Adapt-Pruner: Adaptive Structural Pruning for Efficient Small Language Model Training

Anonymous ACL submission

Abstract

Small language models (SLMs) have attracted considerable attention from both academia and industry due to their broad range of applications in edge devices. To obtain SLMs with strong performance, conventional approaches either pre-train the models from scratch, which incurs substantial computational costs, or compress/prune existing large language models (LLMs), which results in performance drops and falls short in comparison to pre-training. In this paper, we investigate the family of acceleration methods that involve both structured pruning and model training. We found 1) layer-wise adaptive pruning (Adapt-Pruner) is quite effective in LLMs and yields significant improvements over existing pruning techniques, 2) adaptive pruning equipped with further training leads to models comparable to those pre-training from scratch, 3) incremental pruning brings non-trivial performance gain by interleaving pruning with training and only removing a small portion of neurons ($\sim$5%) at a time. Experimental results on LLaMA-3.1-8B demonstrate that Adapt-Pruner outperforms conventional pruning methods, such as LLM-Pruner, FLAP, and SliceGPT, by an average of 1%-7% in accuracy on commonsense benchmarks. Additionally, Adapt-Pruner restores the performance of MobileLLM-125M to 600M on the MMLU benchmark with 200$\times$ fewer tokens via pruning from its larger counterparts, and discovers a new 1B model that surpasses LLaMA-3.2-1B-Instruct in multiple common-sense reasoning benchmarks.

1 Introduction

Large language models (LLMs) (Kalyan, 2024; OpenAI, 2023) have demonstrated remarkable performance across a wide range of benchmarks. As their size increases, these models exhibit enhanced capabilities in understanding natural language and solving complex tasks through text generation (Zhao et al., 2023). However, achieving such performance requires models with billions of parameters, which presents significant challenges for practical deployment. The sheer scale of LLMs leads to high computational costs, making inference both resource-intensive and slow, and potentially introducing issues such as increased latency. Consequently, there is a growing demand for methods to compress LLMs (Zhu et al., 2024), aiming to reduce the number of parameters and improve inference speed, all while preserving the original model performance. Effective compression techniques hold the potential to create more efficient and deployable LLMs.

Several techniques have been proposed to compress LLMs, most of which fall into one of four categories: structured and unstructured pruning (Cheng et al., 2024), quantization (Gholami et al., 2022), low-rank factorization (Sainath et al., 2013), and knowledge distillation (Gou et al., 2021). In this paper, we primarily focus on structured pruning, which can be combined together with further training to obtain strong Small Language Models. Structured pruning removes entire filters, layers, or specific model components from neural networks, enabling both compression and realistic acceleration. On top of that, it does not require specialized hardware or library support to achieve these benefits (He and Xiao, 2023) in contrast to unstructured pruning.

While many works on structured pruning focus on removing a fixed number of filters from weight matrices with minimal performance degradation, these methods often either skip important layers or apply uniform sparsity across all layers. However, as shown in Figure 1, the importance of each decoder layer—and by extension, each weight matrix—varies significantly. To leverage this phenomenon, we introduce a novel approach called Adapt-Pruner. Unlike traditional pruning methods that enforce the same sparsity across all decoder layers, Adapt-Pruner operates in multiple steps. At

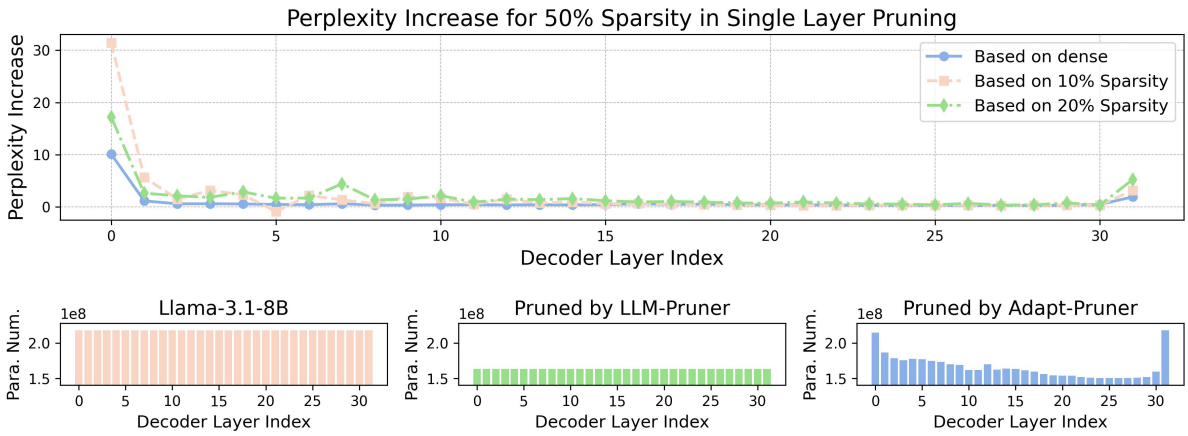


Figure 1: Layer sensitivity and pruned Models. The first row of figures shows the increase in perplexity when a single decoder layer is pruned at 50% sparsity, compared to the dense LLaMA-3.1-8B model, as well as models uniformly pruned across all layers at 10% and 20% sparsity. The second row of figures illustrates the architecture of the pruned models, with each decoder layer represented by its corresponding number of parameters.

each step, it calculates the relative importance of each decoder layer and applies varying sparsity levels, assigning higher sparsity to less important layers and lower sparsity to more critical ones. After determining the sparsity for each layer, the importance of each weight group is accessed using both magnitude and first-order information, leaving the least important groups for pruning.

Furthermore, when computational resources allow, structural pruning methods can be combined together with additional post-training stages, recovering the performance hurt by pruning. Multiple approaches (Xia et al., 2024; Sreenivas et al., 2024) have investigated this fashion of training to obtain SLMs efficiently. Nonetheless, their methods fixate too much on the prune-the-train paradigm, which hinders them from further boosting the target SLM's performance. In contrast, a different paradigm is explored in this paper, offering new opportunities for structural-pruning-based acceleration methods.

Our main contributions are summarized as follows:

1. A strong structured pruning method called Adapt-Pruner is proposed, which exploits the skewed importance distribution across LLM layers and significantly outperforms conventional pruning methods in commonsense benchmarks.

2. A new acceleration paradigm called Adapt-Accel is presented, which interleaves the pruning with training in a highly frequent manner and demonstrates non-trivial improvements compared to past methods (Xia et al., 2024; Sreenivas et al., 2024).

3. A novel family of models called Adapt-LLMs, which is obtained through Adapt-Accel, achieves superior performance over strong open-sourced models. In particular, Adapt-Pruner recovers the performance of MobileLLM (Liu et al., 2024) in MMLU (Hendrycks et al., 2020) with 200× less tokens via pruning from its larger counterparts. On top of that, a strong 1B model is discovered by pruning from DeepSeek-R1-Distill-Qwen-1.5B (DeepSeek-AI et al., 2025), outperforming Llama-3.2-1B (AI@Meta, 2024) in multiple benchmarks, including MMLU, TruthfulQA (Lin et al., 2021), and AGIEval (Zhong et al., 2023).

