On the Efficiency of Structured Pruning in Small Language Model Pretraining

Yixiao Li[1]* **Xianzhi Du**[2] **Ajay Kumar Jaiswal**[2] **Tao Lei**[2]
Tuo Zhao[1] **Chong Wang**[2]* **Jianyu Wang**[2]*
[1]Georgia Tech [2]Apple Inc.
yixiaoli@gatech.edu

Abstract

Companies that deploy generative language models on edge devices often need to train small models from scratch due to legal, data-governance, or product-customization requirements. An alternative approach is to train an enlarged model and compress it to the target size via structured pruning, but this advantage becomes unclear once the cost of pretraining the enlarged model is included. We introduce the enlarge-and-prune framework to study this question under full-cost accounting, and show that conventional enlarge-and-prune pipelines—which train the enlarged model, prune it, and recover in separate stages—do not consistently outperform direct small-model training under the same total token budget. To overcome this limitation, we propose IDEA Prune, an integrated enlarge-and-prune pipeline that unifies enlarged model pretraining, iterative structured pruning, and recovery under a single cosine annealing learning rate schedule. In controlled experiments compressing 2.8B models to 1.3B with up to 2T tokens on the same dataset, IDEA Prune consistently outperforms conventional pipelines and modestly improves overall performance relative to the scratch baseline, improving MMLU accuracy to 46.4%, compared with 31.4–33.4% for conventional pipelines. These advantages are preserved after downstream supervised fine-tuning, where IDEA Prune matches or outperforms all baselines of the same size on instruction following, math reasoning, and code generation. Our analysis further reveals that intermediate checkpoints provide more favorable pruning starting points than fully converged ones, and that an enlargement factor of approximately 2.6$\times$ yields the best pruned models in the tested regime.

1 Introduction

The deployment of generative language models on resource-constrained devices has become increasingly critical for modern applications. Tech companies face growing demands to deploy generative language models on smartphones and edge devices [Abdin et al., 2024, Gunter et al., 2024]. Given the severe memory limitations of these devices, model sizes must remain small, typically under 3 billion parameters for practical deployment [Liu et al., 2024, Team et al., 2024, Lozhkov et al., 2025, Deng et al., 2025].

To obtain small models, prior work suggests two broad strategies. One is to train the target small model directly with more data or more training tokens [Kaplan et al., 2020, Hoffmann et al., 2022]. The other is to start from a larger pretrained model and compress it to the target size. Structured pruning is a common approach in the second strategy: it removes selected model structures, so that the compressed model matches the desired deployment architecture [Wang et al., 2019, Xia et al., 2022, Kwon et al., 2022]. When the larger model is already available, pruning can be attractive

*Work done at Apple.

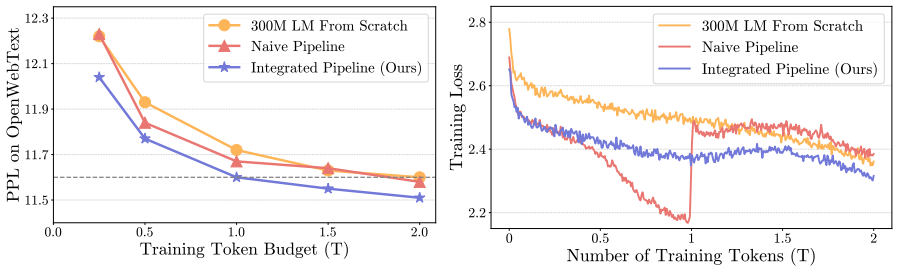
40th Conference on Neural Information Processing Systems (NeurIPS 2026).

because it may transfer the benefit of a higher-capacity source model without increasing the inference cost of the final small model [Xia et al., 2023, Muralidharan et al., 2024].

In this paper, however, we do not assume access to existing large checkpoints in order to match a more realistic industrial production setting. First, if the checkpoint is external, it may not be a valid input to an in-house training pipeline because of legal compliance, data governance, or product customization. Indeed, recent industrial on-device and small-model efforts often train their models within company-controlled pipelines rather than relying on arbitrary external checkpoints [Gunter et al., 2024, Team et al., 2024, Liu et al., 2024]. Second, if the checkpoint is internal, it may still be unsuitable as a pruning source. For example, internal cloud-serving models are often orders of magnitude larger than on-device targets: cloud models can exceed 100B parameters, whereas practical on-device models must remain below a few billion parameters [Team et al., 2024, Liu et al., 2024, Lozhkov et al., 2025, Deng et al., 2025]. This size gap is not only a bookkeeping issue: in our experiments in Section 5.2, even moderate enlargement factors beyond $5\times$ lead to clear pruning degradation, suggesting that compression ratios of tens of times would be even less practical. Thus, an internal checkpoint is not necessarily a suitable pruning source simply because it exists.

Given the aforementioned reasons, using pruning requires a different setup in the production setting: the source model must be deliberately trained as a moderately enlarged intermediate model and then pruned back to the deployment architecture. Because this intermediate model is introduced only to produce the final small model and is never deployed, its training cost must be counted in the end-to-end production budget. However, once this cost is included, is pruning still better than simply training the target small model from scratch? This paper studies this question under the same total token budget and training data.

To enable a rigorous comparison, we introduce the *enlarge-and-prune framework* (Section 2.3), which accounts for the full cost of training the enlarged model. Under this framework, we first examine a *naive enlarge-and-prune pipeline* that directly composes enlarged model pretraining with conventional two-stage pruning, and compare it against training the same small model from scratch under a matched token budget and on the same dataset. Figure 1a shows that the naive pipeline fails to consistently achieve superior token efficiency compared to direct pretraining of small models. This inefficiency stems from the divided nature of the naive pipeline: each stage uses its own learning rate schedule (Figure 2a, 2b), causing sharp loss increases at stage boundaries (Figure 1b) and catastrophic forgetting. Furthermore, naive pipelines often employ disparate training objectives between stages, such as auxiliary regularization losses for mask learning [Xia et al., 2023] and activation-based proxy losses [Muralidharan et al., 2024], leading to suboptimal initialization for the recovery stage.



(a) Perplexity on OpenWebText (b) Training Loss

Figure 1: *Naive pipeline*: 1T for enlarged model training and 1T for pruned model recovery, each with separate learning rate decay. *Integrated pipeline*: single learning rate schedule matching target-size model training, pruning started at 1T. **Left**: Perplexity on OpenWebText of 300M model trained with multiple training token budgets from 0.25T to 2T. The integrated pipeline is **2x** as token efficient as the naive pipeline. **Right**: Training loss curves for enlarge-and-prune pipelines. Training 300M models with 2T total token budget.

To overcome these limitations, we propose IDEA Prune, an **I**ntegrate**D** **E**nlarge-**A**nd-**Prune** pipeline with a novel iterative structured pruning for producing small language models from scratch. The integrated pipeline combines the enlarged model pretraining, pruning, and pruned model recovery stages under one cosine annealing learning rate schedule (Figure 2c), mitigating the knowledge loss caused by learning rate restarts in naive pipelines [Muralidharan et al., 2024, Xia et al., 2023]. Furthermore, we extend iterative pruning [Zhu and Gupta, 2017, Louizos et al., 2017] to structured Feed-Forward Network (FFN) width compression. Our approach progressively removes neurons based on element-wise importance scores computed across FFN weight matrices [Molchanov et al.,

2019, Zhang et al., 2022, Ding et al., 2019], effectively redistributing model capacity among surviving neurons for smooth compression.

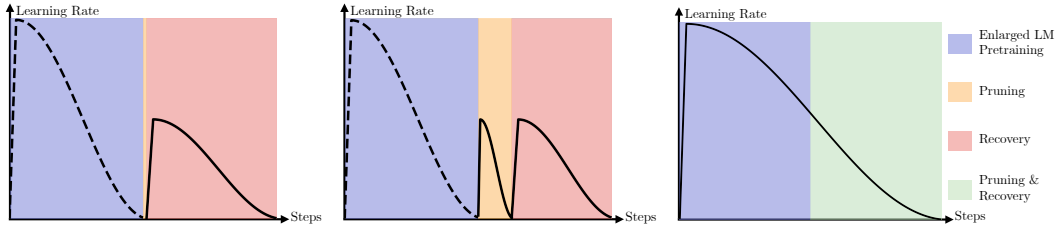
Our contributions in the deployment-constrained, full-cost regime are as follows:

• We introduce the enlarge-and-prune framework for full-cost evaluation and show that naive enlarge-and-prune pipelines do not consistently outperform training small models from scratch when the enlarged model cost is included.

• We propose IDEA Prune, an integrated pipeline that unifies enlarged model pretraining, iterative structured pruning, and recovery under a single learning rate schedule. In controlled comparisons on compressing 2.8B models to 1.3B with up to 2T tokens, IDEA Prune consistently outperforms all conventional pipelines overall and improves over the scratch baseline on most reported metrics. These advantages are preserved after downstream supervised fine-tuning on instruction following, math reasoning, and code generation benchmarks.

• Our analysis reveals that, under a resumed learning rate schedule, intermediate checkpoints—despite weaker absolute performance—provide more favorable starting points for pruning than fully converged checkpoints, and that an enlargement factor of approximately 2.6× yields the best pruned models in the tested regime.

For rigorous evaluation of the pruning methodology, we isolate our analysis from complementary techniques such as knowledge distillation, though we confirm in separate ablation studies that distillation can be effectively combined with our method.



(a) Minitron (b) Sheared LLaMA (c) IDEA Prune

Figure 2: Pipeline overview and learning rate schedules. Naive pipelines (Figure 2a, 2b) use separate LR schedules per stage, causing restarts and knowledge loss. IDEA Prune (Figure 2c) unifies all stages—enlarged pretraining, iterative pruning, and recovery—under a single cosine annealing schedule, gradually reducing model size without LR restarts.

