n^4)$  complexity. A common approach is to use the squared error at each layer as a proxy for the global loss:  $\mathbf{H}_{\mathcal{L}} \simeq \mathbf{X}\mathbf{X}^T$ , such as in SparseGPT, OBC (Frantar & Alistarh, 2022). Although achieving the  $\mathcal{O}(d_{\text{row}} \times d_{\text{col}}^2)$  complexity, Hessian matrixes can not capture the global loss.

This work solves this problem by decouple the Hessian matrix in saliency detection and weight updating. In saliency detection, we approximate with global diagonal Hessian matrixes as LLM-Pruner to detect global saliency; in weight updating, we apply the layerwise approximation as  $\mathbf{H}_{\mathcal{L}} \simeq \mathbf{X}\mathbf{X}^T$  to minimize pruning error. And the remaining weights can be updated by  $\delta w_p = -\frac{w_p}{[\mathbf{H}^{-1}]_{pp}} \cdot \mathbf{H}_{:,p}^{-1}$ . Notice that, in each pruning, we only prune a mini-group including only one column of weights in a layer. To efficiently calculate the inverse of Hessian matrix  $\mathbf{H}_{:,p}^{-1}$ , we only need to solve a linear equation:  $\mathbf{e}_p = \mathbf{H}\mathbf{H}_{:,p}^{-1}$  in a step.

**Discussion.** This work advances LLM pruning in 3 aspects: 1) scaling up LLM pruning in pretraining. Although some industrial LLMs such as LLaMA-3.2 and MiniTron are also pre-trained 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 (Xu et al., 2021; Wu et al., 2019) achieve competitive results to SoTA human designed LLMs in modern pretraining for the first time. 3) Efficient Second-Order Updating. We propose efficient Hessian approximations, making the second-order updating acceptable in pretraining.

## 4. Experiments

Based on pruning-aware pretraining, we scale up LLM compression and extend the efficiency boundary of edge language models. We pretrain a family of top-quality edge language models, named EfficientLLM. Our results bridge the performance gap between LLM compression (Han et al., 2015) and direct pretraining methods for the first time.

Table 2. Zero-shot performance on World Knowledge and Common Sense Reasoning tasks. “Avg.” calculate among the 7 Common Sense Reasoning tasks. #Tokens count continued pretraining for EfficientLLM. All the results are evaluated on the same evaluation (Appendix B), except MobileLLM (Liu et al., 2024), because its close source evaluation can not be reproduced and we cite from the paper.

Model	#Tokens	#Params	MMLU	ARC-c	ARC-e	BoolQ	HellaSwag	OBQA	PIQA	WinoGrande	Avg.
OPT-125M	180B	125M	26.02	22.87	43.31	55.44	31.37	27.80	62.62	49.80	41.89
GPT-neo-125M	300B	125M	26.89	23.29	43.22	61.77	30.49	26.00	62.62	51.93	42.76
Pythia-160M	300B	162M	26.43	22.27	37.84	43.33	29.97	26.40	58.87	49.96	38.38
Memba-130M	1.2T	130M	27.65	24.49	47.56	54.68	35.11	29.00	64.69	53.35	44.13
MobileLLM-LS-125M	1T	125M	—	28.7	45.8	60.4	39.5	41.1	65.7	52.1	47.61
SmolLM-135M	600B	135M	30.05	29.35	61.32	59.85	42.67	34.40	68.55	52.96	49.87
EfficientLLM-A	500B	134M	30.54	30.97	62.88	60.40	43.81	33.60	68.82	53.28	50.54
OPT-350M	180B	331M	26.96	23.98	44.02	57.80	36.63	27.80	64.91	52.96	44.01
BLOOM-560M	350B	559M	27.32	24.40	46.04	44.46	36.54	28.80	62.57	53.20	42.29
Pythia-410M	300B	405M	29.10	24.15	51.39	59.20	40.20	29.40	66.70	53.83	46.41
MobileLLM-LS-350M	1T	345M	—	32.5	54.4	62.8	50.6	45.8	69.8	57.2	53.30
SmolLM-360M	600B	362M	33.89	36.26	70.16	55.23	53.51	37.60	71.38	57.22	54.48
Qwen2-0.5B	15T	494M	31.85	28.50	55.05	61.25	49.16	32.80	69.75	57.22	50.53
Qwen2.5-0.5B	17T	494M	33.37	32.17	64.44	61.99	52.09	35.20	70.29	56.20	53.20
EfficientLLM-A	50B	469M	33.09	35.92	70.50	59.85	53.16	35.00	72.69	56.27	54.77
EfficientLLM-A	500B	469M	34.54	38.40	72.10	62.42	56.84	40.40	73.83	57.46	57.35
OPT-1.3B	180B	1.3B	29.57	30.03	57.49	56.54	53.66	32.80	72.31	59.04	51.70
GPT-neo-1.3B	380B	1.3B	30.00	25.94	56.31	61.90	48.99	33.40	71.00	54.62	50.31
BLOOM-1.1B	350B	1.1B	29.16	25.77	51.73	59.51	43.11	29.60	67.30	54.62	47.38
Pythia-1B	300B	1.0B	30.14	26.96	56.86	60.04	47.15	31.20	70.29	52.88	49.34
TinyLlama-1.1B	3T	1.1B	32.30	30.29	60.40	56.85	59.13	35.80	73.07	59.04	53.51
ShearedLlama-1.3B	50B	1.3B	31.51	29.44	61.07	61.83	59.33	34.40	73.94	58.01	54.00
OLMo-1B	2T	1.2B	32.03	30.72	63.55	61.38	62.86	36.40	75.35	59.35	55.66
Llama3.2-1B	—	1.2B	36.31	31.48	65.28	63.88	63.69	37.40	74.59	60.54	56.69
EfficientLLM-A	50B	1.1B	36.71	40.36	73.61	62.39	60.24	40.20	75.19	61.25	59.03
EfficientLLM-A	320B	1.1B	37.71	42.24	73.48	67.09	64.09	41.80	75.41	61.17	60.75