2 Related Work

Pruning Pruning removes weights and modifies the model's architecture. Formally, given a neural network $f(x; W)$, pruning produces a new model $f(x; M \odot W)$, where $M \in \{0, 1\}^{|W|}$ is a binary mask that sets certain parameters to zero, and $\odot$ denotes element-wise multiplication. Pruning typically hurts the network's performance, so post-training is often employed to recover the loss (Blalock et al., 2020). Overall, pruning methods can be categorized into two types, unstructured pruning and structured pruning. Unstructured pruning removes individual weights, resulting in sparsified weight matrices, but normally face difficulties in inference speedups when specialized hardware

is unavailable (Dery et al., 2024). In contrast, structured pruning operates at a larger granularity, removing entire weight groups. This includes width pruning (Ma et al., 2023; Ashkboos et al., 2024), which removes groups of coupled weights, and depth pruning (Kim et al., 2024; Siddiqui et al., 2024), which eliminates entire layers. Our focus is on post-training structured pruning, balancing generality and hardware efficiency.

Adaptive Sparsity for Pruning Several works have explored adaptive compression. Dong et al. (2024) selects transformer feedforward experts and removes feedforward neurons during inference based on their high activation magnitudes from input prompts. While this method is effective, we seek an approach that can reduce model size without depending on specific input prompts. An et al. (2023) computes the sample variance of each input feature and weights it by the squared norm of the corresponding column in the weight matrix to determine a layer's importance and assign sparsity accordingly. Wang and Kindratenko (2024) determines the optimal layer-wise sparsity distribution through reinforcement learning, where different sparsity patterns are sampled and updated based on their performance rewards. However, all aforementioned methods failed to take into account the loss on the functional aspect of models, where the overall mapping $X \to Y$ of the LLM is expected to be preserved during the pruning process to minimize the performance drop. Here X and Y are input and output tensors of the model. Our method utilizes the mapping information to assign an adaptive sparsity across different decoder layers.

Knowledge Distillation Knowledge distillation is a technique used to transfer the advanced capabilities of high-performing LLMs to smaller models (Xu et al., 2024). Combining knowledge distillation with pruning can yield strong performance, where the original model acts as the teacher and the compressed model serves as the student (Sreenivas et al., 2024). Conventionally, knowledge distillation can be categorized into black-box (Ho et al., 2022; Hsieh et al., 2023) and white-box distillation (Gu et al., 2023; Latif et al., 2023; Agarwal et al., 2023; Zhou et al., 2023; Shum et al., 2024), depending on whether the model weights and prediction logits of the teacher model can be accessed. In that sense, structural-pruning-based acceleration methods can be roughly viewed as white-box knowledge distillation. However, in this paper, we focus on the adaptive pruning algorithm with supervised fine-tuning only. Though it is expected to achieve even better performance when integrated with state-of-the-art knowledge distillation techniques in post-training, it normally incurs more complexity and training cost, hence we leave that for future exploration.

Accelerating Small Language Model Training Training small language models from scratch demands substantial computational resources, making pruning of larger models with recovery post-training an attractive alternative. Recent works have explored various approaches to this challenge. Xia et al. (2024) proposes a systematic pruning framework that optimizes across four architectural dimensions (layer count, hidden dimension, attention heads, and MLP size) towards target architectures, followed by a continual post-training phase phase with dynamic batch loading. Sreenivas et al. (2024) leverages knowledge distillation, first fine-tuning the original model as a teacher and then transferring its knowledge to the pruned student model. However, these methods rely on single-phase post-training and do not exploit the potential benefits of incremental pruning with interleaved recovery phases, which is demonstrated to be effective in the experiments.

3 Method

3.1 Adapt-Pruner: Layer-wise Adaptive Pruning aim for Mapping-Preserving

Given a large language model $\mathcal{M}$, represented as a sequence of embedded layers with $\mathcal{N}$ decoder layers, denoted as $\mathcal{L}^{\mathcal{N}}$, along with a final output layer, our method leverages the insight that each decoder layer contributes differently to the model's overall performance. Furthermore, the contribution of each layer is measured by its importance of maintaining the original functional mapping of the model after pruning.

Specifically, Adapt-Pruner compresses the model through multiple iterations, with each iteration comprising two steps:

1. **Evaluating layer importance**: Quantitatively computing the importance of each decoder layer and assigning a corresponding pruning sparsity.

2. **Pruning coupled weights**: Grouping the weights within each decoder layer, evaluating

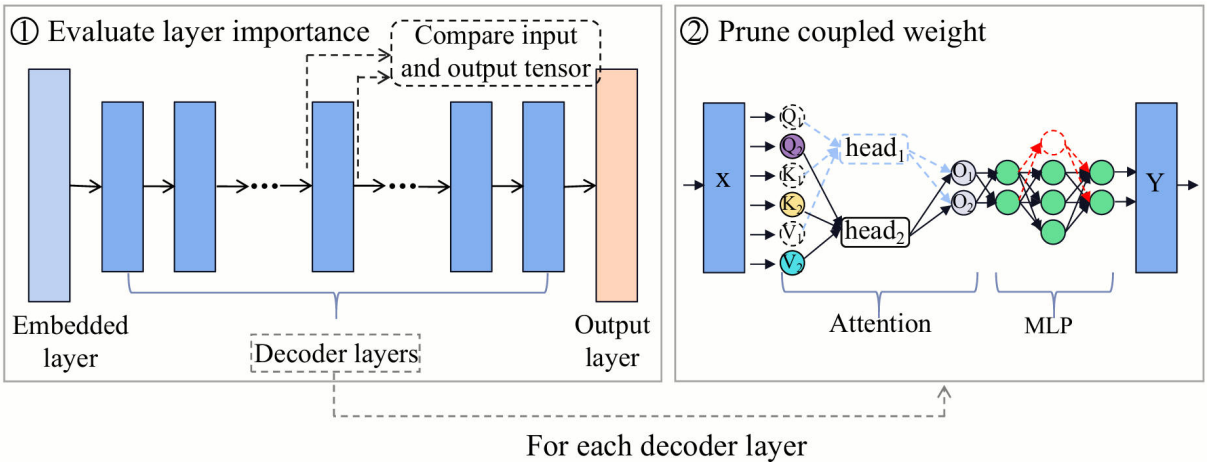


Figure 2: Adapt-Pruner: measuring the distance between each decoder layer's input and output tensors to assess its importance and assigning a corresponding sparsity. Based on this assigned sparsity, the coupled weights in each decoder layer are pruned accordingly.

the importance of each coupled structure, and pruning the least important structures based on the assigned sparsity

Figure 2 gives an illustration of our method.

3.1.1 Assign Sparsity based on Importance

Let $\mathcal{L}^i$ denote the i-th decoder layer, I^i and $\mathcal{S}^i$ represent the importance and sparsity of the i-th decoder layer, and $\mathcal{L}^i_{in}$ and $\mathcal{L}^i_{out}$ denote the input and output tensors of the i-th decoder layer, respectively. Our goal is to estimate the importance of each decoder layer.