2 Background

2.1 Iterative Pruning

Iterative pruning is a parameter reduction technique that alternates between parameter update and removal [Zhu and Gupta, 2017, Louizos et al., 2017]. During pruning, the method computes importance scores for parameters and gradually removes those less critical. For a model with parameters $\boldsymbol{\theta}^{(t)} \in \mathbb{R}^N$, we measure parameter importance by sensitivity [Molchanov et al., 2019], $\boldsymbol{\omega}^{(t)} \in \mathbb{R}^N$, which represents the first-order approximation of each parameter's impact on the training loss. See Appendix C.1 for the definition. We then generate a binary mask $\boldsymbol{v}^{(t)}$ based on the ranking of $\boldsymbol{\omega}^{(t)}$, and apply it to obtain the pruned parameters: $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{v}^{(t)} \odot \boldsymbol{\theta}^{(t)}$. This progressive approach enables a smooth transition from larger to smaller models. Note that these works focus on unstructured pruning, while we extend iterative pruning to structured pruning in Section 3.2.

2.2 Structured Pruning of FFN

Transformers [Vaswani et al., 2017], the mainstream architectures in current generative language models [Radford et al., 2019, Jiang et al., 2023, Dubey et al., 2024, Gunter et al., 2024], consist of sequential layers containing attention and Feed-Forward Network (FFN) components. We consider an FFN layer equipped with a gated Sigmoid Linear Unit (SiLU) activation function, denoted as $\sigma(\cdot)$. Given the input $\boldsymbol{x} \in \mathbb{R}^d$, the layer's output $\boldsymbol{y} \in \mathbb{R}^d$ is defined as

$$\boldsymbol{y} = \boldsymbol{W}_{\text{down}}^{\top}(\sigma(\boldsymbol{W}_{\text{up}}\boldsymbol{x}) \odot (\boldsymbol{W}_{\text{gate}}\boldsymbol{x})) = \sum_{i=1}^{h} \boldsymbol{W}_{\text{down}[i,:]}^{\top} \left(\sigma(\boldsymbol{W}_{\text{up}[i,:]}\boldsymbol{x}) \cdot (\boldsymbol{W}_{\text{gate}[i,:]}\boldsymbol{x})\right),$$

where $\boldsymbol{W}_{\text{up}}, \boldsymbol{W}_{\text{gate}}, \boldsymbol{W}_{\text{down}} \in \mathbb{R}^{h \times d}$, d is the input dimension, h is the hidden dimension or the number of neurons, and $\odot$ represents element-wise multiplication. Structured pruning of FFN width targets the removal of neurons, which corresponds to the rows in $\boldsymbol{W}_{\text{up}}$, $\boldsymbol{W}_{\text{gate}}$, and $\boldsymbol{W}_{\text{down}}$ with the same indices, for example, $\boldsymbol{W}_{\text{up}[i,:]}, \boldsymbol{W}_{\text{gate}[i,:]}, \boldsymbol{W}_{\text{down}[i,:]}$. This is achieved by finding a binary mask $\boldsymbol{m} \in \{0,1\}^h$ that determines which neurons to retain. The pruned weight matrices are obtained as:

$$\boldsymbol{W}_{\text{up}}^{'} = \text{diag}(\boldsymbol{m}) \times \boldsymbol{W}_{\text{up}}, \boldsymbol{W}_{\text{gate}}^{'} = \text{diag}(\boldsymbol{m}) \times \boldsymbol{W}_{\text{gate}}, \boldsymbol{W}_{\text{down}}^{'} = \text{diag}(\boldsymbol{m}) \times \boldsymbol{W}_{\text{down}}. \quad (2.1)$$

After mask application, the zero rows in each matrix are eliminated to obtain compressed weight matrices, resulting in reduced memory consumption and inference latency.

2.3 Enlarge-and-Prune Framework and Naive Pipeline

We introduce the enlarge-and-prune framework to evaluate the end-to-end token efficiency of pruning for producing small models from scratch. Given an inference budget, practitioners first fix the target small-model architecture for deployment. The target is then expanded into an enlarged counterpart, for example by increasing the FFN width in each layer, and this enlarged model is pretrained on a corpus. Next, structured pruning compresses the enlarged model back to the exact target architecture. A subsequent recovery stage continues pretraining the pruned model to regain performance. Under the full-cost accounting described in Section 1, this framework enables a controlled comparison between enlarge-and-prune pipelines and direct small-model training.

An immediate implementation under the enlarge-and-prune framework is the naive enlarge-and-prune pipeline, which consists of three distinct stages: enlarged model pretraining, pruning, and pruned model recovery. From the perspective of learning rate specifically, each of the three stages uses its own independent learning rate schedule [Xia et al., 2023]. The learning rate at the t-th step through the naive enlarge-and-prune pipeline is

$$\beta^{(t)} = \begin{cases} \eta(t; T_l, \eta_1, \eta_2) & \text{if } 0 < t \leq T_l, \\ \eta(t - T_l; T_p, \eta_3, \eta_4) & \text{if } T_l < t \leq T_l + T_p, \\ \eta(t - T_l - T_p; T_r, \eta_5, \eta_6) & \text{if } T_l + T_p < t \leq T, \end{cases}$$

where $\eta(\cdot;\cdot)$ is the cosine annealing learning rate schedule [Loshchilov and Hutter, 2016] (see Appendix A.2), T is the total training steps and T_l, T_p, T_r are the number of enlarged model training steps, pruning steps, pruned model recovery steps, respectively. An example can be found in Figure 2b. In the naive enlarge-and-prune pipeline, we need to tune different sets of peak and end learning rates $\eta_1, \ldots, \eta_6$, which requires considerable effort to find the optimal set.

2.4 Related Work

Train Large, Then Compress. Li et al. [2020] show that pretraining larger models with early stopping and then compressing them is more efficient than training smaller models to convergence. Our work differs in three respects: we use structured (not unstructured) pruning to preserve inference latency, we prune during pretraining to obtain a general-purpose model rather than during task-specific finetuning, and we target decoder-only architectures where pruning dynamics differ from encoder-only models [Liu et al., 2019, Xia et al., 2022, 2023].

Pruning during Pretraining. Recent concurrent work also moves pruning into the pretraining phase: Liu et al. [2025] propose pruning-aware pretraining that extends post-training criteria (LLM-Pruner, SparseGPT) into the pretraining phase for edge LLMs, and Jin et al. [2025] study sparse pretraining schedules and unify sparse/dense scaling laws, finding that pruning between 25% and 75% of training compute is near-optimal for sparse pretraining. These works focus on unstructured or architecture-search style sparsification, whereas we instantiate structured pruning within a unified cosine annealing schedule that eliminates stage-boundary LR restarts, and we evaluate under full-cost accounting against direct small-model training.

Structured Pruning in Post-Training. Recent works [Ma et al., 2023, Ashkboos et al., 2024, An et al., 2024, Wei et al., 2024, Xu et al., 2025, Guo et al., 2025, Zhao et al., 2025] compress pretrained models via structured pruning when checkpoints are off-the-shelf. Although the goals differ from ours, their pruning criteria can serve as drop-in components within our framework; we demonstrate this with activation-based and learned mask pruning in Section 5.7.

Alternative Structured Compression. Beyond FFN width reduction, depth pruning [Men et al., 2024, Kim et al., 2024], attention head pruning [Voita et al., 2019, Michel et al., 2019], and joint pruning of multiple dimensions [Krishnakumar et al., 2025] offer complementary strategies. Our framework is

not restricted to any single pruning dimension. We choose FFN-width as the primary instantiation because FFN layers account for the majority of parameters in decoder-only transformers, making width enlargement the most parameter-efficient strategy [Muralidharan et al., 2024]. Width pruning also preserves the layer count and attention structure, ensuring the pruned model exactly matches a predetermined deployment architecture. We provide a matched comparison with attention-head and joint pruning in Section 5.5.

Learning Rate Schedules. Recent work shows that the shape of the learning rate schedule, not only its peak value, governs the pretraining loss. Wen et al. [2025] explain Warmup-Stable-Decay (WSD) schedules through a river valley loss landscape, where a high learning rate makes progress along the river and the decay phase descends to the valley floor; Dremov et al. [2025] further analyze the training dynamics of this cooldown stage. Tissue et al. [2024] and Luo et al. [2025] predict the loss curve from the entire schedule, attributing the final loss to both the accumulated learning rate and the additional reduction from annealing. These views are consistent with our resumed-versus-restarted finding (Section 5.1): restarting the schedule from an annealed checkpoint re-raises the learning rate and discards part of the annealing benefit, whereas resuming a single schedule from an intermediate checkpoint preserves the accumulated progress and keeps a single final decay.

3 Method

We now describe IDEA Prune (**I**ntegrate**D** **E**nlarge-**A**nd-**Prune**), which unifies enlarged model pretraining, iterative structured pruning, and recovery into a single training run.

3.1 Integrated Enlarge-and-Prune Pipeline

We denote the number of enlarged model pretraining, pruning, and pruned model recovery steps as T_l, T_p, T_r, respectively. The learning rate for weight matrix update at step t is given by

$$\alpha^{(t)} = \eta(t; T_l + T_p + T_r, \eta_p, \eta_e),$$

where $\eta(\cdot;\cdot)$ is the cosine annealing learning rate schedule defined in (A.2). Unlike the naive enlarge-and-prune pipeline in Section 2.3, our integrated approach eliminates learning rate warm-up during pruning and recovery stages, thus mitigating potential knowledge loss.

Critically, this learning rate schedule method is not constrained to specific pruning techniques. We conduct ablation studies to evaluate its effectiveness across diverse approaches, including one-shot random pruning, activation-based pruning, and learned mask pruning (Section 5.7).

Furthermore, the integrated pipeline offers additional flexibility by enabling its application to existing pretrained models with available intermediate checkpoints and historical learning rate schedules, as discussed in Section 5.1.