#### 4.1. Settings

**Models.** To compare with the most general post-training pruning, EfficientLLM-A basically approximates Eq.9 as LLM-Pruner. EfficientLLM-B additionally applies the second-order weight updating based on EfficientLLM-A. Detailed settings are as follows: 1) in main results, we pre-train EfficientLLM 134M from the source model SmolLM-360M (Allal et al., 2024); EfficientLLM 469M and 1.1B from SmolLM-1.7B (Allal et al., 2024). 2) in comparisons with LLM pruning, we keep the same source models as baselines. For Llama-7B (Touvron et al., 2023a), large pruning ratios, including 50%, 70% are explored; for Llama2-7B (Touvron et al., 2023b), the 1.3B, 2.7B target models are applied to compare with ShearedLlama. Architecture details are shown in Appendix A.

**Data Composition.** EfficientLLM keeps the similar data distribution with the source model: (1) in main results, our pretraining data composition is similar to SmolLM, including 220B tokens from FineWeb-Edu (Lozhkov et al., 2024), 28B tokens from Cosmopedia v2 (Ben Allal et al., 2024a), 4B tokens from Python-Edu (Ben Allal et al., 2024b), and 27.5B tokens randomly sampled from OpenWebMath (Paster et al., 2023). (2) In comparisons with LLM pruning, we sample from RedPajama-1T (Weber et al., 2024) as pretraining data with the Llama family as the source model.

**Training.** In main results, we train EfficientLLM with both large scale pruning-aware pretraining and continued pre-training. For EfficientLLM-134M, 460M, and 1.1B, we pretrain 50.3B, 72.1B, and 36.7B tokens for pruning-aware

pretraining followed by 500B, 500B, and 320B tokens continued pretraining. Notice that, the large-scale continued pretraining is not necessary and 50B tokens also achieve competitive performance. Also notice that, the token number for pruning-aware pretraining is determined by iterations that reach the target number of parameters. We typically use the batchsize of 1M tokens with  $32 \sim 64$  A800 GPUs for  $100M \sim 1.1B$  models in pretraining. All the training details are shown in Appendix B.

**Evaluations.** For pretrained base models, we follow Llama, MobileLLM, and ShearedLlama to evaluate Common Sense Reasoning tasks: ARC (Clark et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), OBQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), and WinoGrande (Sakaguchi et al., 2021). The MMLU (Hendrycks et al., 2020) for Word Knowledge evaluation is also applied. For instruct finetuned model, we use Alpaca-Eval (Li et al., 2023) to compare the win rate.

#### 4.2. Main Results

**Edge Language Modeling.** For fair comparison, we collect main streams of edge language models in  $100M \sim 1B$  sizes, evaluate in the same conditions (Appendix B), and make a benchmark in Table 2. Early edge models including OPT (Zhang et al., 2023b), GPT-neo (Black et al., 2022), Pythia (Biderman et al., 2023), and BLOOM (Le Scao et al., 2023) are often pretrained in limited tokens and sub-optimal architectures, which largely hinder the performance. EfficientLLM efficiently solves these problems without too much additional costs: (i) instead of direct scaling up pre-

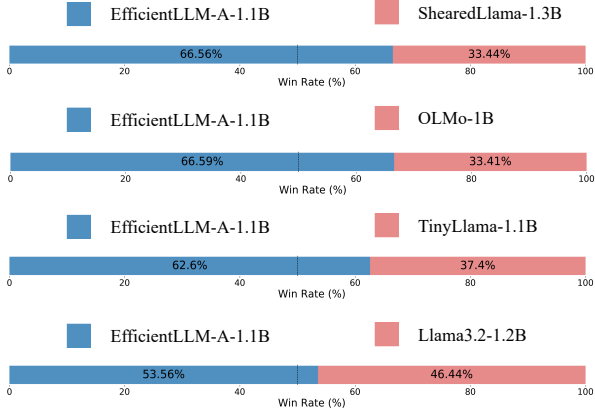


Figure 4. Win rate of EfficientLLM in the instruction tuning task.

training tokens like Qwen (Yang et al., 2024a;b), OLMo (Groeneveld et al., 2024), TinyLlama (Zhang et al., 2024a), EfficientLLM tries to retain the performance of larger and stronger models, which is significantly data-efficient. (ii) Through target-agnostic pruning, EfficientLLM automatically adapts to more salient architectures, achieving similar purpose in recent SoTA MobileLLM that rely on manual search for optimal architectures. As a result, EfficientLLM achieves higher accuracy even with smaller model/data sizes. For instance, EfficientLLM-134M exceeds Pythia-410M by 4.13% average accuracy. Specifically, with the same training data, EfficientLLM-469M with 50B continued pretraining tokens exceeds SmolLM-360M with 600B tokens in Common Sense Reasoning tasks. EfficientLLM-1.1B with 50B tokens exceeds OLMo-1B, TinyLlama, Llama3.2-1B in accuracy. Scaling up LLM compression achieves data-efficient pretraining which exceeds the traditional LLM scaling law.

According to Appendix B.3, most performance is retained by EfficientLLM-1.1B, and 1.05% accuracy is dropped from SmolLM-1.7B. Compared with SoTA industrial models such as Qwen2.5-0.5B and Llama3.2-1B trained by saturated tokens, EfficientLLM outperforms 4.15% and 4.06% respectively with limited pretraining data.