Estimate Decoder Layer's Importance Our pruning method targets only the multi-head attention and multilayer perceptron components within the self-attention layers, leaving the hidden size unchanged. Consequently, the input and output tensors for each decoder layer have identical shapes:

$$\forall i = 0, 1, \ldots, N-1, \text{Shape}(\mathcal{L}^i_{in}) = \text{Shape}(\mathcal{L}^i_{out}) \quad (1)$$

$$= (\mathcal{B}, \mathcal{L}, \mathcal{H}) \quad (2)$$

where $\mathcal{B}, \mathcal{L}, \mathcal{H}$ denote the batch size, sequence length, and hidden size, respectively. Based on this, we use a function that measures the vector similarity or distance between $\mathcal{L}^i_{in}$ and $\mathcal{L}^i_{out}$ to assess the changes in the tensor caused by each decoder layer. The greater the similarity or the smaller the distance between $\mathcal{L}^i_{in}$ and $\mathcal{L}^i_{out}$, the less important that decoder layer is. This intuition derives from Li et al. (2024), where it is observed that top self-attention layers have diminished gradient norms and those layers serve similar purposes as identity functions.

A practical choice for this distance measurement is cosine similarity, where $\forall i = 0, 1, \ldots, N-1$ a decoder layer's importance is computed as follows:

$$I^i = -\text{cosine_similarity}(\mathcal{L}^i_{in}, \mathcal{L}^i_{out}) \quad (3)$$

This method can easily be extended to alternative similarity or distance functions, such as Euclidean or Manhattan distance. To ensure consistency, we normalize the decoder layer's importance to the range $[-1, 1]$ with a mean of 0, as follows:

$$I^i \leftarrow I^i - I_{\text{mean}} \quad (4)$$

$$I^i \leftarrow \frac{I^i}{\max |\text{abs}(I)|} \quad (5)$$

Assign Sparsity After obtaining the importance of each layer, an ad-hoc approach is adopted to link a layer's importance to its sparsity, which decides the number of neurons it will be pruned. Let A being any constants, the targeted sparsity $\mathcal{S}^i$ for each layer i can be:

$$\forall i = 0, 1, \ldots, N-1, \mathcal{S}^i = \mathcal{S}_{base} - A \cdot I^i \quad (6)$$

where $\mathcal{S}_{base}$ is the targeted overall sparsity of the model. This formula ensures that each decoder layer's sparsity is inversely proportional to its importance, and the averaged sparsity is consistent with the intended overall model sparsity. We call the hyperparameter A as the amplitude of sparsity.

In addition, since the importance distribution of each layer varies throughout the pruning process according to the observations from Figure 1, progressive adjustment during pruning becomes necessary for good performance. This leads to the multi-stage-pruning design in Adapt-Pruner.

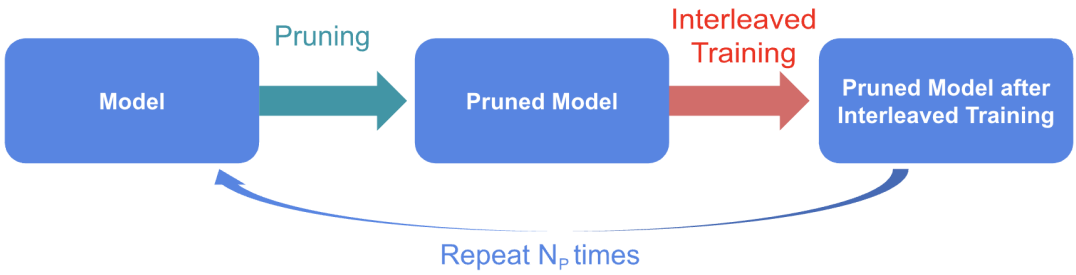


Figure 3: Adapt-Accel: Incremental pruning with interleaved training. N_P number of interleaves are adopted in the whole process. Given a model with size $|\mathcal{L}_{\text{large}}|$ and target size $|\mathcal{L}_{\text{small}}|$, this leads to an incremental pruning ratio of $P = (|\mathcal{L}_{\text{small}}|/|\mathcal{L}_{\text{large}}|)^{1/N_P}$ each time, where the training set is randomly split into N_P subsets for N_P interleaved trainings separately. Notice that the number of training samples gradually increases according to Algorithm 1, as more important weights are expected to be pruned in later phases.

Algorithm 1 Adaptive Pruning Algorithm

Require: Number of decoder layer in the LLM N, decoder layer instances in the LLM $\{\mathcal{L}^i\}_{i=1}^N$, overall sparsity after pruning $\mathcal{S}$, iteration times to prune T, pruning ratio threshold to apply post-train P, training data $\mathcal{D}$, amplitude of sparsity between decoder layers A, similarity function, use cos in default $Sim_{func} \leftarrow$ cosine_similarity()

for $i \leftarrow 1 \ldots T$ **do**
  $S_{cur} \leftarrow (S \cdot i/T)$
  $I \leftarrow \{0\}^N, S \leftarrow \{0\}^N$
  for $j \leftarrow 1 \ldots N$ **do**
    $I^j \leftarrow Sim_{func}(\mathcal{L}^j_{in}, \mathcal{L}^j_{out})$
  end for
  Normalize I^N to have 0 mean value, limit range to [-1, 1] and times -1 if lower is better
  for $j \leftarrow 1 \ldots N$ **do**
    $S^j \leftarrow S_{cur} - A \cdot I^j$
  end for
  while Current sparsity $> P$ **do**
    Adaptively prune the LLM based on S_j
  end while
end for

3.1.2 Pruning Weight Groups Inside Decoder Layer

To enable structure-aware pruning in Adapt-Pruner, methods from Ma et al. (2023); Fang et al. (2023) are employed to build dependency graphs for LLMs, which facilitates automatic identification and extraction of coupled structures in LLMs.

Weight Group Importance With coupled structures and target sparsity defined for each group, the next step is selecting weight matrices to prune with minimized performance degradation. For any grouped weight structure $\mathcal{G} = W^k$, containing k weight matrices, a calibration dataset $\mathcal{D}$ is adopted to assess the relative importance of each matrix. Following (LeCun et al., 1989; Ma et al., 2023), the importance of the i-th weight matrix in layer $\mathcal{L}$ is defined as:

$$I_{W_i} = |\Delta\mathcal{L}(\mathcal{D})| \tag{7}$$

$$= |\mathcal{L}_{W_i}(\mathcal{D}) - \mathcal{L}_{W_i=0}(\mathcal{D})| \tag{8}$$

$$= \left| \frac{\partial \mathcal{L}^\top(\mathcal{D})}{\partial W_i} W_i - \frac{1}{2} W_i{}^\top H W_i + \mathcal{O}\left(\|W_i\|^3\right) \right| \tag{9}$$

where $H = \frac{\partial \mathcal{L}^2(\mathcal{D})}{\partial W_i^2}$ is the Hessian matrix. Calculating the Hessian requires $\mathcal{O}(N^2)$ computational resources, so only first-order terms are retained for acceleration purposes. This simplifies the estimated weight matrix importance to:

$$\hat{I}_{W_i} = \left| \frac{\partial \mathcal{L}^\top(\mathcal{D})}{\partial W_i} W_i \right| \tag{10}$$

Thus, each weight matrix's importance can be approximated by taking the l_1 norm of the element-wise product between its gradient (derived from the calibration dataset) and its weight value. After computing importance scores, the matrices are sorted, where the ones with the lowest scores are pruned to achieve the desired sparsity level. The complete pruning procedure is detailed in Algorithm 1.