3.2 Iterative Structured Pruning in FFN

To complement the integrated learning rate schedule, we propose a specialized iterative pruning method for FFN width based on the importance score of neurons. This approach further aligns the pruning and recovery stages by mitigating the suboptimal model initialization inherent in naive enlarge-and-prune pipelines during recovery training.

Concretely, for each element in a weight matrix $\boldsymbol{W}^{(t)} \in \mathbb{R}^{h\times d}$ at the t-th step, we denote the importance score as the sensitivity $\boldsymbol{S}_{ij}^{(t)}$ defined in (C.1). Since the sensitivity is defined on the full dataset, we use the moving average of importance scores on a mini-batch of size B to approximate it as

$$\widetilde{\boldsymbol{S}}_{ij}^{(t)} = (1-\lambda)\boldsymbol{T}_{ij}^{(t)} + \lambda\widetilde{\boldsymbol{S}}_{ij}^{(t-1)}, \tag{3.1}$$

where $\boldsymbol{T}_{ij}^{(t)} = \left|\frac{1}{B}\sum_{n=1}^{B}\nabla l_n(\boldsymbol{W}_{ij}^{(t)})\boldsymbol{W}_{ij}^{(t)}\right|$, λ is the moving average coefficient, and $l_n(\cdot)$ is the loss of the n-th sample in the mini-batch. The importance score for the k-th neuron is then derived by combining the moving average scores across the three weight matrices $\boldsymbol{W}_{\text{up}}, \boldsymbol{W}_{\text{gate}}, \boldsymbol{W}_{\text{down}}$:

$$\boldsymbol{c}_k^{(t)} = f_2\left(f_1(\widetilde{\boldsymbol{S}}_{\text{up}[k,:]}^{(t)}), f_1(\widetilde{\boldsymbol{S}}_{\text{gate}[k,:]}^{(t)}), f_1(\widetilde{\boldsymbol{S}}_{\text{down}[k,:]}^{(t)})\right), \tag{3.2}$$

where $f_1 : \mathbb{R}^d \to \mathbb{R}$ aggregates element-wise scores into a scalar and $f_2 : \mathbb{R}^3 \to \mathbb{R}$ combines across the three matrices. Options of f_1 and f_2 can be $\max(\cdot), \text{mean}(\cdot)$.

Finally, we update the pruning mask using a scheduled sparsity by

$$\boldsymbol{m}_k^{(t)} = \begin{cases} 1 & \text{if } \boldsymbol{c}_k^{(t)} \text{ is in top } (1 - r(t; T_p)) \text{ fraction of } \boldsymbol{c}^{(t)}, \\ 0 & \text{otherwise,} \end{cases} \tag{3.3}$$

where $r(t; T_p) \in [0, R]$ is the sparsity (fraction of neurons removed) at step t, defined as a cubically increasing function in Appendix C.2 that reaches the target sparsity R. The full IDEA Prune algorithm is summarized in Algorithm 1 (Appendix B).

4 Experiments

We investigate the effectiveness of IDEA Prune compared to the naive pipeline with baseline pruning methods, and study the token efficiency of the enlarge-and-prune pipeline compared to training small models from scratch. We further fine-tune all pretrained models with an identical recipe to examine whether these advantages are preserved in downstream tasks.

Experimental Setup. We choose a 1.3B target model and a 2.8B enlarged model differing only in FFN hidden dimension (Appendix F.1). All models are trained on DCLM [Li et al., 2024] (4T tokens) using AdamW [Loshchilov and Hutter, 2019] with sequence length 4096 and batch size 1024 on Google Cloud v5p TPUs using AXLearn [Lee et al., 2024]. We compare against (i) training 1.3B from scratch with 1T and 2T tokens, and (ii) naive enlarge-and-prune pipelines using OSRP, Minitron [Muralidharan et al., 2024], and Sheared LLaMA [Xia et al., 2023] (Appendix F.3). For IDEA Prune, we allocate 1T, 0.5T, and 0.5T tokens to enlarged pretraining, pruning, and recovery under a single cosine annealing schedule. We report perplexity on OpenWebText [Gokaslan et al., 2019], 0-shot accuracy on Arc-C [Clark et al., 2018], SciQA [Welbl et al., 2017], and HellaSwag [Zellers et al., 2019], 1-shot on TriviaQA [Joshi et al., 2017], and 5-shot on MMLU [Hendrycks et al., 2020]. Since the pruned model has the same architecture as the scratch baseline, inference latency and memory are identical; see Appendix F.4 and F.5 for full details.

Main Results. As shown in Table 1, IDEA Prune improves over all naive enlarge-and-prune pipelines. The improvement on OpenWebText and MMLU is the most significant: our method attains an OpenWebText perplexity of 8.88, outperforming the previous best of 8.96 achieved by Sheared LLaMA, and advances MMLU accuracy to 46.4%, compared to 31.4–33.4% for naive methods. However, different pipelines show minimal difference on reading comprehension and commonsense tasks: Arc-C, SciQA, HellaSwag, and TriviaQA. This is likely because performance saturates on these tasks at this model scale. As we increase the training tokens from 1T to 2T for the 1.3B model, these tasks are not significantly improved, but MMLU continues to improve. Results on Arc-Easy, WinoGrande, and CommonsenseQA show the same pattern (Appendix I).

Table 1: Evaluation of models trained from scratch or through enlarge-and-prune pipelines. We train a 1.3B model for 2T tokens as the baseline. We apply naive pipelines with different pruning methods to prune a 2.8B model to 1.3B with 2T tokens. We report the perplexity(↓) of OpenWebText and accuracy(↑) of other benchmarks. The best results are in **bold**.

Method	OWT ↓	Arc-C ↑	SciQA ↑	HellaSwag ↑	TriviaQA ↑	MMLU ↑
1.3B-1T from scratch	9.10	39.3	93.6	52.8	30.3	28.9
2.8B-1T from scratch	8.12	44.5	94.2	56.9	37.3	50.8
1.3B-2T from scratch	8.95	**39.4**	93.4	53.6	30.5	45.7
OSRP-2T (1.3B)	8.98	38.9	92.9	53.4	29.6	32.5
Minitron-2T (1.3B)	8.97	38.7	**94.6**	53.6	30.7	31.4
Sheared LLaMA-2T (1.3B)	8.96	**39.4**	94.0	53.6	30.1	33.4
IDEA Prune-2T (1.3B)	**8.88**	39.0	**94.6**	**54.0**	**31.1**	**46.4**

Compared to the 1.3B-2T scratch baseline, IDEA Prune is slightly better or comparable on most metrics (Table 1), confirming that carefully designed enlarge-and-prune pipelines can match or exceed direct training under the same token budget.

For controlled comparison against naive pipelines, the main 2.8B→1.3B experiment uses a matched 50/50 allocation (1T for enlarged model training, 1T for pruning and recovery). Proxy-scale ablations in Section 5.3 suggest that more recovery-heavy allocations (e.g., 30/70) can further improve IDEA Prune, but we do not claim the main-scale result is fully tuned. Even under this matched configuration, IDEA Prune achieves improvements over both training small models from scratch and naive pipelines.

Downstream Fine-Tuning. To examine whether the pretraining-stage gains persist after post-training, we fine-tune the pretrained checkpoints of Table 1 with the same supervised fine-tuning (SFT) recipe. Every checkpoint is fine-tuned on Dolci-Instruct-SFT [Team Olmo et al., 2025] (approximately 2M examples) for 2 epochs, with a cosine learning rate schedule from 2×10^{-5} to 1×10^{-8} and a batch size of 128. The data, recipe, and compute are identical for every row; only the starting checkpoint differs. We evaluate on MMLU [Hendrycks et al., 2020], GSM8K [Cobbe et al., 2021], IFEval [Zhou et al., 2023], and HumanEval [Chen et al., 2021].

Table 2 shows that IDEA Prune is the best 1.3B model on every benchmark under the same total pretraining budget: it matches or exceeds the scratch baseline and outperforms every naive pipeline, with the largest margin on HumanEval (+3.1 over the scratch baseline and +4.9 to +7.3 over naive pipelines). The ordering of methods before SFT (IDEA Prune $\geq$ scratch $>$ naive pipelines) is preserved after SFT, indicating that the pretraining-stage gains do not wash out under fine-tuning.

Table 2: Results after SFT. Each row is fine-tuned from the corresponding pretrained checkpoint in Table 1 with an identical recipe. IFEval reports prompt-level (P) and instruction-level (I) accuracy under loose and strict criteria. The best 1.3B results are in **bold**.

Initial Checkpoint	MMLU	GSM8K	IFEval				HumanEval
			P-loose	P-strict	I-loose	I-strict	
2.8B-1T from scratch	51.4	41.2	71.2	67.7	78.9	76.1	31.4
1.3B-2T from scratch	45.1	32.1	**66.5**	**62.5**	74.7	71.6	24.3
OSRP-2T (1.3B)	42.2	28.4	64.1	60.3	73.4	70.1	20.1
Minitron-2T (1.3B)	43.8	32.0	63.2	59.7	73.3	70.4	20.1
Sheared LLaMA-2T (1.3B)	44.2	30.3	63.8	61.5	75.0	71.7	22.5
IDEA Prune-2T (1.3B)	**46.0**	**33.0**	**66.5**	**62.5**	**76.0**	**72.8**	**27.4**

5 Analysis and Extensions

5.1 Analysis of Initialization and Learning Rate

If we only consider the pruning and recovery stages, the major differences of the integrated and naive enlarge-and-prune pipelines are twofold: the *initial weights* of the model before pruning and the *learning rate schedule*.

Initialization. At the beginning of the pruning stage, naive enlarge-and-prune pipelines initialize the model by the last checkpoint of an enlarged model trained with, e.g., 1T tokens. In contrast, IDEA Prune loads the model weights that are equivalent to the intermediate checkpoint of an enlarged model with a larger token budget, e.g., 2T tokens if we did not prune the enlarged model and continued to fully train the enlarged model. To ablate the impact of the initialization, we initialize the model by two checkpoints: the checkpoint of the 2.8B-2T model at the 1T-th step and the checkpoint of the 2.8B-1T at its last step.