**Instruction Tuning.** We finetune EfficientLLM-1.1B base model and other 3 top-quality pretrained base models includes OLMo-1B, ShearedLlama-1.3B, TinyLlama-1.1B and Llama3.2-1B in the same condition. We use the Alpaca dataset (Taori et al., 2023) with 52K instructions and finetune for 3 epochs. As shown in Fig. 4, EfficientLLM-1.1B significantly outperforms SoTA baselines, indicating the generalization ability in the supervised finetuning (SFT). More case studies are shown in Appendix C.

### 4.3. Ablation Studies.

**Scalability.** According to Eq. 7, we set the ratio of pruning steps to gradient descent steps to 4:1, 2:1, 1:1, and 1:9

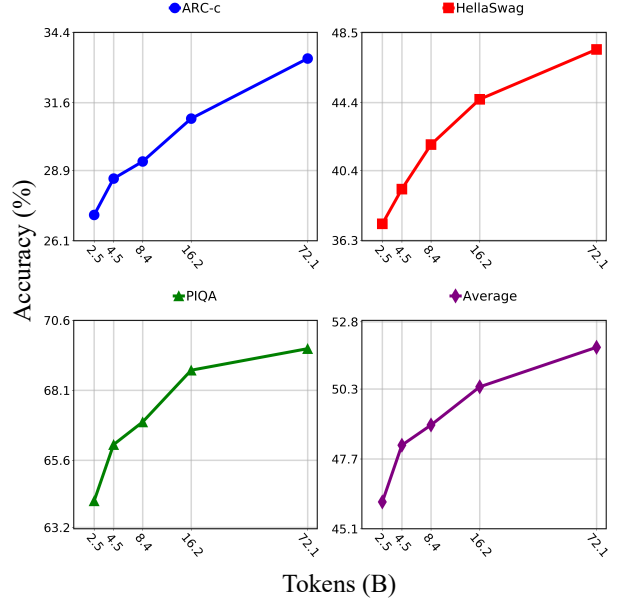


Figure 5. Scalability of pruning-aware pretraining.

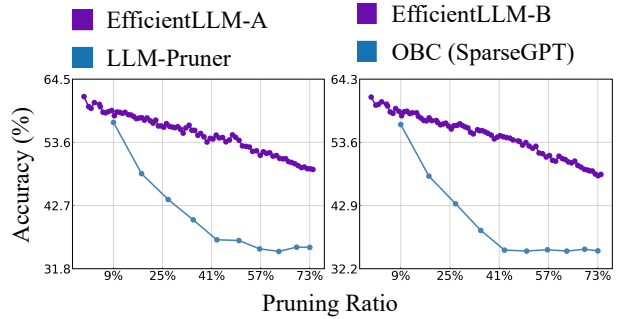


Figure 6. Generalization of pruning-aware pretraining for different pruning metrics. LLM-Pruner and SparseGPT are scaled up.

in a iteration, respectively. When the target model size is reached, the pruning-aware pretraining requires 2.5B, 4.5B, 8.4B, and 72.1B tokens of pretraining, respectively. Fig. 5 indicates that scaling up pruning-aware pretraining continuously improves pruning performance. Therefore, by scaling up LLM pruning during pretraining, the upper boundary of LLM compression can be extended.

**Generalization.** There is a large number of methods that perform post-training pruning based on second-order Taylor expansion, such as OBC and SparseGPT. As shown in Fig. 6, we generalize EfficientLLM to the second order updating case as EfficientLLM-B. Compared with post-training settings, EfficientLLM retains source model performance consistently in large pruning ratio. We observe similar performance of EfficientLLM-A/B in large scale pruning-aware pretraining, but when the pruning data is small ( $< 1B$ ), EfficientLLM-B significantly improves accuracy in Table 4.

Table 3. Comparisons of LLM pruning in Llama-7B. We scale up pruning-aware pretraining to 5B tokens for EfficientLLM. #Tuning donates whether to finetune after pruning. Most works report finetuned results.

#Ratio	Model	#Tuning	ARC-c	ARC-e	BoolQ	HellaSwag	OBQA	PIQA	WinoGrande	Avg.
50%	MaP	✓	30.63	49.32	39.69	42.49	31.40	66.81	50.67	44.43
	MvP	✓	26.79	44.07	59.94	40.98	31.80	63.06	55.64	46.04
	WANDA	✓	34.20	42.68	50.90	38.12	38.78	57.38	55.98	45.43
	LLM-Pruner	✓	28.24	46.46	61.47	47.56	35.20	68.82	55.09	48.98
	LoRAPrune	✓	31.62	45.13	61.88	47.86	34.98	71.53	55.01	49.72
	LoRAShear	✓	32.26	47.68	62.12	48.01	34.61	71.80	56.29	50.40
	Compresso	✓	27.82	48.82	60.09	39.31	33.40	66.70	51.93	46.87
	NutePrune	✗	31.74	46.59	62.20	53.87	35.80	69.91	57.77	51.13
	NutePrune	✓	32.17	51.68	62.26	55.88	34.40	71.00	57.54	52.13
	EfficientLLM-A	✗	30.80	52.15	62.29	54.70	35.20	71.33	56.75	51.89
	EfficientLLM-A	✓	34.04	64.81	64.83	60.12	34.60	73.88	61.48	56.25
70%	LLM-Pruner	✓	24.83	39.56	47.28	31.66	28.80	60.83	50.75	40.53
	NutePrune	✓	26.19	42.17	62.08	39.43	30.20	62.30	51.46	44.83
	EfficientLLM-A	✗	27.73	54.50	47.89	47.77	31.00	68.17	55.17	47.46
	EfficientLLM-A	✓	29.95	58.59	58.13	52.02	34.60	70.08	55.96	51.33

Table 4. Comparisons with ShearedLlama in Llama2-7B. #Pruning and #Tuning donate tokens used in the pruning and finetuning stages respectively. ShearedLlama is evaluated from the official huggingface checkpoint and we finetune in the same condition.