3.2 Adapt-Accel: Incremental Pruning with Interleaved Recovery Training

To achieve the target overall sparsity, our model undergoes multiple rounds of pruning, which inevitably leads to performance degradation. Inspired by neuroplasticity in biological neural networks (Walløe et al., 2014), periodic interleaved recovery phases are introduced through post-training after each pruning.

In addition, it is observed that different pruning phases lead to different levels of performance deterioration. Specifically, pruning in later phases is

Model		Method	ARC-e	ARC-c	HellaSwag	OBQA	PIQA	SIQA	Winogrande	Average↑	WikiText2 ppl. ↓
Ratio 20% tune	= w/o	LLM-Pruner	63.92	39.53	64.85	38.47	76.64	42.80	62.85	55.58	14.78
		FLAP	61.01	37.37	58.43	35.40	73.27	43.40	64.46	53.33	16.73
		SliceGPT	52.27	29.01	56.37	32.00	69.84	41.85	59.46	48.69	19.61
		Adapt-Pruner	66.43	39.65	66.71	38.93	76.87	44.10	66.19	**56.98**	**14.54**
Ratio 40% tune	= w/o	LLM-Pruner	33.90	21.79	30.00	25.40	57.02	35.45	51.30	36.41	162.81
		FLAP	25.53	26.25	26.49	25.73	52.36	34.12	50.59	34.44	6987.58
		SliceGPT	33.94	21.79	33.72	26.27	57.73	36.44	49.72	37.09	85.60
		Adapt-Pruner	45.16	25.97	44.88	30.40	66.74	39.03	56.75	**44.13**	**33.75**
Ratio 60% tune	= w/o	LLM-Pruner	26.94	24.43	26.67	27.93	51.22	34.14	49.43	34.39	2501.76
		FLAP	26.60	26.53	26.00	27.07	51.63	33.95	49.62	34.49	141572.73
		SliceGPT	28.95	21.61	28.12	26.13	53.01	34.49	49.01	34.48	218.96
		Adapt-Pruner	32.49	23.64	30.84	26.40	56.66	35.59	49.43	**36.44**	**119.95**

Table 1: Comparison of structured pruning methods across different sparsity levels in Llama-3.1-8B over three trials.

more likely to remove important neurons. Based on this intuition, a linear growth schedule is introduced for training data allocation: For i-th post-training of T total pruning and post-training iterations, $|D_i|$ tokens are sampled from $|D|$ total training tokens where

$$|D_i| = \frac{2(i+1)}{|D|(|D|+1)} \tag{11}$$

This linear growth schedule for data allocation serves two key purposes:

- It enables more frequent parameter updates during later phases when the model requires more extensive recovery.
- And it preserves knowledge acquired in earlier phases from being pruned in subsequent iterations.

This recovery phase is further optimized by splitting and distributing the learning rate schedule across post-training phases, ensuring efficient restoration of model performance after each pruning step.

4 Experiment

4.1 Adapt-Pruner as Effective LLM Pruners

Adapt-Pruner exploits the skewness of importance across layers in a mapping-preserved manner, which allows the pruning process to automatically identify prunable layers that least affect the functionality of the target LLM.

Setup To demonstrate the effectiveness of Adapt-Pruner, different pruners are evaluated and compared on Llama-3.1-8B (AI@Meta, 2024). For task-agnostic performance evaluation of the pruned models, zero-shot classification is performaned on popular common-sense reasoning datasets: ARC-easy and ARC-challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019) and WinoGrande (Sakaguchi et al., 2021), where the averaged accuracy is reported. In particular, length-normalized accuracy is reported for any benchmarks that require length-dependent accuracy. Additionally, we supplement our evaluation with a generation task using WikiText2 (Merity et al., 2016). Following prior work (Ma et al., 2023; An et al., 2023; Ashkboos et al., 2024), we employ the LM Evaluation Harness (Gao et al., 2024) with default parameters, except that all models use the bfloat16 data type, and the batch size is set to 'auto' during evaluation.

Results As shown in Table 1, Adapt-Pruner outperforms all baselines by a non-trivial margin. In particular, Adapt-Pruner excels at preserving commonsense knowledge in LLMs during the pruning process. This is highly relevant to the skewed knowledge distribution across LLM layers, where the bottom layers are observed to be more important than the top layers (Pan et al., 2024; Li et al., 2024).

4.2 Adapt-Accel as Efficient LLM Trainers

Built on top of Adapt-Pruner, Adapt-Accel incorporates interleaved pruning and training to accelerate the optimization of SLMs, which is shown to be a better strategy compared to past methods (Xia et al., 2024; Sreenivas et al., 2024).

Dataset	#Tokens
`Open-Orca/OpenOrca`	1.80B
`allenai/WildChat-1M`	0.66B
`lmsys/lmsys-chat-1m`	0.49B
`teknium/OpenHermes-2.5`	0.49B
`HuggingFaceH4/ultrachat_200k`	0.20B
`openbmb/UltraInteract_sft`	0.18B
`O1-OPEN/OpenO1-SFT`	74M
`yahma/alpaca-cleaned`	12M
`databricks/databricks-dolly-15k`	3.6M
Total	**3.87B**

Table 2: Dataset for training or interleaved training after pruning.

Setup As the training of language models requires a non-trivial amount of computational resources, a smaller family of models are adopted for this section of experiments. MobileLLM (Liu et al., 2024) is a series of SLMs developed by Meta for on-device deployment purposes and stands for one of the strongest SLMs on the scale of 125M/350M.

To demonstrate the superiority of Adapt-Accel, three benchmarks are employed for evaluation, including BBH (Srivastava et al., 2022), TruthfulQA (Lin et al., 2021), AGIEval (Zhong et al., 2023), which assess all methods performance in different aspects beyond commonsense reasoning. A hybrid dataset with 3.87B tokens is utilized for training, where the data source is available in Table 2.

For the interleaved training in Adapt-Accel, a total number of $N_P = 20$ interleavings are adopted in Adapt-Accel, which leads to a pruning ratio of $(125/300)^{1/N_P} \approx 95.7\%$ per training. In other words, the pruning and training will be applied alternatively for 20 times, where each pruning will remove $\sim 95.7\%$ of the current models' weights, along with a follow-up recovery training in a $|D_i| = 2(i+1)/(|D|^2+|D|)$ random samples (without replacement) from the training set at the i-th iteration.

Results As shown in Table 3, Adapt-Accel outperforms both ShearedLLaMA (Xia et al., 2024) and NVIDIA's Minitron Approach (Sreenivas et al., 2024) by a non-trivial margin. This demonstrates that the adaptively pruned models have preserved essential components in the original SLMs, and are still capable of learning new knowledge from the training set effectively.

Method	BBH (3-shot)	TruthfulQA (1-shot)	AGIEval (0-shot)
ShearedLLaMA (Xia et al., 2024)	12.29	34.88	30.53
Minitron (Sreenivas et al., 2024)	12.56	34.13	30.36
Adapt-Accel	**13.75**	**36.18**	**31.34**

Table 3: Comparison of different structural-pruning-based acceleration methods on MobileLLM-350M $\rightarrow$ 125M.

4.3 Adapt-LLMs as Strong LLMs

Adapt-Accel is a favorable tool for fast and flexible customization of model sizes depending on the practical use cases. Specifically, once the large version of LLMs has been obtained from pre-training or other sources, Adapt-Accel can be utilized to inherit the capabilities from the target LLM and accelerate the training of its smaller versions. The family of SLMs obtained in this fashion, named Adapt-LLMs, not only reduces costs during its training process, but also exhibits significant performance improvements.