Learning rate schedule. In IDEA Prune, we resume the learning rate schedule of the 2.8B-2T model at the 1T point, which provides a relatively small learning rate continuation, while naive enlarge-and-prune pipelines restart the cosine learning rate schedule with a linear warmup. We formalize these approaches into two learning rate schedule types: first, the *resumed* learning rate schedule, corresponding to the integrated enlarge-and-prune pipeline, mathematically represented by $\eta(t + T_l; T_l + T_p + T_r, \eta_p, \eta_e)$, where the learning rate continues from a previous training stage; and second, the *restarted* learning rate schedule, corresponding to the naive enlarge-and-prune pipeline, represented by $\eta(t; T_p + T_r, \eta_p, \eta_e)$, which initiates a new learning rate schedule from the beginning. We fix η_p, η_e in each setting.

Discussion 1. We present the results in Table 3. First, both the resumed learning rate schedule and the use of intermediate checkpoints are crucial components; omitting either leads to substantial degradation in MMLU. Intriguingly, we find that the learning rate schedule has a greater impact than initialization, with the resumed schedule consistently outperforming the restarted approach across different initialization methods.

Discussion 2. Contrary to conventional wisdom, our results challenge the presumption that better initialization directly translates to better pruned model performance. Most notably in the resumed

learning schedule, we discover that an intermediate checkpoint, despite showing lower performance than the final checkpoint, paradoxically provides a more favorable starting point for pruning, resulting in higher pruned model performance. This counterintuitive result stems from the learning dynamics: the final checkpoint's convergence at a low learning rate creates a mismatch with the resumed schedule's relatively higher learning rate, effectively mimicking a restarted schedule but with smaller peak learning rates. These findings emphasize the critical importance of aligning initialization with learning rate schedules in enlarge-and-prune pipelines.

Table 3: Ablation of the initialization and the learning rate schedule in enlarge-and-prune pipelines. 2.8B-2T@1T indicates the checkpoint at the 1T-token step of a 2.8B training with a 2T total token budget. Pruned model size: 1.3B.

Model	LR Schedule	OWT↓	Arc-C↑	SciQA↑	HellaSwag↑	TriviaQA↑	MMLU↑
2.8B-2T@1T	-	9.99	38.7	93.5	51.6	24.8	26.0
1.3B pruned from 2.8B-2T@1T	Resumed	**8.88**	39.0	94.6	**54.0**	**31.1**	**46.4**
	Restarted	8.94	39.7	93.7	54.5	30.1	37.0
2.8B-1T@1T	-	8.12	44.5	94.2	56.9	37.3	50.8
1.3B pruned from 2.8B-1T@1T	Resumed	8.89	39.3	93.5	53.6	31.8	38.2
	Restarted	8.95	39.9	92.9	53.3	31.0	36.8

5.2 Analysis of Enlarged Model Size

Unlike pruning existing enlarged models, enlarge-and-prune pipelines have more freedom on the choice of the enlarged model size. Therefore, we investigate the impact of enlarged model size on performance through the enlarge-and-prune pipelines. Starting with a 300M parameter baseline model featuring an FFN layer with 3×1024 hidden dimensions (detailed architecture in Appendix F.1), we explore enlarged models by incrementally increasing FFN width to $4\times$, $6\times$, $8\times$, $12\times$, $16\times$ of 1024, while keeping other model parameters constant. We conduct IDEA Prune using 500B tokens from DCLM, allocating 250B tokens to the enlarged model and 250B tokens to pruning and recovery.

Figure 4a illustrates our experiment results. Notably, integrated pipelines with all enlarged model sizes significantly outperform the training from scratch baseline. This robust performance across a wide range of model sizes demonstrates the flexibility of our integrated enlarge-and-prune pipeline and minimal need for model size tuning. Additionally, the results reveal an important trade-off between model capacity and pruning efficiency. Models with smaller FFN width enlargement factors (close to $1.3\times$) suffer from insufficient capacity to learn rich representations, while extremely large models (approaching $5.3\times$) face challenges of pruning degradation. Based on these observations, we identify $2.6\times$ as the optimal FFN width enlargement factor in the tested regime.

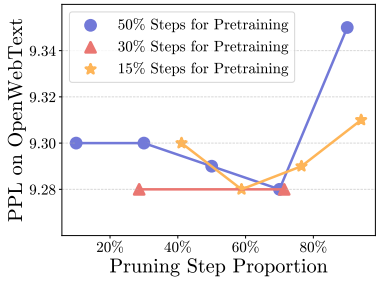


Figure 3: Pruning schedule.

5.3 Robustness of Hyperparameters

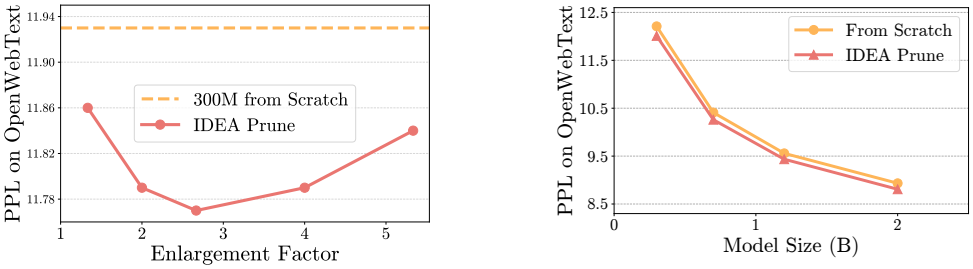
We study how the pruning schedule, importance score combination functions, and the moving average coefficient affect IDEA Prune; all are robust and require little tuning.

Pruning Schedule. Using a 400B token budget, we vary pretraining step proportions at 15%, 30%, and 50%, and examine different pruning steps T_p used in $r(t; T_p)$. Figure 3 reveals robust performance across a wide range of steps. For instance, with 50% steps in pretraining, valid pruning steps extend from 10% to 70%. The analysis suggests an optimal pruning proportion around 70%, indicating potential for further optimization beyond the 50/50 split used in our main experiments. Other hyperparameters—importance score combination (f_1, f_2) and moving average coefficient (λ)—are also robust; see Appendix G for details.

5.4 Scaling Property

We study whether the advantage of IDEA Prune over training from scratch holds across different target model sizes. We choose target sizes of 300M, 700M, 1.2B, and 2B with corresponding enlarged

models of 680M, 1.5B, 2.8B, and 4.2B. Each model is trained with 300B tokens from DCLM. As shown in Figure 4b, IDEA Prune outperforms the scratch baseline in perplexity across all target sizes within the on-device model range.



(a) Enlargement factor

(b) Target model size

Figure 4: **Left**: Ablation of the enlarged model size. **Right**: Scaling property of IDEA Prune.

5.5 Extension to Attention-Head and Joint Pruning

IDEA Prune is not restricted to FFN-width pruning. To test this, we apply IDEA Prune to attention-head pruning and to joint FFN-and-attention pruning at the 300M target scale, with a 300B-token budget for all models. For attention-only enlargement, we double the number of attention heads and KV heads while keeping the per-head dimension fixed, yielding enlarged models of 380M and 450M parameters. For joint enlargement, we scale both attention and FFN dimensions, yielding a 750M enlarged model (see Appendix F.1 for configurations). All models are pruned back to the 300M target architecture.

Table 4: Extension of IDEA Prune to attention-head and joint pruning at the 300M target scale. All models are trained with a 300B-token budget. "–" denotes the small model from scratch baseline.

Enlarged Dims Enlarged Size	– 300M	Attn (2×) 380M	Attn (3×) 450M	FFN 680M	Attn&FFN 750M
OWT Perplexity	12.21	12.18	12.20	11.99	11.98

As shown in Table 4, attention-only pruning provides a small improvement over training from scratch (12.18 vs. 12.21), while FFN-only pruning yields a substantially larger gain. Joint FFN-and-attention pruning (11.98) performs comparably to FFN-only, suggesting that the marginal benefit of attention pruning is small once FFN pruning is applied. These results support our choice of FFN-width as the primary pruning target in the tested 300M setting.

5.6 Combination with Knowledge Distillation

Knowledge distillation (KD) [Kim and Rush, 2016, Hinton et al., 2015] is complementary to pruning and is often combined with it in practice [Team et al., 2024, Liang et al., 2023, Gunter et al., 2024, Li et al., 2023]. Under our full-cost framework, we study whether KD can further improve the integrated enlarge-and-prune pipeline when the teacher model cost is included in the total token budget. We train a 2.8B-2T model as the teacher and establish a KD baseline by training a 1.3B model from scratch over 1T tokens with KD. In our pipeline, we prune and train the 2.8B-2T@1T intermediate checkpoint using a resumed cosine learning rate schedule for 1T tokens, with the 2.8B-2T model as the KD teacher. As shown in Table 5, pruning with distillation outperforms the pruning baseline across most benchmarks and surpasses or is on par with the KD baseline on all benchmarks, particularly on OpenWebText and MMLU.

Table 5: KD combined with IDEA Prune. Teacher: 2.8B-2T.

	Prune w/o KD	1.3B-1T w/ KD	Prune w/ KD
OWT	8.88	8.95	8.86
Arc-C	39.0	39.6	39.8
SciQA	94.6	93.9	93.8
HSwag	54.0	52.2	52.7
TQA	31.1	31.7	32.0
MMLU	46.4	44.5	46.6

5.7 Extension to Other Pruning Methods

To show the integrated pipeline is not tied to iterative pruning, we apply the integrated learning rate schedule to activation-based pruning (one-shot pruning of the 2.8B-2T@1T checkpoint using activation magnitudes; Appendix D) and learned mask pruning (mask learning from the same

checkpoint, with a resumed LR schedule for weights and a fully decaying schedule for mask parameters). Recovery uses the resumed schedule of Section 5.1.