Model	#Pruning	#Tuning	ARC-c	ARC-e	BoolQ	HellaSwag	OBQA	PIQA	WinoGrande	Avg.
ShearedLlama-2.7B	0.4B	–	26.37	49.62	59.02	47.15	33.00	66.59	53.83	47.94
EfficientLLM-A-2.7B	0.66B	–	26.71	55.51	57.31	47.58	32.20	67.14	56.51	48.99
EfficientLLM-A-2.7B	10.56B	–	29.61	61.15	62.23	54.91	34.40	71.11	57.46	52.98
ShearedLlama-1.3B	0.4B	–	22.78	41.08	60.18	34.66	28.20	63.00	50.67	42.94
EfficientLLM-A-1.3B	0.95B	–	23.12	45.66	55.78	38.23	28.00	62.73	50.99	43.50
EfficientLLM-B-1.3B	0.95B	–	24.57	45.75	60.40	38.24	30.40	63.11	51.54	44.86
ShearedLlama-1.3B	0.4B	5.6B	26.54	55.60	60.12	48.28	31.60	68.77	56.20	49.59
EfficientLLM-A-1.3B	0.95B	5B	27.30	56.44	57.58	50.03	31.00	69.10	54.70	49.45
EfficientLLM-B-1.3B	0.95B	5B	28.58	56.90	62.42	49.81	32.20	68.93	55.49	50.62
EfficientLLM-B-1.3B	10.56B	5B	30.20	60.02	60.18	53.22	31.60	70.40	58.01	51.95

#### 4.4. Comparisons with LLM Pruning

In this section, we conduct light-weight pruning-aware pre-training to compare with existing LLM pruning methods and ShearedLlama. Experiments reveal that only scaling up the pruning stage to 5B tokens can achieve much higher performance than what was possible previously (Table 3).

**Traditional LLM Pruning.** We mainly focus on large pruning ratio because it is more practical to achieve highly efficiency based on heavy source LLMs. In Table 3, we scale up pruning-aware pretraining to only 5B tokens. We report both results with or without finetuning after pruning. Because previous works finetune in different settings, we finetune additional 1B tokens if with it. Notice that, even without finetuning, EfficientLLM exceeds all the according baselines. It is shown that existing LLM pruning is impractical in large pruning ratio. By simply scaling up LLM-Pruner metric in pruning-aware pretraining, EfficientLLM-A significantly exceeds SoTA NutePrune 6.5% in 70% ratio without bells and whistles, while NutePrune integrates distillation and additional learnable masks. In 50% ratio, EfficientLLM exceeds LoRAPrune by 2.18% and 6.54% when with and without tuning. Results indicate that with effectively scaling up pruning data, even using the vanilla LLM-Pruner metric, performance can significantly outperform SoTA methods.

Therefore, the scalability in large pruning data is more crucial than finding better pruning metrics.

**Comparison with ShearedLlama.** ShearedLlama is optimized by constrained objective via lagrange multipliers, and joint trains binary masks. These additional targets influence the stability and slow down throughputs in large scale pretraining, so that, ShearedLlama only trains on 0.4B tokens in the pruning stage. As shown in Table 4, we conduct small scale pruning-aware pretraining with 0.66B/0.95B tokens. The token numbers are determined by pruning iterations to reach target model sizes. To keep fair comparisons, we use less tokens in finetuning to keep similarly total data. EfficientLLM-A-2.7B, EfficientLLM-B-1.3B exceed ShearedLlama 1.05% and 1.92% respectively. With finetuning, EfficientLLM-B-1.3B also exceeds ShearedLlama. Scaling up pruning stage would address higher accuracy, but ShearedLlama is inefficient to scale up the constraint optimization (Platt & Barr, 1987).

## 5. Conclusion

This work primarily advances the edge language model pretraining to exceed the traditional LLM scaling law. Distinguished from almost LLM compression in post-training, this work scales up existing pruning metric in the pretraining



stage, promoting LLM compression to the era of pretraining. Technically, minimal parameter groups are defined and optimized by saliency to address scalable target-agnostic pruning. The results reveal that even if vanilla LLM-Pruner can surpass SoTA pruning methods by scaling up and outperform direct pretraining edge models.

Future work mainly focuses on the high-level abilities such as code, math, and long context of edge language models, with the hypothesis that retaining existing abilities from LLMs is easier than direct emergency by the tiny model itself, which will boost the mobile edge intelligence.

## Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

## References

- Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.
- Allal, L. B., Lozhkov, A., Bakouch, E., von Werra, L., and Wolf, T. Smollm - blazingly fast and remarkably powerful, 2024.
- Ashkboos, S., Croci, M. L., Nascimento, M. G. d., Hoefler, T., and Hensman, J. Slicept: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024a.
- Ashkboos, S., Mohtashami, A., Croci, M. L., Li, B., Cameron, P., Jaggi, M., Alistarh, D., Hoefler, T., and Hensman, J. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024b.
- Ben Allal, L., Lozhkov, A., Penedo, G., Wolf, T., and von Werra, L. Cosmopedia, 2024a. URL <https://huggingface.co/datasets/HuggingFaceTB/cosmopedia>.
- Ben Allal, L., Lozhkov, A., Penedo, G., Wolf, T., and von Werra, L. Smollm-corpus, 2024b. URL <https://huggingface.co/datasets/HuggingFaceTB/smollm-corpus>.
- Bhaskar, A., Wettig, A., Friedman, D., and Chen, D. Finding transformer circuits with edge pruning. *arXiv preprint arXiv:2406.16778*, 2024.
- Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O’Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.
- Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.
- Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., et al. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*, 2022.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.
- Cai, H., Gan, C., Wang, T., Zhang, Z., and Han, S. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.
- Chen, T., Ding, T., Yadav, B., Zharkov, I., and Liang, L. Lorashear: Efficient large language model structured pruning and knowledge recovery. *arXiv preprint arXiv:2310.18356*, 2023.
- Choukse, E., Erez, M., and Alameldeen, A. R. Compresso: Pragmatic main memory compression. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 546–558. IEEE, 2018.
- Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.
- Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.
- Dong, P., Li, L., Tang, Z., Liu, X., Pan, X., Wang, Q., and Chu, X. Pruner-zero: Evolving symbolic pruning metric from scratch for large language models. *arXiv preprint arXiv:2406.02924*, 2024.
- Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan,

- A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- Fourrier, C., Habib, N., Wolf, T., and Tunstall, L. Lighteval: A lightweight framework for llm evaluation, 2023. URL <https://github.com/huggingface/lighteval>.
- Frantar, E. and Alistarh, D. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.
- Frantar, E. and Alistarh, D. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pp. 10323–10337. PMLR, 2023.
- Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac’h, A., Li, H., McDonnell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, 07 2024. URL <https://zenodo.org/records/12608602>.
- Groeneveld, D., Beltagy, I., Walsh, P., Bhagia, A., Kinney, R., Tafjord, O., Jha, A. H., Ivison, H., Magnusson, I., Wang, Y., et al. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*, 2024.
- Gu, Y., Dong, L., Wei, F., and Huang, M. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024.
- Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.
- Hassibi, B., Stork, D. G., and Wolff, G. J. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pp. 293–299. IEEE, 1993.
- Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- Hooper, C., Kim, S., Mohammadzadeh, H., Mahoney, M. W., Shao, Y. S., Keutzer, K., and Gholami, A. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*, 2024.
- Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
- Ko, J., Kim, S., Chen, T., and Yun, S.-Y. Distillm: Towards streamlined distillation for large language models. *arXiv preprint arXiv:2402.03898*, 2024.
- Le Scao, T., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., et al. Bloom: A 176b-parameter open-access multilingual language model. 2023.
- LeCun, Y., Denker, J., and Solla, S. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.
- Li, J., Fang, A., Smyrnis, G., Ivgi, M., Jordan, M., Gadre, S., Bansal, H., Guha, E., Keh, S., Arora, K., et al. Datacomp-1m: In search of the next generation of training sets for language models, 2024. URL <https://arxiv.org/abs/2406.11794>.
- Li, S., Chen, J., Han, X., and Bai, J. Nuteprune: Efficient progressive pruning with numerous teachers for large language models. *arXiv preprint arXiv:2402.09773*, 2024.
- Li, X., Zhang, T., Dubois, Y., Taori, R., Gulrajani, I., Guestrin, C., Liang, P., and Hashimoto, T. B. AlpacaEval: An automatic evaluator of instruction-following models, 2023.
- Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.
- Liu, Z., Zhao, C., Iandola, F., Lai, C., Tian, Y., Fedorov, I., Xiong, Y., Chang, E., Shi, Y., Krishnamoorthi, R., et al. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. *arXiv preprint arXiv:2402.14905*, 2024.
- Lozhkov, A., Ben Allal, L., von Werra, L., and Wolf, T. Fineweb-edu: the finest collection of educational content, 2024. URL <https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu>.
- Lv, B., Zhou, Q., Ding, X., Wang, Y., and Ma, Z. Kvpruner: Structural pruning for faster and memory-efficient large language models. *arXiv preprint arXiv:2409.11057*, 2024.

- Ma, X., Fang, G., and Wang, X. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.
- Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.
- Paster, K., Santos, M. D., Azerbayev, Z., and Ba, J. Open-webmath: An open dataset of high-quality mathematical web text, 2023.
- Platt, J. and Barr, A. Constrained differential optimization. In *Neural Information Processing Systems*, 1987.
- Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.
- Sreenivas, S. T., Muralidharan, S., Joshi, R., Chochowski, M., Patwary, M., Shoeybi, M., Catanzaro, B., Kautz, J., and Molchanov, P. Llm pruning and distillation in practice: The minitron approach. *arXiv preprint arXiv:2408.11796*, 2024.
- Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.
- Tang, Y., Liu, F., Ni, Y., Tian, Y., Bai, Z., Hu, Y.-Q., Liu, S., Jui, S., Han, K., and Wang, Y. Rethinking optimization and architecture for tiny language models. *arXiv preprint arXiv:2402.02791*, 2024.
- Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Stanford alpaca: An instruction-following llama model. [https://github.com/tatsu-lab/stanford\\_alpaca](https://github.com/tatsu-lab/stanford_alpaca), 2023.
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.
- van der Ouderaa, T. F., Nagel, M., Van Baalen, M., Asano, Y. M., and Blankevoort, T. The llm surgeon. *arXiv preprint arXiv:2312.17244*, 2023.
- Weber, M., Fu, D., Anthony, Q., Oren, Y., Adams, S., Alexandrov, A., Lyu, X., Nguyen, H., Yao, X., Adams, V., et al. Redpajama: an open dataset for training large language models. *arXiv preprint arXiv:2411.12372*, 2024.
- Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., and Keutzer, K. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10734–10742, 2019.
- Xia, M., Gao, T., Zeng, Z., and Chen, D. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, 2023.
- Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pp. 38087–38099. PMLR, 2023.
- Xu, J., Tan, X., Luo, R., Song, K., Li, J., Qin, T., and Liu, T.-Y. Nas-bert: Task-agnostic and adaptive-size bert compression with neural architecture search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1933–1943, 2021.
- Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., et al. Qwen2 technical report, 2024a. URL <https://arxiv.org/abs/2407.10671>.
- Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024b.
- Yu, J., Jin, P., Liu, H., Bender, G., Kindermans, P.-J., Tan, M., Huang, T., Song, X., Pang, R., and Le, Q. Bignas: Scaling up neural architecture search with big single-stage models. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pp. 702–717. Springer, 2020.
- Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.
- Zhang, M., Chen, H., Shen, C., Yang, Z., Ou, L., Yu, X., and Zhuang, B. Loraprune: Pruning meets low-rank parameter-efficient fine-tuning. *arXiv preprint arXiv:2305.18403*, 2023a.
- Zhang, P., Zeng, G., Wang, T., and Lu, W. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024a.

- Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models, 2022. URL <https://arxiv.org/abs/2205.01068>, 3:19–0, 2023b.
- Zhang, Y., Bai, H., Lin, H., Zhao, J., Hou, L., and Canistraci, C. V. Plug-and-play: An efficient post-training pruning method for large language models. In *The Twelfth International Conference on Learning Representations*, 2024b.
- Zhao, B., Hajishirzi, H., and Cao, Q. Apt: Adaptive pruning and tuning pretrained language models for efficient training and inference. *arXiv preprint arXiv:2401.12200*, 2024.
- Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.



## Appendix

### A. Auto-Designed Architectures

#### A.1. Visualization

As shown in Fig.7, we Visualize the pruning-aware pretraining. We prune SmolLM-1.7B to EfficientLLM-A-469M. In Fig.7 (right), the self-attention parameter groups and FFN parameter groups are iteratively pruned in the initial stage. After 44.49B-token pretraining, the transformer stem parameter groups start pruned. This indicates that for the typical human-designed transformer shape, there are more redundant parameters in the attention head and the intermediate of FFN compared with the transformer stem.

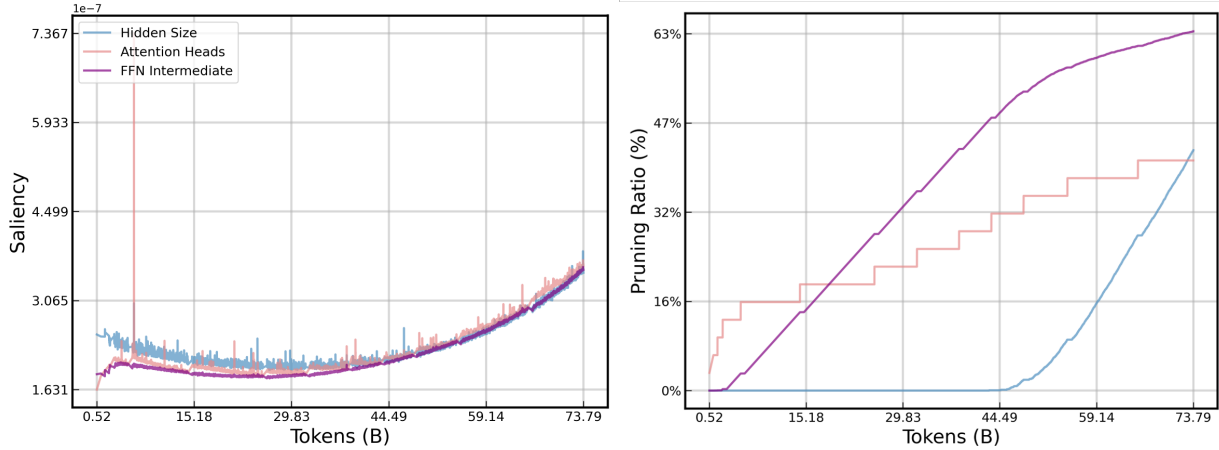


Figure 7. Visualization of pruning-aware pretraining. We plot the saliency of the three pruning types and their pruning ratio in training.

#### A.2. Architecture Comparisons

Table 5. Architecture comparisons between EfficientLLM and human-designed models.

Model	Hidden Size	FFN Intermediate	Attention Heads	Head Dim	Layer
MobileLLM-125M	576	1536	9	64	30
EfficientLLM-A-134M	757	966	5	64	32
MobileLLM-350M	960	2560	15	64	32
Qwen2/2.5	896	4864	14	64	24
EfficientLLM-A-469M	1195	3006	19	64	24
MobileLLM-1B	1280	3584	20	64	54
ShearedLlama-1.3B	2048	5504	16	128	24
OLMo-1B	2048	8192	16	128	16
Llama3.2-1B	2048	8192	32	64	16
EfficientLLM-A-1.1B	2048	4870	24	64	24

Table 6. Architectures in different pruning metrics to scale up by pruning-aware pretraining. We compare the approximate 460M model size. “x1” indicates that the number of gradient descent steps and pruning steps in each iteration are 1:1.

Model	Hidden Size	FFN Intermediate	Attention Heads	Head Dim	Layer
LLM-Pruner x1 (Ma et al., 2023)	1169	3082	19	64	24
OBC x1 (Frantar & Alistarh, 2022)	1131	3258	19	64	24
OBD x1 (LeCun et al., 1989)	1963	1542	12	64	24

As shown in Table 5, we compare the auto-designed architectures by saliency via pruning and the best practices of human design, including MobileLLM and Qwen2/2.5-0.5B, OLMo-1B, ShearedLlama-1.3B. In EfficientLLM, the pruning ratio of

hidden-size is smaller than attention heads and FFN intermediate channels driven by saliency.