Setup To provide evidence in support of the claimed strengths of Adapt-LLM, two types of experiments are conducted, individually demonstrating the acceleration and performance benefits of Adapt-LLM. The first branch of experiments focuses on the acceleration aspect of Adapt-Accel, where different sizes of MobileLLMs, ranging from 350M to 1B, are employed. The second branch of experiments emphasizes performance, where Adapt-Accel is applied to Qwen-2.5-0.5B (Team, 2024) and Deepseek-R1-Distill-Qwen-1.5B (DeepSeek-AI et al., 2025), proving that the pruned Adapt-LLMs can still match or even surpass popular strong open-source models without the heavy cost of pretraining. The same benchmarks and datasets of Section 4.2 are adopted here.

Results As shown in Table 4, Adapt-LLM pruned from the larger version of MobileLLMs recovers its performance across all model sizes in all benchmarks for models larger than 350M, at a reduced cost of 200$\times$ less training tokens. This offers strong evidence that Adapt-Accel is a promising acceleration technique especially suitable for cus-

Model	#Training Tokens	BBH (3-shot)	TruthfulQA (1-shot)	AGIEval (0-shot)	MMLU (5-shot)
MobileLLM-125M	$\sim 10^{12}$	**18.45**	32.88	31.19	24.65
MobileLLM-350M $\rightarrow$ 125M	3.87×10^9	13.75	**36.18**	**31.34**	**25.20**
MobileLLM-350M	$\sim 10^{12}$	19.98	29.86	30.85	26.11
MobileLLM-600M $\rightarrow$ 350M	3.87×10^9	**23.61**	**34.22**	**31.93**	**32.50**
MobileLLM-600M	$\sim 10^{12}$	23.99	29.29	30.53	26.46
MobileLLM-1B $\rightarrow$ 600M	3.87×10^9	**26.60**	**32.35**	**34.24**	**36.77**

Table 4: Recovering MobileLLM's performance with much 200$\times$ less tokens. Here MobileLLM-X $\rightarrow$ Y means to prune the model from size X to size Y with Adapt-Accel.

Model	#Training Tokens	BBH (3-shot)	TruthfulQA (1-shot)	AGIEval (0-shot)	MMLU (5-shot)
MobileLLM-350M	$\sim 10^{12}$	19.98	29.86	30.85	26.11
Qwen-2.5-0.5B $\rightarrow$ 350M	3.87×10^9	**26.97**	**34.86**	**33.23**	**38.48**
MobileLLM-1B	$\sim 10^{12}$	25.88	30.26	31.34	24.98
LLaMA-3.2-1B	$\sim 9 \times 10^{12}$	**31.01**	30.41	31.68	31.27
Deepseek-R1-Distill-Qwen-1.5B $\rightarrow$ 1B	3.87×10^9	30.46	**37.50**	**34.34**	**34.62**

Table 5: Surpassing popular open-sourced SLMs via pruning from strong LLMs.

tomizing model sizes flexibly.

Table 5 further demonstrates the performance gain brought by Adapt-Accel via pruning from strong LLMs. It is worth noticing that Deepseek-R1-Distill-Qwen-1.5B (DeepSeek-AI et al., 2025) $\rightarrow$ 1B leads to a 1B model even stronger than LLaMA-3.2-1B (AI@Meta, 2024) in MMLU, delivering three orders of magnitude cost reduction in terms of training tokens. This indicates that Adapt-Accel can serve as a favorable tool for inheriting performance from strong open-source LLMs, allowing researchers and engineers with limited computational resources to still keep up with the fast iteration speed of state-of-the-art LLMs.

Computation Cost An 8B-sized model can be pruned adaptively in 2 to 5 minutes on a single NVIDIA A40 GPU, and in 15 to 30 minutes on an Intel(R) Xeon(R) Gold 6346 CPU. After applying iterative post-training, the full compression process takes between 3 and 18 hours. Benchmark evaluation of the compressed models requires an additional 15 to 30 minutes. In the experiments of Qwen2.5-0.5B $\rightarrow$ 350M, the training process costs $\sim$ 72 GH200 GPU hours. Compared with the $\sim$ 4608 A100 GPU hours pretraining cost of MobileLLM-350M, it is at least 15$\times$ more efficient.

5 Conclusion

In this paper, a system called Adapt-Pruner is introduced to enable flexible LLM model size customization via adaptive pruning. Adapt-Pruner is motivated by 1) the skewness of importance across decoder layers and 2) the goal of preserving the input-output mapping for all the layers. These two properties of AdaptPruner lead to significant improvement over conventional pruning methods. On top of that, Adapt-Pruner gives rise to a novel acceleration paradigm called Adapt-Accel, which combines Adapt-Pruner with interleaved training, a technique shown to provide non-trivial performance gain over the traditional prune-then-train framework, including ShearedLLaMA and Minitron. Adapt-Accel further enables efficient and flexible customization of model sizes by pruning from larger-sized LLMs. Specifically, it is capable of recovering MobileLLMs' performance from their larger counterparts, and discovers a 1B-sized model with better performance than LLaMA-3.2-1B in multiple benchmarks.

Limitations

Adaptive Pruning has been shown to be effective in improving the commonsense reasoning performance of large language models (LLMs). By reducing unnecessary computations, it allows models to operate more efficiently without significantly compromising accuracy. However, the current software and hardware stack needed to accelerate the inference of such pruned models is still underdeveloped and lacks widespread support, which is expected to be advanced in the near future.

References

Rishabh Agarwal, Nino Vieillard, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. 2023. Gkd: Generalized knowledge distillation for auto-regressive sequence models. *arXiv preprint arXiv:2306.13649*.

AI@Meta. 2024. Llama 3 model card.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2023. Fluctuation-based adaptive structured pruning for large language models. *Preprint*, arXiv:2312.11983.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicegpt: Compress large language models by deleting rows and columns. *Preprint*, arXiv:2401.15024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. 2020. What is the state of neural network pruning? In *Proceedings of Machine Learning and Systems*, volume 2, pages 129–146.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. 2024. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Yuntian Deng, Wenting Zhao, Jack Hessel, Xiang Ren, Claire Cardie, and Yejin Choi. 2024. Wildvis: Open source visualizer for million-scale chat logs in the wild. *Preprint*, arXiv:2409.03753.

Lucio Dery, Steven Kolawole, Jean-François Kagy, Virginia Smith, Graham Neubig, and Ameet Talwalkar. 2024. Everybody prune now: Structured pruning of llms with only forward passes. *Preprint*, arXiv:2402.05406.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *Preprint*, arXiv:2305.14233.

Harry Dong, Beidi Chen, and Yuejie Chi. 2024. Prompt-prompted adaptive structured pruning for efficient llm generation. In *First Conference on Language Modeling*.

Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. 2023. Depgraph: Towards any structural pruning. *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. A framework for few-shot language model evaluation.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2022. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*.

Yang He and Lingao Xiao. 2023. Structured pruning for deep convolutional neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071*.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai,

and 6 others. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. *Preprint*, arXiv:2404.06395.