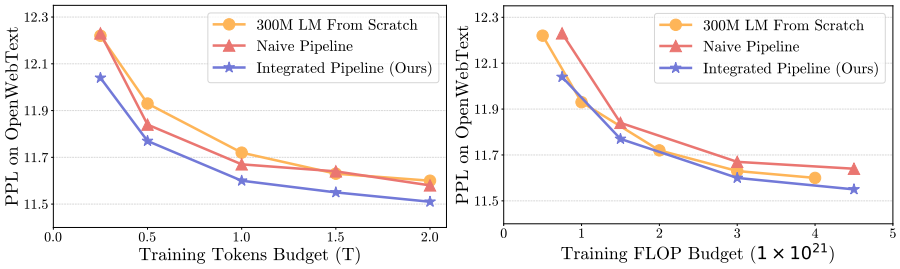
Table 6 shows the integrated pipeline consistently beats its naive counterpart across all pruning methods, with the largest gain on MMLU (9–13 points). This confirms that the LR restart in naive pipelines causes substantial knowledge loss regardless of criterion, and that any criterion can benefit from replacing the naive LR restarts with our unified schedule.

Table 6: Extension of the integrated pipeline to activation-based pruning and learned mask pruning, alongside OSRP. We prune a 2.8B model to 1.3B with 2T tokens. The integrated pipeline consistently outperforms the naive counterpart for every pruning criterion.

Method		OWT↓	Arc-C↑	SciQA↑	HellaSwag↑	TriviaQA↑	MMLU↑
OSRP	Naive	8.980	38.9	92.9	53.4	29.6	32.5
	Integrated	8.926	41.1	93.4	53.8	30.1	42.9
Activation-based	Naive	8.973	38.7	94.6	53.6	30.7	31.4
	Integrated	8.916	39.8	94.0	53.6	30.8	43.9
Learned mask	Naive	8.960	39.4	94.0	53.6	30.1	33.4
	Integrated	8.892	43.2	93.9	54.0	30.5	42.4

6 Discussion and Limitations

Tokens or FLOPs? Our main comparisons fix the training token budget, since tokens are the practical bottleneck in our regime. Pruning pipelines intrinsically consume more FLOPs than training the target-sized small model given the same training tokens. In wall-clock time, however, the step time of the 1.3B, 2.8B, and pruning runs is nearly identical in our setup (Appendix F.5). As shown in Figure 5, even under matched FLOPs, IDEA Prune still outperforms naive pipelines at all budgets and surpasses the scratch baseline at higher budgets, which is practical when practitioners have sufficient computational resources.



(a) Token Budget (b) FLOPs Budget

Figure 5: **Left**: Perplexity on OpenWebText of 300M model trained with multiple training token budgets from 0.25T to 2T. **Right**: FLOPs of each point from the left figure.

Significance and Reliability. The gains of IDEA Prune over training from scratch are modest in absolute terms but meaningful in a regime where small-model performance saturates quickly. To verify that these gains are not due to noise: (1) final-stage checkpoints of the 1.3B scratch model show minimal variance (Table 9), used here as a proxy for convergence stability in the 2.8B→1.3B regime, consistent with prior findings on convergence stability [van der Wal et al., 2025]; and (2) proxy-scale multi-seed runs (300M target, 3 seeds, 300B tokens) yield perplexities of 12.01/11.99/11.99 for IDEA Prune and 12.22/12.21/12.23 for the scratch baseline, confirming negligible variance.

Limitations inherent to large-scale pretraining. Pretraining experiments at the scales studied in this paper are computationally expensive, which inevitably constrains the scope of our empirical investigation: (1) the matched comparison between FFN-width and attention-head pruning is conducted at the 300M proxy scale (Section 5.5) but not at the main 2.8B→1.3B scale; (2) scaling validation covers four target sizes up to 2B (Section 5.4), but not beyond; (3) the main experiments are single-run, though proxy multi-seed experiments and convergence analysis (Table 9) suggest negligible variance; (4) larger source models are beyond our resources, as a single 2.8B→1.3B run already costs approximately 3.8×10^{22} FLOPs (Appendix F.5).

References

M. Abdin, S. A. Jacobs, A. A. Awan, J. Aneja, A. Awadallah, H. Awadalla, N. Bach, A. Bahree, A. Bakhtiari, H. Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, 2023.

Y. An, X. Zhao, T. Yu, M. Tang, and J. Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 10865–10873, 2024.

S. Ashkboos, M. L. Croci, M. G. d. Nascimento, T. Hoefler, and J. Hensman. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.

M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

C. Deng, L. Sun, J. Jiang, Y. Zeng, X. Wu, W. Zhao, Q. Xiao, J. Wang, H. Li, L. Chen, L. M. Ni, H. Zhang, and J. Wang. PLM: Efficient peripheral language models hardware-co-designed for ubiquitous computing. *arXiv preprint arXiv:2503.12167*, 2025.

X. Ding, X. Zhou, Y. Guo, J. Han, J. Liu, et al. Global sparse momentum sgd for pruning very deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

A. Dremov, A. Hägele, A. Kosson, and M. Jaggi. Training dynamics of the cooldown stage in warmup-stable-decay learning rate scheduler. *arXiv preprint arXiv:2508.01483*, 2025.

A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

A. Gokaslan, V. Cohen, E. Pavlick, and S. Tellex. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019.

T. Gunter, Z. Wang, C. Wang, R. Pang, A. Narayanan, A. Zhang, B. Zhang, C. Chen, C.-C. Chiu, D. Qiu, et al. Apple intelligence foundation language models. *arXiv preprint arXiv:2407.21075*, 2024.

J. Guo, X. Chen, Y. Tang, and Y. Wang. Slimllm: Accurate structured pruning for large language models. *arXiv preprint arXiv:2505.22689*, 2025.

D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

T. Jin, A. I. Humayun, U. Evci, S. Subramanian, A. Yazdanbakhsh, D. Alistarh, and G. K. Dziugaite. The journey matters: Average parameter count over pre-training unifies sparse and dense scaling laws. *arXiv preprint arXiv:2501.12486*, 2025.

M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

B.-K. Kim, G. Kim, T.-H. Kim, T. Castells, S. Choi, J. Shin, and H.-K. Song. Shortened llama: Depth pruning for large language models with comparison of retraining methods. *arXiv preprint arXiv:2402.02834*, 2024.

Y. Kim and A. M. Rush. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*, 2016.

A. Krishnakumar, R. S. Sukthanker, H. J. Mahadik, G. Kadlecová, V. Moroshan, T. Carstensen, F. Hutter, and A. Klein. Where to begin: Efficient pretraining via subnetwork selection and distillation. *arXiv preprint arXiv:2510.07227*, 2025.

W. Kwon, S. Kim, M. W. Mahoney, J. Hassoun, K. Keutzer, and A. Gholami. A fast post-training pruning framework for transformers. *Advances in Neural Information Processing Systems*, 35: 24101–24116, 2022.

M. Lee et al. Axlearn: Modular, hardware-agnostic large model training. `https://github.com/apple/axlearn`, 2024.

J. Li, A. Fang, G. Smyrnis, M. Ivgi, M. Jordan, S. Gadre, H. Bansal, E. Guha, S. Keh, K. Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. *arXiv preprint arXiv:2406.11794*, 2024.

Y. Li, Y. Yu, Q. Zhang, C. Liang, P. He, W. Chen, and T. Zhao. Losparse: Structured compression of large language models based on low-rank and sparse approximation. In *International Conference on Machine Learning*, pages 20336–20350. PMLR, 2023.

Z. Li, E. Wallace, S. Shen, K. Lin, K. Keutzer, D. Klein, and J. Gonzalez. Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *International Conference on machine learning*, pages 5958–5968. PMLR, 2020.

C. Liang, H. Jiang, Z. Li, X. Tang, B. Yin, and T. Zhao. Homodistil: Homotopic task-agnostic distillation of pre-trained transformers. *arXiv preprint arXiv:2302.09632*, 2023.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Z. Liu, C. Zhao, F. Iandola, C. Lai, Y. Tian, I. Fedorov, Y. Xiong, E. Chang, Y. Shi, R. Krishnamoorthi, et al. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. In *Forty-first International Conference on Machine Learning*, 2024.

Z. Liu, S. Xiao, B. Gao, Y. Liang, W. Zhang, H. Lin, G. Li, and J. Zhang. EfficientLLM: Scalable pruning-aware pretraining for architecture-agnostic edge language models. In *International Conference on Machine Learning (ICML)*, 2025.

I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

C. Louizos, M. Welling, and D. P. Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

A. Lozhkov, E. Bakouch, G. Martín Blázquez, G. Penedo, L. Tunstall, A. Marafioti, H. Kydlíček, A. Piqueres Lajarín, V. Srivastav, J. Lochner, C. Fahlgren, X.-S. Nguyen, C. Fourrier, B. Burtenshaw, H. Larcher, H. Zhao, C. Zakka, M. Morlon, C. Raffel, L. von Werra, and T. Wolf. SmolLM2: When smol goes big – data-centric training of a small language model. In *International Conference on Machine Learning (ICML)*, 2025.

K. Luo, H. Wen, S. Hu, Z. Sun, Z. Liu, M. Sun, K. Lyu, and W. Chen. A multi-power law for loss curve prediction across learning rate schedules. In *International Conference on Learning Representations*, 2025.

X. Ma, G. Fang, and X. Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

X. Men, M. Xu, Q. Zhang, B. Wang, H. Lin, Y. Lu, X. Han, and W. Chen. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*, 2024.

P. Michel, O. Levy, and G. Neubig. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32, 2019.

P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11264–11272, 2019.