As shown in Table 6, we compare the influence of different pruning metrics including the classic LLM-Pruner (Ma et al., 2023), OBC (Frantar & Alistarh, 2022) and OBD (LeCun et al., 1989). The OBD only uses the second-order term in Eq.9, which applied the diagonal of the Hessian matrix for approximate calculation.

### A.3. Cluster Attention

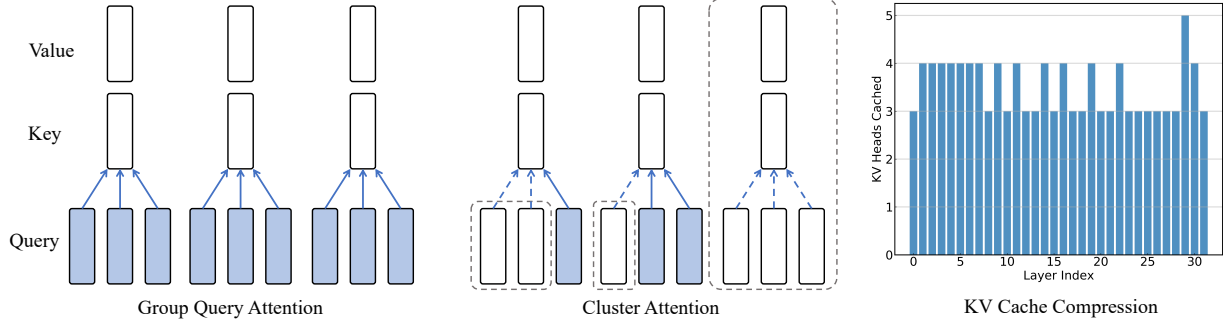


Figure 8. Group Query Attention (GQA) pruning. In the case of GQA, cluster attention can be obtained through pruning. After pruning, the number of query heads is the same in each layer, and the cluster attention compresses the KV Cache.

Pruning-aware pretraining could structurally prune the Group Query Attention (GQA) (Ainslie et al., 2023), which is usually applied for KV cache compression (Lv et al., 2024; Hooper et al., 2024) in LLMs. When the source model applies GQA, there are different cases in pruning:

- in all of the following cases, the query attention heads is the same in each layer, and the same as self-attention operation. The difference is how to share key and value for queries.
- As shown in Fig.8, if all queries corresponding to a key and value are pruned, then the key and value are also pruned.
- If a part of the query corresponding to a key and value is pruned, then the key and value are retained. This eventually forms cluster attention.

We plot an example of EfficientLLM-A-134M in Fig.8. And the source model of EfficientLLM-469M and EfficientLLM-1.1B do not apply GQA.

## B. Training and Evaluation Details

### B.1. Training

Our training code and models will be fully open sourced on GitHub and Huggingface. Detailed hyper-parameters are shown in Table 7. Notice that, the number of tokens in pruning-aware pretraining is determined by iterations to achieve the target model size, which is not directly defined. It can be adjusted through batchsize and the pruning frequency in each iteration.

Table 7. Hyper-parameters in pruning-aware pretraining and continued pretraining stages.

Model	#Tokens	Learning Rate	WarmUp Steps	Batchsize	Text Length	#GPU
Pruning-134M	50.3B	$2 \times 10^{-3}$	500	2 M	2048	32
Continued Pretrain-134M	500B	$2 \times 10^{-3}$	10000	1 M	2048	32
Pruning-469M	72.1B	$5 \times 10^{-4}$	500	1 M	2048	32
Continued Pretrain-469M	50B/500B	$2 \times 10^{-3}$	10000	1 M	2048	40
Pruning-1.1B	36.7	$5 \times 10^{-4}$	500	1 M	2048	32
Continued Pretrain-1.1B	50B/500B	$5 \times 10^{-4}$	10000	1 M	2048	64

## B.2. Evaluation

- **MMLU:** According to Datacomp-lm (Li et al.) (Appendix G of Datacomp-lm) and SmolLM (Allal et al., 2024), taking into account the log probabilities of complete answer sequences in MMLU is more related to weaker model performance, such as edge language models. Following SmolLM (Allal et al., 2024), we apply the open sourced Lighteval-v0.7.0 (Fourrier et al., 2023) to evaluate MMLU zero-shot performance.
- **Common Sense Reasoning:** Follow most of recent works (Xia et al., 2023; Ma et al., 2023; Li et al., 2024), we apply the widely used lm-evaluation-harness package (Gao et al., 2024) to evaluate zero-shot common sense reasoning tasks. To avoid different results introduced by different versions, we evaluate all the benchmarks with the 0.4.3 version, except Table 3 (50% pruning ratio). Because some previous works evaluate in older version 0.3.0 and we keep the same version in Table 3, 50% pruning ratio. Finally, all the comparisons keep in the same versions.

## B.3. Comparison with Source Models

Table 8. Comparison between SmolLM-1.7B and EfficientLLM-A-1.1B.

Model	#Params	ARC-c	ARC-e	BoolQ	HellaSwag	OBQA	PIQA	WinoGrande	Avg.
SmolLM-1.7B	1.7B	<b>46.16</b>	<b>76.60</b>	65.99	<b>65.74</b>	<b>42.00</b>	<b>75.95</b>	60.14	<b>61.80</b>
EfficientLLM-A-1.1B	1.1B	42.24	73.48	<b>67.09</b>	64.09	41.80	75.41	<b>61.17</b>	60.75

## C. Case Study in Alpaca Finetuning

In the instruction finetuning task, we finetune EfficientLLM-A-1.1B, OLMo-1B, ShearedLlama-1.3B and Llama3.2-1B in the Alpaca dataset. And then, we evaluate using the Alpaca-Eval. We sample some instructions in Alpaca-Eval make comparisons as follows:

### Input:

I want to eventually become a plant/manufacturing/production manager, and eventually a Chief Operating Officer. What sort of experience and post-graduate education path would you recommend? Please provide non-obvious and atypical career paths leading to such positions, on top of the classic ones.