Katikapalli Subramanyam Kalyan. 2024. A survey of gpt-3 family large language models including chatgpt and gpt-4. *Natural Language Processing Journal*, 6:100048.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. 2024. Shortened llama: Depth pruning for large language models with comparison of retraining methods. *Preprint*, arXiv:2402.02834.

Ehsan Latif, Luyang Fang, Ping Ma, and Xiaoming Zhai. 2023. Knowledge distillation of llm for education. *arXiv preprint arXiv:2312.15842*.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. *Advances in neural information processing systems*, 2.

Pengxiang Li, Lu Yin, and Shiwei Liu. 2024. Mix-ln: Unleashing the power of deeper layers by combining pre-ln and post-ln. *arXiv preprint arXiv:2412.13795*.

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". 2023. Openorca: An open dataset of gpt augmented flan reasoning traces. https://https://huggingface.co/datasets/Open-Orca/OpenOrca.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*.

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, and 1 others. 2024. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. *arXiv preprint arXiv:2402.14905*.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning. *Preprint*, arXiv:2301.13688.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *Preprint*, arXiv:2306.02707.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. 2024. Lisa: Layerwise importance sampling for memory-efficient large language model fine-tuning. *arXiv preprint arXiv:2403.17919*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. PyTorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. 2013. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6655–6659. IEEE.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *Preprint*, arXiv:1904.09728.

KaShun Shum, Minrui Xu, Jianshu Zhang, Zixin Chen, Shizhe Diao, Hanze Dong, Jipeng Zhang, and Muhammad Omer Raza. 2024. First: Teach a reliable large language model through efficient trustworthy distillation. *arXiv preprint arXiv:2408.12168*.

Shoaib Ahmed Siddiqui, Xin Dong, Greg Heinrich, Thomas Breuel, Jan Kautz, David Krueger, and Pavlo Molchanov. 2024. A deeper look at depth pruning of llms. *Preprint*, arXiv:2407.16286.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-versio

Sharath Turuvekere Sreenivas, Saurav Muralidharan, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. 2024. Llm pruning and distillation in practice: The minitron approach. *Preprint*, arXiv:2408.11796.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, and 1 others. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Teknium. 2023. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023b. Llama 2: Open foundation and fine-tuned chat models.

Solveig Walløe, Bente Pakkenberg, and Katrine Fabricius. 2014. Stereological estimation of total cell numbers in the human cerebral and cerebellar cortex. *Frontiers in human neuroscience*, 8:508.

Boyao Wang and Volodymyr Kindratenko. 2024. Rl-pruner: Structured pruning using reinforcement learning for cnn compression and acceleration. *arXiv preprint arXiv:2411.06463*.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2024. Sheared llama: Accelerating language model pre-training via structured pruning. *Preprint*, arXiv:2310.06694.

Shijie Xia, Yiwei Qin, Xuefeng Li, Yan Ma, Run-Ze Fan, Steffi Chern, Haoyang Zou, Fan Zhou, Xiangkun Hu, Jiahe Jin, and 1 others. 2025. Generative ai act ii: Test time scaling drives cognition engineering. *arXiv preprint arXiv:2504.13828*.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024. A survey on knowledge distillation of large language models. *Preprint*, arXiv:2402.13116.

Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Jia Deng, Boji Shan, Huimin Chen, Ruobing Xie, Yankai Lin, Zhenghao Liu, Bowen Zhou, Hao Peng, Zhiyuan Liu, and Maosong Sun. 2024. Advancing llm reasoning generalists with preference trees. *Preprint*, arXiv:2404.02078.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. Wildchat: 1m chatGPT interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric. P Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. 2023. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *Preprint*, arXiv:2309.11998.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. 2023. Distillspec: Improving speculative decoding via knowledge distillation. *arXiv preprint arXiv:2310.08461*.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. A survey on model compression for large language models. *Preprint*, arXiv:2308.07633.

A Experimental Details

Basic Setup We extend the LLM-Pruner framework (Ma et al., 2023; Fang et al., 2023) as our baseline, which incorporates modules for computing similarity scores and adaptive pruning using PyTorch (Paszke et al., 2019). Our experiments utilize both NVIDIA GH200 and H100 GPUs, where a single GPU (either GH200 or H100) is utilized for pruning and evaluation tasks while 4 additional GPUs are employed in parallel for post-training. The metrics and the number of shots used for each benchmark are available in Table **??**.

Benchmark	Metric	n-shot
ARC-e	acc_norm	0
ARC-c	acc_norm	0
BoolQ	acc	0
HellaSwag	acc_norm	0
OBQA	acc_norm	0
PIQA	acc_norm	0
SIQA	acc	0
Winogrande	acc	0
WikiText2	word_perplexity	0
BBH	exact_match	3
TruthfulQA	acc	1
AGIEval	acc_norm	0
MMLU	acc	5

Table 6: Details for Evaluation Benchmarks.

Experimental Settings for Adapt-Pruner For Adapt-Pruner, we set $A = 0.02$ in Equation 6 based on Ablation study B.2. We evaluate three models with three sparsity choices for each model. All the methods are aligned using Slimpajama (Soboleva et al., 2023) as the calibration dataset, which comprises 512 sequences with a maximum length of 64 tokens.

Experimental Settings for Adapt-Accel and Adapt-LLMs For Adapt-Accel and Adapt-LLMs, we keep the same choice of sparsity amplitude $A = 0.02$ in Equation 6. The global batch size is set to 128, with maximal learning rate of 2×10^{-5}, and minimal learning rate of 2×10^{-6}. WSD scheduler (Hu et al., 2024) is applied, where the combination of learning rate in all interleaved training iterations forms a WSD scheduler, with 5% warmup steps and 10% linear decay steps at the end. For the ShearedLLaMA (Xia et al., 2024) comparison, we keep these hyperparameters while allocating 10% of tokens for pruning optimization and 90% for post-training. Similarly, for Minitron comparison (Sreenivas et al., 2024), we keep the same settings and fine-tune the teacher model for one epoch on the dataset.

B Additional Experimental Results

B.1 Adapt-Pruner as Strong Pruners

We evaluate various structured pruning methods on three LLaMA-series models: LLaMA-3.1-8B, LLaMA-3.2-3B, and LLaMA-3.2-1B (AI@Meta, 2024). For each model, we test sparsity levels of 20%, 40%, and 60% over three trials to assess our method's effectiveness across different model sizes.

As shown in Table 7, Adapt-Pruner demonstrates significant improvements in commonsense benchmarks, especially at 40% sparsity level. We conjecture the sparsity of ∼60% to be the ceiling of the redundancy information in LLaMA-series models, since further pruning incurs severe performance degradation, with closing gaps across different pruning methods.

B.2 More Ablation Study

Optimal Interleaving Frequency Interleaved training is shown to be quite beneficial compared to the traditional prune-then-train paradigm. To further investigate the optimal frequency for interleaved training during the pruning process, additional experiments are conducted on MobileLLM-350M→125M.

All experiments are conducted on MobileLLM-350M, with 1B tokens sampled from Slimpajama (Soboleva et al., 2023) and 1/6 random samples in the aforementioned dataset (Table 2). Two types of benchmarks are adopted, including the commonsense benchmarks in Section 4.1 and MMLU benchmark in Section 4.2. Pruning ratios per training, ranging from $\{0.36, 0.90, 0.925, 0.95, 0.975, 0.99\}$, are searched to decide the optimal value, which corresponds to the number of interleaves $N_P \in \{1, 9, 12, 20, 38, 96\}$ separately.