S. Muralidharan, S. T. Sreenivas, R. Joshi, M. Chochowski, M. Patwary, M. Shoeybi, B. Catanzaro, J. Kautz, and P. Molchanov. Compact language models via pruning and knowledge distillation. *arXiv preprint arXiv:2407.14679*, 2024.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

K. Sakaguchi, R. Le Bras, C. Bhagavatula, and Y. Choi. WinoGrande: An adversarial Winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

A. Talmor, J. Herzig, N. Lourie, and J. Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4149–4158, 2019.

G. Team, P. Georgiev, V. I. Lei, R. Burnell, L. Bai, A. Gulati, G. Tanzer, D. Vincent, Z. Pan, S. Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Team Olmo, A. Ettinger, A. Bertsch, B. Kuehl, D. Graham, D. Heineman, D. Groeneveld, F. Brahman, F. Timbers, H. Ivison, J. Morrison, J. Poznanski, K. Lo, L. Soldaini, M. Jordan, M. Chen, M. Noukhovitch, N. Lambert, P. Walsh, P. Dasigi, R. Berry, S. Malik, S. Shah, S. Geng, S. Arora, S. Gupta, T. Anderson, T. Xiao, T. Murray, T. Romero, V. Graf, A. Asai, A. Bhagia, A. Wettig, A. Liu, A. Rangapur, C. Anastasiades, C. Huang, D. Schwenk, H. Trivedi, I. Magnusson, J. Lochner, J. Liu, L. J. V. Miranda, M. Sap, M. Morgan, M. Schmitz, M. Guerquin, M. Wilson, R. Huff, R. L. Bras, R. Xin, R. Shao, S. Skjonsberg, S. Z. Shen, S. S. Li, T. Wilde, V. Pyatkin, W. Merrill, Y. Chang, Y. Gu, Z. Zeng, A. Sabharwal, L. Zettlemoyer, P. W. Koh, A. Farhadi, N. A. Smith, and H. Hajishirzi. Olmo 3, 2025. URL `https://arxiv.org/abs/2512.13961`.

H. Tissue, V. Wang, and L. Wang. Scaling law with learning rate annealing. *arXiv preprint arXiv:2408.11029*, 2024.

O. van der Wal, P. Lesci, M. Müller-Eberstein, N. Saphra, H. Schoelkopf, W. Zuidema, and S. Biderman. Polypythias: Stability and outliers across fifty language model pre-training runs. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR 2025)*, pages 1–25, 2025.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, 2019.

Z. Wang, J. Wohlwend, and T. Lei. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*, 2019.

J. Wei, Q. Lu, N. Jiang, S. Li, J. Xiang, J. Chen, and Y. Liu. Structured optimal brain pruning for large language models. In Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13991–14007, Miami, Florida, USA, Nov. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.775. URL https://aclanthology.org/2024.emnlp-main.775/.

J. Welbl, N. F. Liu, and M. Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark, 2017. Association for Computational Linguistics.

K. Wen, Z. Li, J. Wang, D. Hall, P. Liang, and T. Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective. In *International Conference on Learning Representations*, 2025.

M. Xia, Z. Zhong, and D. Chen. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*, 2022.

M. Xia, T. Gao, Z. Zeng, and D. Chen. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, 2023.

Z. Xu, Y. Xu, H. Xu, Y. Liao, Z. Yao, and Z. Xie. Lightweight and post-training structured pruning for on-device large language models. *arXiv preprint arXiv:2501.15255*, 2025.

G. Yang, E. J. Hu, I. Babuschkin, S. Sidor, X. Liu, D. Farhi, N. Ryder, J. Pachocki, W. Chen, and J. Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022.

R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Q. Zhang, S. Zuo, C. Liang, A. Bukharin, P. He, W. Chen, and T. Zhao. Platon: Pruning large transformer models with upper confidence bound of weight importance. In *International conference on machine learning*, pages 26809–26823. PMLR, 2022.

P. Zhao, H. Hu, P. Li, Y. Zheng, Z. Wang, and X. Yuan. FISTAPruner: Layer-wise post-training pruning for large language models. In C. Christodoulopoulos, T. Chakraborty, C. Rose, and V. Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 29469–29487, Suzhou, China, Nov. 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1498. URL https://aclanthology.org/2025.emnlp-main.1498/.

J. Zhou, T. Lu, S. Mishra, S. Brahma, S. Basu, Y. Luan, D. Zhou, and L. Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

M. Zhu and S. Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

A Model Optimization

A.1 AdamW Optimizer

We use the AdamW optimizer [Loshchilov and Hutter, 2019] with weight decay. Given a weight matrix $\boldsymbol{W}^{(t-1)}$ at the t-th step, the weight update is

$$\boldsymbol{W}^{(t)} = (1 - \gamma^{(t)}\lambda_{\text{wd}})\boldsymbol{W}^{(t-1)} - \gamma^{(t)}\widetilde{\boldsymbol{G}}^{(t)}, \tag{A.1}$$

where $\gamma^{(t)}$ is the learning rate, λ_{wd} is the decoupled weight decay coefficient, and $\widetilde{\boldsymbol{G}}^{(t)}$ is calculated as

$$\widetilde{\boldsymbol{G}}^{(t)} = \frac{\widetilde{\boldsymbol{M}}^{(t)}}{\sqrt{\widetilde{\boldsymbol{V}}^{(t)}} + \epsilon},$$

where ϵ is a hyperparameter, and $\widetilde{\boldsymbol{M}}^{(t)}, \widetilde{\boldsymbol{V}}^{(t)}$ are determined by $\boldsymbol{G}^{(t-1)}$, the gradient of $\boldsymbol{W}^{(t-1)}$, and the optimization states $\boldsymbol{M}^{(t-1)}, \boldsymbol{V}^{(t-1)}$. Specifically, we update $\boldsymbol{M}^{(t-1)}, \boldsymbol{V}^{(t-1)}$ by

$$\boldsymbol{M}^{(t)} = \beta_1 \boldsymbol{M}^{(t-1)} + (1-\beta_1)\boldsymbol{G}^{(t-1)},$$
$$\boldsymbol{V}^{(t)} = \beta_2 \boldsymbol{V}^{(t-1)} + (1-\beta_2)\boldsymbol{G}^{(t-1)} \odot \boldsymbol{G}^{(t-1)},$$

where β_1 and β_2 are hyperparameters. Then, $\widetilde{\boldsymbol{M}}^{(t)}$ and $\widetilde{\boldsymbol{V}}^{(t)}$ are calculated as

$$\widetilde{\boldsymbol{M}}^{(t)} = \boldsymbol{M}^{(t)}/(1-\beta_1^t),$$
$$\widetilde{\boldsymbol{V}}^{(t)} = \boldsymbol{V}^{(t)}/(1-\beta_2^t).$$

A.2 Cosine Annealing Learning Rate Schedule

The learning rate $\gamma^{(t)}$ in (A.1) is scheduled by a cosine annealing with linear warm-up [Loshchilov and Hutter, 2016]. Given a total of T training steps, the schedule is

$$\eta(t; T, \eta_p, \eta_e) = \begin{cases} \frac{t}{T_0}\eta_p & \text{if } 0 < t \le T_0, \\ c_1 \cos\left(\frac{t-T_0}{T-T_0}\pi\right) + c_2 & \text{if } T_0 < t \le T, \end{cases} \tag{A.2}$$

where $c_1 = \frac{1}{2}(\eta_p - \eta_e), c_2 = \frac{1}{2}(\eta_p + \eta_e)$, T_0 represents the number of warmup steps, and η_p and η_e are the peak and end learning rates, respectively.

B IDEA Prune Algorithm

Algorithm 1 IDEA Prune

Require: Sparsity schedule $r(t; T_p)$, learning rate schedule $\eta(t; T, \eta_p, \eta_e)$, pruning interval K, initial FFN width h.

1: Randomly initialize model weights $\boldsymbol{W}^{(0)}$.
2: Zero initialize approximate importance scores $\widetilde{\boldsymbol{S}}^{(0)}$; set living neurons $\mathcal{A} \leftarrow \{1, \ldots, h\}$.
3: **for** t = 1 to T **do**
4: Compute gradients $\nabla l_i(\boldsymbol{W}^{(t-1)})$ on a mini batch.
5: Compute learning rate $\alpha^{(t)} \leftarrow \eta(t; T, \eta_p, \eta_e)$.
6: Update weights $\boldsymbol{W}^{(t)}$ following (A.1) with $\gamma^{(t)} = \alpha^{(t)}$.
7: Update importance scores $\widetilde{\boldsymbol{S}}^{(t)}$ following (3.1).
8: **if** $t \bmod K = 0$ and $h - |\mathcal{A}| < \lfloor r(t; T_p)\, h \rfloor$ **then**
9: Compute neuron importance $\boldsymbol{c}_k^{(t)}$ for $k \in \mathcal{A}$ following (3.2).
10: $k^* \leftarrow \operatorname{argmin}_{k \in \mathcal{A}} \boldsymbol{c}_k^{(t)}$; $\mathcal{A} \leftarrow \mathcal{A} \setminus \{k^*\}$.
11: Permanently remove neuron k^*: set $\boldsymbol{m}_{k^*} = 0$ in (2.1) and drop the k^*-th rows of $\boldsymbol{W}_{\text{up}}, \boldsymbol{W}_{\text{gate}}, \boldsymbol{W}_{\text{down}}$.
12: **end if**
13: **end for**

Ensure: $\boldsymbol{W}^{(T)}$

Pruning and recovery are two phases of the same training loop and differ only in whether neuron removal (lines 8–12) fires. During enlarged pretraining ($t \le T_l$), $r(t; T_p) = 0$, so no neuron is removed. During pruning, one neuron is removed every K steps, so that the number of removed neurons tracks the cubic sparsity schedule $r(t; T_p)$ (Appendix C.2); the discrete one-neuron-every-K-steps cadence realizes this continuous schedule given that the neuron count is an integer. During recovery, the target width has been reached, so no neuron is removed. The weight update runs identically in all three phases, and the optimizer state and the single cosine learning rate schedule carry through without any restart. Removal is permanent rather than a soft mask: a removed neuron no longer exists in the compacted dense weight matrices, so it cannot regain a score or re-enter later. In the main experiment, enlarged pretraining, pruning, and recovery span 1T, 0.5T, and 0.5T tokens, respectively.