### Output (EfficientLLM-A-1.1B):

To become a plant/manufacturing/production manager and eventually a Chief Operating Officer, you will need to have a combination of experience and post-graduate education.

For experience, you can start by working in a production or manufacturing role. This will give you the opportunity to gain hands-on experience in the industry and learn about the day-to-day operations of a production or manufacturing facility. You can also gain experience by working in a related field, such as engineering, logistics, or supply chain management.

For post-graduate education, you can pursue a degree in a related field, such as industrial engineering, operations management, or supply chain management. You can also pursue a master’s degree in a related field, such as operations management or supply chain management. Additionally, you can pursue professional certifications, such as the Certified Production and Inventory Management Professional (CPIMP) or the Certified Supply Chain Professional (CSCP).

In addition to experience and post-graduate education, you will also need to have excellent communication, organizational, and problem-solving skills. You will also need to be able to lead and motivate a team, as well as be able to think strategically and make decisions quickly.

Overall, to become a plant/manufacturing/production manager and eventually a Chief Operating Officer, you will need to have a combination of experience and post-graduate education, as well as excellent communication, organizational, and problem-solving skills. Additionally, you will need to be able to lead and motivate a team, as well as be able to think strategically and make decisions quickly.

**Output (OLMo-1B):**

To become a plant/manufacturing/production manager, you will need at least a Bachelor's degree in a relevant field, such as industrial engineering, manufacturing engineering, or production management. To become a Chief Operating Officer, you will need at least a Master's degree in a relevant field, such as business administration, management, or marketing. To become a plant/manufacturing/production manager, you will need at least a PhD in a relevant field, such as industrial engineering, manufacturing engineering, or production management. Additionally, to become a plant/manufacturing/production manager, you may also want to obtain a certification such as the Certified Production Manager (CPM) or Certified Quality Manager (CQM) to demonstrate your expertise in the field.

**Output (ShearedLlama-1.3B):**

To become a plant/manufacturing/production manager, you will need to have a bachelor's degree in a relevant field such as engineering, business, or manufacturing. After that, you can pursue a master's degree in a related field such as operations management, manufacturing engineering, or production and operations management. Additionally, you may want to pursue post-graduate education in areas such as project management, lean manufacturing, or supply chain management.

For a Chief Operating Officer position, you will need to have a bachelor's degree in a relevant field such as engineering, business, or manufacturing. After that, you can pursue a master's degree in a related field such as operations management, manufacturing engineering, or production and operations management. Additionally, you may want to pursue post-graduate education in areas such as project management, lean manufacturing, or supply chain management.

Overall, to become a plant/manufacturing/production manager or Chief Operating Officer, you will need to have a bachelor's degree in a relevant field such as engineering, business, or manufacturing, and then pursue a master's degree in a related field such as operations management, manufacturing engineering, or production and operations management. Additionally, you may want to pursue post-graduate education in areas such as project management, lean manufacturing, or supply chain management.

**Output (Llama3.2-1B):**

To become a plant/manufacturing/production manager and eventually a Chief Operating Officer, it is recommended to have a degree in engineering, business, or a related field, as well as relevant post-graduate education and experience. Examples of post-graduate education include an MBA, a Master's degree in engineering, or even a PhD in a related field. Relevant experience could include working in a manufacturing or production role, managing a team of engineers, or leading a plant or production department.

**Input:**

Why so many people like film photography when digital camera and mobile phone are more convenient and generate higher quality of photos?

**Output (EfficientLLM-A-1.1B):**

Many people still prefer film photography because of its unique aesthetic qualities. Film photography allows for more creative control, as the photographer can adjust the ISO, shutter speed, and aperture to achieve the desired effect. Additionally, film photography produces a unique look and feel that cannot be replicated with digital cameras and mobile phones. Film photography also allows for more experimentation, as the photographer can adjust the ISO, shutter speed, and aperture to achieve the desired effect. Finally, film photography can be a more personal and intimate experience, as it allows the photographer to capture moments in a way that digital cameras and mobile phones cannot.



**Output (OLMo-1B):**

Film photography has many advantages compared to digital camera and mobile phone, such as higher quality of photos, more creative control, and the ability to capture moments that would otherwise go unnoticed. Additionally, film photography requires more time and patience to develop the film, which adds an element of adventure and creativity to the process. Finally, many people find the physical process of developing film to be aesthetically pleasing, while digital camera and mobile phone processing can often produce photos with a less-than-desirable outcome.

**Output (ShearedLlama-1.3B):**

People like film photography because it allows for more creative control. With film photography, the photographer has the ability to adjust the exposure, focus, and other settings to create the desired effect. Digital cameras and mobile phones do not offer this level of creative control, making it difficult to achieve the desired effect. Additionally, film photography requires more thought and preparation than digital camera and mobile phone photography, making it more enjoyable and rewarding.

**Output (Llama3.2-1B):**

People like film photography because it produces higher quality of photos than digital cameras and mobile phones. Film photography is more expensive than digital cameras and mobile phones, but the quality of photos produced by film is much better than those produced by digital cameras and mobile phones. Additionally, film photography requires more skill and patience to take good photos, which many people enjoy.