As shown in Figure 4, the optimal interleave frequency is around 95% pruning ratio per training, i.e. every recovery training after $\sim 5\%$ removal of weights or neurons. It is worth noticing that the performance degrades significantly when the interleave frequency is too low or too high, implying the occurrence of knowledge loss caused by

Model	Pruning Sparsity	Method	#Param.	ARC-e	ARC-c	HellaSwag	OBQA	PIQA	SIQA	Winogrande	Average↑	WikiText2 ppl.↓
LLaMA-3.1-8B	Ratio = 20%	LLM-Pruner	6.73B±0.00	63.92±0.76	39.53±1.48	64.85±0.99	38.47±1.11	76.64±0.47	42.80±0.47	62.85±0.81	55.58±0.54	14.78±0.25
		FLAP	6.48B±0.00	61.01±1.35	37.37±0.12	58.43±0.88	35.40±0.57	73.27±1.05	43.40±0.58	64.46±0.60	53.33±0.23	16.73±0.35
		SliceGPT	7.22B±0.00	52.27±0.33	29.01±0.12	56.37±0.58	32.00±0.71	69.84±0.56	41.85±0.57	59.46±0.55	48.69±0.17	19.61±0.14
		Adapt-Pruner	6.66B±0.00	66.43±0.85	39.65±0.67	66.71±0.49	38.93±0.34	76.87±0.16	44.10±0.32	66.19±1.41	**56.98±0.35**	**14.54±0.51**
	Ratio = 40%	LLM-Pruner	5.27B±0.00	33.90±0.48	21.79±0.11	30.00±0.29	25.40±1.28	57.02±0.08	35.45±0.09	51.30±0.73	36.41±0.19	162.81±33.99
		FLAP	4.98B±0.00	25.53±0.55	26.25±0.41	26.49±0.12	25.73±0.57	52.36±0.86	34.12±0.61	50.59±0.42	34.44±0.18	6987.58±3502.55
		SliceGPT	5.44B±0.00	33.94±0.46	21.79±0.44	33.72±0.32	26.27±0.41	57.73±0.74	36.44±0.25	49.72±0.95	37.09±0.20	85.60±1.84
		Adapt-Pruner	5.25B±0.00	45.16±0.21	25.97±0.26	44.88±0.72	30.40±0.85	66.74±0.40	39.03±0.24	56.75±0.11	**44.13±0.26**	**33.75±0.25**
	Ratio = 60%	LLM-Pruner	3.98B±0.00	26.94±1.39	24.43±1.65	26.67±0.69	27.93±1.59	51.22±0.69	34.14±0.26	49.43±1.31	34.39±0.07	2501.76±2400.07
		FLAP	3.71B±0.00	26.60±0.36	26.53±0.54	26.00±0.17	27.07±1.84	51.63±0.41	33.95±0.81	49.62±1.27	34.49±0.20	141572.73±70694.04
		SliceGPT	3.73B±0.00	28.95±0.42	21.61±0.53	28.12±0.23	26.13±0.66	53.01±0.34	34.49±0.33	49.01±0.29	34.48±0.07	218.96±2.34
		Adapt-Pruner	3.96B±0.00	32.49±0.26	23.64±0.43	30.84±0.17	26.40±0.49	56.66±0.29	35.59±0.36	49.43±1.24	**36.44±0.13**	**119.95±12.88**
LLaMA-3.2-3B	Ratio = 20%	LLM-Pruner	2.70B±0.00	56.65±2.38	32.14±1.33	55.94±0.97	33.60±0.28	73.14±1.07	41.79±0.65	56.70±0.39	**50.00±0.79**	20.89±1.83
		FLAP	2.58B±0.00	50.42±0.70	29.75±0.77	48.87±0.83	31.47±0.25	67.99±0.53	41.21±0.80	58.04±0.14	46.82±0.30	19.90±0.26
		SliceGPT	3.31B±0.00	44.77±1.70	27.25±1.22	49.57±1.15	29.27±0.66	66.07±0.40	38.26±0.49	56.09±0.65	44.47±0.81	23.12±0.13
		Adapt-Pruner	2.68B±0.00	54.70±1.50	31.43±0.22	55.37±0.28	33.73±1.20	71.69±0.40	42.67±0.32	59.14±0.54	49.82±0.48	**17.41±0.07**
	Ratio = 40%	LLM-Pruner	2.10B±0.00	32.41±1.22	21.84±1.39	29.44±0.20	25.93±0.90	55.10±0.78	34.18±0.55	50.78±1.07	35.67±0.66	409.38±51.28
		FLAP	2.00B±0.00	27.06±0.59	24.34±1.28	26.15±0.25	27.53±1.05	50.36±0.44	35.21±0.45	50.49±0.49	34.45±0.34	89300.67±12876.34
		SliceGPT	2.52B±0.00	31.86±0.24	21.33±0.35	31.90±0.23	26.47±0.98	56.07±1.04	36.06±0.25	49.28±0.89	36.14±0.14	89.82±0.87
		Adapt-Pruner	2.06B±0.00	40.67±0.19	24.57±0.18	36.89±0.31	26.27±0.77	61.37±0.42	37.10±0.23	51.83±0.21	**39.82±0.24**	**45.13±0.50**
	Ratio = 60%	LLM-Pruner	1.59B±0.00	28.59±0.73	23.04±0.42	27.11±0.16	27.20±0.71	51.91±0.80	33.86±0.17	49.72±2.09	34.49±0.48	2032.20±48.33
		FLAP	1.50B±0.00	25.26±0.33	26.00±0.78	26.62±0.23	30.07±0.82	51.61±0.29	33.54±0.61	49.14±0.43	34.61±0.22	347771.12±79021.97
		SliceGPT	1.78B±0.00	28.27±0.21	22.12±0.21	28.08±0.13	26.13±0.50	52.83±0.16	35.38±0.02	50.20±0.84	34.72±0.06	**257.56±4.96**
		Adapt-Pruner	1.54B±0.03	31.09±0.73	22.13±0.45	29.99±0.45	27.60±0.65	54.62±0.62	34.78±0.29	49.25±0.91	**35.64±0.15**	259.39±64.54
LLaMA-3.2-1B	Ratio = 20%	LLM-Pruner	1.05B±0.00	49.34±0.52	28.53±0.28	47.15±0.43	29.60±1.18	68.57±0.72	38.55±0.82	52.25±0.34	44.85±0.38	28.39±1.38
		FLAP	0.94B±0.00	28.73±0.09	24.89±0.40	27.38±0.15	26.73±0.09	52.03±0.72	34.09±0.18	49.12±1.19	34.71±0.23	766.81±86.10
		SliceGPT	1.34B±0.00	39.97±1.53	24.86±0.16	42.02±0.53	28.60±0.43	62.62±0.40	37.22±0.54	52.09±0.22	41.06±0.47	31.12±0.40
		Adapt-Pruner	1.04B±0.00	49.62±0.57	28.44±0.81	47.36±0.25	31.47±0.25	68.97±0.14	41.03±0.21	55.06±0.85	**45.99±0.13**	**22.50±0.34**
	Ratio = 40%	LLM-Pruner	0.85B±0.00	34.58±0.56	22.18±0.66	28.88±0.24	23.47±0.62	57.98±0.59	34.85±0.07	51.59±1.58	36.22±0.37	214.73±33.60
		FLAP	0.77B±0.00	25.87±0.62	26.82±0.61	26.80±0.08	27.20±0.33	50.87±0.19	34.48±0.33	48.75±0.64	34.40±0.19	96346.37±6993.27
		SliceGPT	1.06B±0.00	30.42±0.04	21.78±0.16	29.64±0.10	25.27±0.77	54.79±0.16	36.08±0.25	49.51±0.65	35.36±0.22	116.33±3.77
		Adapt-Pruner	0.84B±0.00	36.70±1.39	23.29±1.17	33.87±0.51	25.27±0.57	60.08±0.13	36.79±0.44	51.84±0.66	**38.26±0.44**	**64.91±0.92**
	Ratio = 60%	LLM-Pruner	0.67B±0.00	28.39±0.32	24.29±0.29	26.27±0.28	25.13±0.57	51.83±0.51	34.02±0.46	49.33±0.55	34.18±0.17	1859.01±191.57
		FLAP	0.62B±0.00	26.61±0.19	26.79±0.46	26.53±0.21	31.27±0.90	50.80±0.71	34.08±0.15	49.93±0.16	35.14±0.17	122310.98±25718.08
		SliceGPT	0.79B±0.00	28.16±0.33	22.98±0.18	27.90±0.17	26.33±0.25	53.28±0.19	35.19±0.30	48.09±0.82	34.56±0.14	380.60±9.94
		Adapt-Pruner	0.66B±0.00	32.57±0.18	22.35±0.86	29.14±0.12	25.00±0.33	54.92±0.35	35.28±0.15	50.09±1.04	**35.62±0.35**	**295.87±31.63**