C Iterative Pruning

C.1 Sensitivity Score

For a model with parameters $\boldsymbol{\theta}^{(t)} \in \mathbb{R}^N$, we measure parameter importance by sensitivity [Molchanov et al., 2019]:

$$\boldsymbol{\omega}^{(t)} = \left| \nabla \mathcal{L}(\boldsymbol{\theta}^{(t)}) \odot \boldsymbol{\theta}^{(t)} \right|, \tag{C.1}$$

where $\mathcal{L}(\cdot)$ is the loss function. This score represents the first-order approximation of each parameter's impact on the loss: $\boldsymbol{\omega}^{(t)} \approx \left| \mathcal{L}(\boldsymbol{\theta}^{(t)}) - \mathcal{L}(\boldsymbol{\theta}^{(t)}_{-k}) \right|$, where $\boldsymbol{\theta}^{(t)}_{-k} = (\theta_1, \ldots, \theta_{k-1}, 0, \theta_{k+1}, \ldots, \theta_N)$.

C.2 Cubic Sparsity Schedule in Iterative Pruning

The sparsity $r(t; T_p)$ (fraction of neurons removed) at the t-th step increases cubically from 0 to the target sparsity R:

$$r(t; T_p) = \begin{cases} 0 & 0 < t \le T_w, \\ R\left(1 - \left(1 - \dfrac{t - T_w}{T_p}\right)^3\right) & T_w < t \le T_w + T_p, \\ R & T_w + T_p < t \le T, \end{cases}$$

where R is the target sparsity (fraction of neurons removed), and T_w, T_p, T are the number of pruning warmup steps, iterative pruning steps, and total training steps, respectively. The schedule starts at $r = 0$ (full model retained), smoothly increases to $r = R$ (target sparsity), and stays at R during recovery. We set $T_w = T_l$ for integrated enlarge-and-prune pipelines, and equal to the learning rate warmup steps for naive enlarge-and-prune pipelines.

D Minitron: Activation-Based Pruning

We denote the activation of the n-th data sample of an FFN layer as

$$\boldsymbol{Z}^{(n)} = \sigma(\boldsymbol{X}^{(n)} \boldsymbol{W}_{\text{up}}^\top) \odot (\boldsymbol{X}^{(n)} \boldsymbol{W}_{\text{gate}}^\top),$$

where $\boldsymbol{Z}^{(n)} \in \mathbb{R}^{l \times h}$, l is the sequence length. The importance score of the i-th neuron is defined as

$$\boldsymbol{c}_i = \frac{1}{B} \sum_{n=1}^{B} \| \boldsymbol{Z}^{(n)}_{[:,i]} \|_2,$$

where B is the size of the calibration dataset (a random subset of the pre-training corpus), $\boldsymbol{Z}^{(n)}_{[:,i]}$ is the i-th column of $\boldsymbol{Z}^{(n)}$, and $|| \cdot ||_2$ is the L2-norm of a vector. Following Muralidharan et al. [2024], we set $B = 1024$. Finally, we apply (3.3) with $T_p = 1$ to generate the pruning mask.

E Structured Pruning of Attention Heads

We describe structured pruning of attention heads in parallel with Section 2.2 for FFN width. We consider a multi-head attention layer with Grouped Query Attention (GQA) [Ainslie et al., 2023]. Given an input $\boldsymbol{X} \in \mathbb{R}^{l \times d}$, the layer computes n_q query heads and n_{kv} key-value heads:

$$\begin{aligned} \boldsymbol{Z}_j &= \text{Attention}(\boldsymbol{X}\boldsymbol{W}_{Q_j}^\top,\ \boldsymbol{X}\boldsymbol{W}_{K_{g(j)}}^\top,\ \boldsymbol{X}\boldsymbol{W}_{V_{g(j)}}^\top), \quad j = 1, \ldots, n_q, \\ \text{MHA}(\boldsymbol{X}) &= [\boldsymbol{Z}_1; \ldots; \boldsymbol{Z}_{n_q}]\boldsymbol{W}_O, \end{aligned}$$

where $\boldsymbol{W}_{Q_j} \in \mathbb{R}^{d_h \times d}$ is the query projection for the j-th head, $\boldsymbol{W}_{K_{g(j)}}, \boldsymbol{W}_{V_{g(j)}} \in \mathbb{R}^{d_h \times d}$ are the shared key and value projections for KV group $g(j)$, d_h is the per-head dimension, $\boldsymbol{W}_O \in \mathbb{R}^{d \times n_q d_h}$ is the output projection, and $g : \{1, \ldots, n_q\} \to \{1, \ldots, n_{kv}\}$ maps each query head to its KV group.

Head importance scoring. Following the sensitivity-based approach in Section 3.2, we compute the importance of each query head j by aggregating element-wise sensitivity scores over its query projection matrix:

$$\boldsymbol{c}_j^{(t)} = f_1\left(\widetilde{\boldsymbol{S}}_{Q_j}^{(t)}\right), \tag{E.1}$$

where $\widetilde{\boldsymbol{S}}_{Q_j}^{(t)} \in \mathbb{R}^{d_h \times d}$ is the moving-average sensitivity score matrix for $\boldsymbol{W}_{Q_j}$ (computed as in (3.1)), and $f_1 : \mathbb{R}^{d_h \times d} \to \mathbb{R}$ aggregates element-wise scores to a scalar (e.g., $\max$ or mean).

Pruning mask and weight compression. We define a binary mask $\boldsymbol{m}_{\text{attn}} \in \{0,1\}^{n_q}$ over query heads using the same sparsity schedule as (3.3):

$$\boldsymbol{m}_{\text{attn},j}^{(t)} = \begin{cases} 1 & \text{if } \boldsymbol{c}_j^{(t)} \text{ is in top } (1 - r(t; T_p)) \text{ fraction of } \boldsymbol{c}^{(t)}, \\ 0 & \text{otherwise.} \end{cases}$$

A KV group k is retained if at least one of its associated query heads is retained:

$$\boldsymbol{m}_{\text{kv},k}^{(t)} = \max_{j:\, g(j)=k} \boldsymbol{m}_{\text{attn},j}^{(t)}.$$

The pruned weight matrices are obtained by selecting the rows/columns corresponding to retained heads:

$$\begin{aligned} \boldsymbol{W}_{Q_j}^{'} &= \boldsymbol{W}_{Q_j} \cdot \boldsymbol{m}_{\text{attn},j}^{(t)}, \quad j = 1, \ldots, n_q, \\ \boldsymbol{W}_{K_k}^{'} &= \boldsymbol{W}_{K_k} \cdot \boldsymbol{m}_{\text{kv},k}^{(t)}, \quad \boldsymbol{W}_{V_k}^{'} = \boldsymbol{W}_{V_k} \cdot \boldsymbol{m}_{\text{kv},k}^{(t)}, \quad k = 1, \ldots, n_{kv}, \\ \boldsymbol{W}_{O}^{'} &= \boldsymbol{W}_{O}[:,\ \{j \cdot d_h, \ldots, (j{+}1)d_h - 1\}_{j:\boldsymbol{m}_{\text{attn},j}=1}]. \end{aligned}$$

After removing masked-out rows and columns, the pruned layer has $n_q^{'}$ query heads and $n_{kv}^{'} \leq n_{kv}$ KV groups, where $n_q^{'} = \|\boldsymbol{m}_{\text{attn}}\|_1$. The attention hidden dimension reduces from $n_q d_h$ to $n_q^{'} d_h$ while keeping the per-head dimension d_h fixed, preserving the model's positional encoding and layer structure.

In our experiments (Section 5.5), we enlarge the number of query and KV heads by a factor of $2\times$ or $3\times$ and prune back to the target head count using iterative importance-based pruning as described above.

F Detailed Experiment Setups

F.1 Model Architectures

We present the key configurations of the models in Table 7.

Table 7: Model configurations. # Params is non-embedding parameter count. Per-head dimension is 64 for all models. For GQA, # KV Heads gives the shared key/value group count. Attention hidden size equals # Q Heads $\times$ Head Dim.

Model	# Params	Input Length	Model Dim	# Q Heads	# KV Heads	Head Dim	Attn Hidden	FFN Hidden	# Layers
300m	302M	4096	1024	16	8	64	1024	3072	24
300m-attn2x	378M	4096	1024	32	16	64	2048	3072	24
300m-attn3x	453M	4096	1024	48	24	64	3072	3072	24
300m-ffn8x	679M	4096	1024	16	8	64	1024	8192	24
300m-f8x-a2x	755M	4096	1024	32	16	64	2048	8192	24
700m	717M	4096	1536	24	8	64	1536	5120	24
700m-ffn8x	1.51B	4096	1536	24	8	64	1536	12288	24
1b	1.21B	4096	2048	32	8	64	2048	6528	24
1b-ffn8x	2.67B	4096	2048	32	8	64	2048	16384	24
2b	1.98B	4096	2560	40	8	64	2560	8704	24
2b-ffn8x	4.15B	4096	2560	40	8	64	2560	20480	24
3b	2.91B	4096	3072	48	8	64	3072	10752	24
3b-ffn8x	5.96B	4096	3072	48	8	64	3072	24576	24

F.2 Datasets

We train all models on DCLM [Li et al., 2024], an open-source pretraining corpus containing 4T unique tokens of diverse domains.

F.3 Baselines

We compare against two categories of baselines:

(i) Training from scratch. We train the target-size model (1.3B) from scratch for 1T and 2T tokens.

(ii) Naive enlarge-and-prune pipelines. The 2.8B enlarged model is pretrained for 1T tokens, pruned to 1.3B using one of the following techniques, and then continually pretrained for another 1T tokens.