Table 7: Experiment details of comparing structured pruning methods across different sparsity levels and models over three trials.

large-portion pruning, or unstable learning due to too-frequent pruning. This phenomenon is quite intriguing as it resembles the disappearance of neurons in human brains.

Sensitivity Analysis of Sparsity Amplitude Parameter A We empirically investigate the impact of amplitude parameter A in Equation 6 on our algorithm's performance. The amplitude A directly influences the architectural search space of the compressed model: insufficient amplitude constrains the exploration of potential architectural configurations, while excessive amplitude can lead to structural imbalances that degrade model performance. To systematically analyze this relationship, we conduct experiments on LLaMA-3.1-8B using a controlled setup with consistent parameters (50% overall sparsity ratio and 50 pruning iterations) while varying the amplitude A across pruning steps. This experimental design allows us to isolate and quantify the specific effects of amplitude variation on model compression outcomes.

As demonstrated in Table 8, the baseline case of $A = 0$ represents uniform pruning without our proposed adaptive mechanism. Notably, varying the amplitude A yields different final parameter counts due to its influence on architectural decisions during compression. Our experimental results reveal that $A = 0.02$ achieves optimal performance, maximizing the benchmark average while minimizing model complexity on WikiText2. The performance exhibits a non-monotonic relationship with A: as amplitude increases, model performance initially improves before degrading, consistent with the theoretical trade-off between exploration capacity and architectural stability.

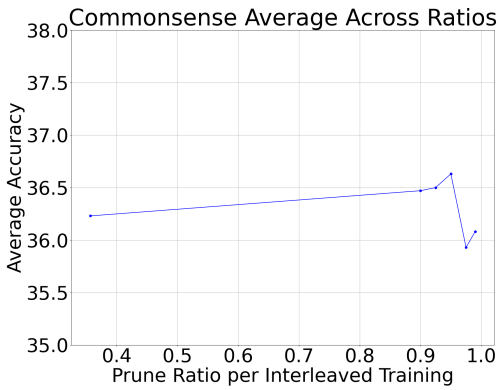
C Broader Impacts

This paper presents work whose goal is to advance the field of Machine Learning. Specifically, we expect the proposed methods can greatly reduce the pre-training cost of SLMs, leading to less computational resource consumption, less carbon dioxide emissions, and faster development of SLMs.

To the best of our knowledge, the only risk of the technique is that the LLM may lose certain abilities during pruning, which may lead to hallucination. Alignment methods can be further applied to compensate for this risk.

D Licenses

OpenOrca (Lian et al., 2023; Mukherjee et al., 2023; Longpre et al., 2023; Touvron et al., 2023a,b), WildChat-1M (Zhao et al., 2024; Deng et al., 2024), ultrachat-200k (Ding et al., 2023),



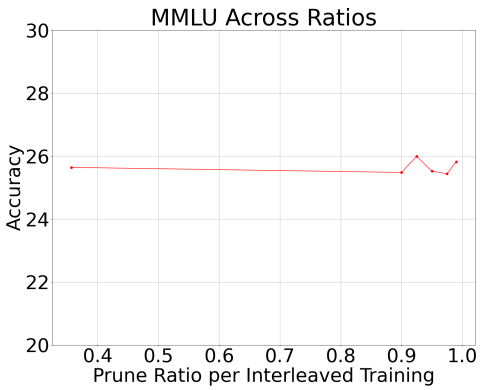


Figure 4: Ablation over pruning ratio per interleaved training, which shows the optimal value is $\sim 95\%$, meaning it is best to interleave the training of SLM after every $\sim 5\%$ weight/neuron removals.

Amplitude A	Para. Num.	ARC-e	ARC-c	BoolQ	HellaSwag	OBQA	PIQA	Winogrande	Average↑	WikiText2 ppl.↓
0	4.54B	29.04	21.84	45.47	28.22	27.00	53.81	48.30	36.24	621.03
0.005	4.62B	37.25	22.87	55.90	33.89	27.80	59.52	52.64	41.41	122.04
0.01	4.61B	37.29	25.17	60.73	35.60	28.80	60.77	52.09	42.92	95.59
0.02	4.43B	40.61	26.45	62.05	35.18	28.20	62.30	53.67	**44.07**	**90.80**
0.04	4.04B	34.51	24.06	56.21	31.17	26.40	57.34	51.78	40.21	314.90

Table 8: Sensitivity analysis of amplitude parameter A in progressive pruning, with 50% sparsity and 50 pruning steps.

and UltraInteract-sft (Yuan et al., 2024) are released under MIT license. OpenO1-SFT (Xia et al., 2025) is released under Apache-2.0 license. `yahma/alpaca-cleaned` (Taori et al., 2023) is released under cc-by-4.0 license. Dolly-15K (Conover et al., 2023) is released under cc-by-sa-3.0 license. OpenHermes-2.5 (Teknium, 2023) is released under FAFO license, where the source of each independent sub-dataset is unidentifiable, but the research-purpose usage is in accordance with its intend usage. LMSYS-Chat-1M (Zheng et al., 2023) is released under LMSYS-Chat-1M Dataset License Agreement (`https://huggingface.co/datasets/lmsys/lmsys-chat-1m#lmsys-chat-1m-dataset-license-agreement`), which potentially contains unsafe conversations that may be perceived as offensive or unsettling. To avoid negative impacts to society, we will not release the model until it is fully aligned and in compliance with all pertinent laws and regulations.

E AI Usage

ChatGPT is used to correct grammatical errors and polish the paper writing.