• *One-shot random pruning (OSRP)* randomly generates a pruning mask satisfying the target sparsity in a single step.

• *Minitron* computes the pruning mask based on FFN activations in a single pass (see Appendix D). We do not apply the distillation introduced in its original method.

• *Sheared LLaMA* [Xia et al., 2023] learns the pruning mask via auxiliary parameters and modified training objectives. The pruning process consumes 100B tokens. We do not apply the dynamic batch loading introduced in its original method.

F.4 Training Hyperparameters

All experiments use the AdamW optimizer [Loshchilov and Hutter, 2019] with a sequence length of 4096 and a batch size of 1024, regardless of model size or pruning method. Peak and end learning rates follow optimal hyperparameter transfer [Yang et al., 2022]. Table 8 lists the learning rate schedule and weight decay for each configuration.

Table 8: Training hyperparameters. All configurations use AdamW with a cosine schedule and linear warmup.

Configuration	Peak LR	End LR	Warmup Steps	Weight Decay
300m from scratch	7.5e-3	3.75e-5	5000	3.16e-4
300m enlarged (302m8x)	7.5e-3	3.75e-5	5000	3.16e-4
302m-IDEA Prune (302m8x)	7.5e-3	3.75e-5	5000	3.16e-4
700m from scratch	5.0e-3	2.5e-5	5000	3.16e-4
700m enlarged (700m8x)	5.0e-3	2.5e-5	5000	3.16e-4
700m-IDEA Prune (700m8x)	5.0e-3	2.5e-5	5000	3.16e-4
1b from scratch	3.75e-3	1.88e-5	5000	3.16e-4
1b enlarged (1b8x)	3.75e-3	1.88e-5	5000	3.16e-4
1b-IDEA Prune (1b8x)	3.75e-3	1.88e-5	5000	3.16e-4
2b from scratch	3.0e-3	1.5e-5	5000	3.16e-4
2b enlarged (2b8x)	3.0e-3	1.5e-5	5000	3.16e-4
2b-IDEA Prune (2b8x)	3.0e-3	1.5e-5	5000	3.16e-4
3b from scratch	2.5e-3	1.25e-5	5000	3.16e-4
3b enlarged (3b8x)	2.5e-3	1.25e-5	5000	3.16e-4
3b-IDEA Prune (3b8x)	2.5e-3	1.25e-5	5000	3.16e-4

F.5 Hardware and Compute

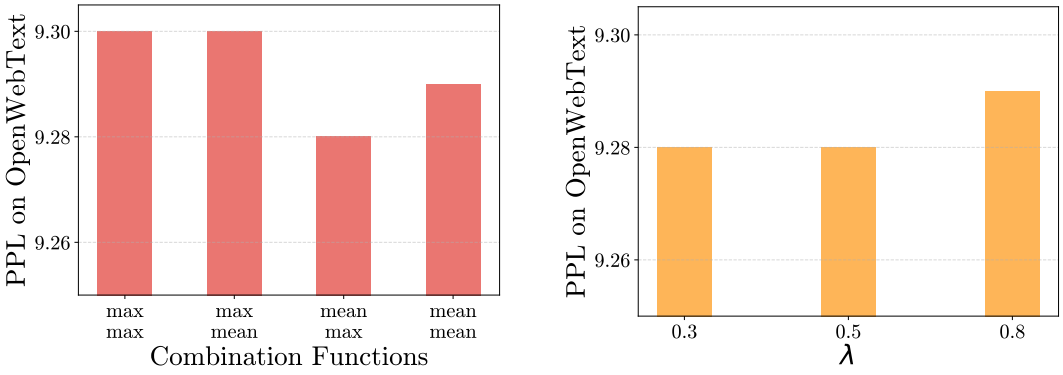
All experiments are conducted on Google Cloud v5p TPU accelerators using the AXLearn framework [Lee et al., 2024]. Training the 2.8B model for 1T tokens consumes approximately 2.5×10^{22} FLOPs. The total compute for the main IDEA Prune experiment (2T tokens end-to-end) is approximately 3.8×10^{22} FLOPs, comparable to the naive enlarge-and-prune baselines which consume 2.5×10^{22} (enlarged pretraining) + 1.3×10^{22} (recovery) FLOPs.

Wall-clock time. On TPU v5p-512 with the batch size fixed at 1024 across settings, the step time is 0.60s for training the 1.3B model, 0.58s for training the 2.8B model, and 0.60s for pruning 2.8B to 1.3B, each with a 2T-token budget. At this scale, a training step is bottlenecked by fixed memory and communication overhead rather than compute, so the wider FFN of the 2.8B model fills otherwise idle matrix-unit cycles (MFU roughly doubles) and its extra FLOPs are nearly free. Hence, tokens are a fair cost metric in our setting. We note that MFU is not matched across settings, and that wall-clock time depends on chip type, kernel implementation, communication pattern, batch size, and sequence length; we therefore report tokens and FLOPs as the primary cost metrics, following Muralidharan et al. [2024], Xia et al. [2023], Li et al. [2020]. Since pruning removes whole neurons, the compacted weight matrices are ordinary dense matrices of smaller width, requiring no masking or gather/scatter operations.

G Additional Hyperparameter Ablations

Importance Score Combination. We study how the choice of importance score combination function $f_1(\cdot)$ and $f_2(\cdot)$ in (3.2) affects model performance. We try $\max(\cdot)$ and $\text{mean}(\cdot)$ for each of $f_1(\cdot)$ and $f_2(\cdot)$. As shown in Figure 6a, "mean-max" is the best choice, though the others do not show significant degradation.

Moving Average Coefficient. We investigate the impact of the coefficient λ in (3.1), exploring values of 0.3, 0.5, and 0.8. As shown in Figure 6b, the optimal λ lies between 0.3 and 0.5, with 0.8 also acceptable.



(a) Importance Combination

(b) EMA Coefficient λ

Figure 6: Additional hyperparameter ablations. "Mean-max" is the best choice of f_1 and f_2. The EMA coefficient λ is robust from 0.3 to 0.8.

H Results of Final Stage

To assess convergence stability, we evaluate the 1.3B model trained from scratch (1T tokens) at the final three checkpoints—corresponding to 0.96T, 0.98T, and 1.00T training tokens. As shown in Table 9, the results are remarkably consistent across checkpoints, with variations typically within 0.2 points. This confirms that the training has entered the convergence regime and that performance noise at this stage is minimal. We use these scratch-model results as a proxy to demonstrate convergence stability in the 2.8B→1.3B setting; analogous final-stage results for IDEA Prune are not available.

Table 9: Convergence stability of the 1.3B model trained from scratch for 1T tokens, evaluated at the final three checkpoints. Results for IDEA Prune at this stage are not available; these scratch-model results serve as a proxy for the 2.8B→1.3B setting.

Eval at (tokens)	**OWT ↓**	**Arc-C ↑**	**HellaSwag ↑**	**TriviaQA ↑**	**MMLU↑**
0.96T	9.10	39.59	52.84	30.03	28.72
0.98T	9.10	38.40	52.69	30.56	29.12
1.00T	9.10	39.76	52.81	30.33	28.95
Average	9.10	39.25	52.78	30.31	28.93

I Additional Benchmarks

We further evaluate the 1.3B models of Table 1 on Arc-Easy [Clark et al., 2018], WinoGrande [Sakaguchi et al., 2021], and CommonsenseQA (CSQA) [Talmor et al., 2019]. As shown in Table 10, IDEA Prune is better than or on par with every baseline. Similar to MMLU in Table 1, naive pipelines fall far behind on CSQA, while IDEA Prune matches the scratch baseline.

Table 10: Accuracy(↑) of 1.3B models on additional benchmarks. All models use a 2T-token budget.

Method	Arc-Easy	WinoGrande	CSQA
1.3B-2T from scratch	**73.8**	66.7	50.04
OSRP-2T (1.3B)	72.8	67.0	25.88
Minitron-2T (1.3B)	73.1	**67.4**	26.70
Sheared LLaMA-2T (1.3B)	73.4	66.8	27.02
IDEA Prune-2T (1.3B)	73.2	**67.4**	**50.45**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the scope (deployment-constrained small language models trained from scratch) and the specific contributions (enlarge-and-prune framework, IDEA Prune pipeline, controlled experiments). Assumptions about the training regime are explicitly described in Section 1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 discusses limitations including the focus on FFN-width pruning, the scale of experiments, and the single-run nature of the main experiments.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.

- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not contain theoretical results or proofs. All contributions are empirical.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide model architectures (Appendix F.1), hyperparameters (Appendix F.4), the algorithm (Algorithm 1), training details, and the publicly available dataset (DCLM). Hardware details are provided in Appendix F.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We do not release code at submission time. The training dataset (DCLM) is publicly available. All algorithmic details are provided in the paper to enable reimplementation.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 and Appendix F.4 specify model architectures, learning rates, batch size, sequence length, and token allocations. Hyperparameter choices follow optimal hyperparameter transfer [Yang et al., 2022].

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The main experiments are single-run due to the high cost of pretraining at the 2.8B scale. We provide proxy multi-seed results at a smaller scale (Section 6) and convergence analysis (Appendix) to support result stability.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 4 specifies Google Cloud v5p TPU accelerators and the AXLearn framework. FLOPs estimates are provided in Appendix F.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research studies training efficiency for language models using publicly available data. We have reviewed and conform to the NeurIPS Code of Ethics.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper studies training methodology for language models and does not release models or datasets. The societal impact is indirect and not specific to this work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release models or datasets. The paper studies training methodology.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: DCLM [Li et al., 2024], the SFT dataset Dolci-Instruct-SFT [Team Olmo et al., 2025], and all evaluation benchmarks are properly cited. DCLM is released under the CC BY 4.0 license, and Dolci-Instruct-SFT under the ODC-BY license.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not release new datasets, models, or code.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or human subjects research.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as a component of the core methodology. Any use of LLMs for writing assistance does not impact the scientific content.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.