DarwinLM: Evolutionary Structured Pruning of Large Language Models with Speedup Guarantees

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have achieved significant success across various NLP tasks. However, their massive computational costs limit their widespread use, particularly in real-time applications. Structured pruning offers an effective solution by compressing models and directly providing end-to-end speed improvements, regardless of the hardware environment. Meanwhile, it is known that different components of a model exhibit varying sensitivities towards pruning. To address this, we propose *DarwinLM*, a novel non-uniform, speed-aware structured pruning method. Our approach integrates evolutionary search with conventional second-order structured pruning to generate non-uniform sparse models. *DarwinLM* guarantees speedup by maintaining a fixed, hardware-specific acceleration across the evolutionary process, and producing speed-optimized offspring models. Additionally, we propose group-wise and in-group search strategies to further reduce the search space. Extensive experiments on Llama-2-7B and Llama-3.1-8B demonstrate that our method achieves state-of-the-art performance compared to previous structured pruning methods. For instance, our approach achieves better results than ShearedLlama with $5\times$ less training data during post-compression training. We will release our code to facilitate reproducibility.

1 Introduction

The high accuracy of Transformer-based models on a wide range of tasks comes with massive computational requirements, which hinders deployability. A long line of research has been conducted to reduce the computational cost of large language models via methods such as quantization (Frantar et al., 2022; Dettmers et al., 2023), pruning (Xia et al., 2024; Frantar and Alistarh, 2023) and distillation (Hsieh et al., 2023). In this work, we explore structured pruning on large language models (LLMs) by removing whole rows or columns in a weight matrix, resulting in regular but smaller matrices. Unlike unstructured pruning (Frantar and Alistarh, 2023), the model produced by structured pruning can be accelerated on mainstream hardware without any specific design for computation.

While conventional pruning methods generally prune each layer or block uniformly, *non-uniform compression methods*, e.g. (Yin et al., 2023; Anonymous, 2024), showed that different LLM layers can have significantly different "sensitivities" to pruning, which can be leveraged to obtain higher compression, while preserving accuracy. Moreover, previous LLM structured pruning techniques (Xia et al., 2024) target a fixed sparsity instead of actual speedup. Even if a pruned model with fixed sparsity achieves the desired speedup, it may still not necessarily be the best-performing model at that specific speedup. To address this, smaller-scale pruning methods such as ZipLM (Kurtić et al., 2024) propose to utilize a dynamic-programming-based search (Frantar and Alistarh, 2022) to obtain a sparse model with runtime guarantees. However, there are several challenges for methods such as ZipLM on large-scale models such as Llama-2-7B (Touvron et al., 2023): for instance, ZipLM only considers the local layer-wise error during the search, which is not consistent with performance on in-context learning (ICL) or downstream tasks.

Overview. In this paper, we propose a new structured pruning method based on evolutionary search called *DarwinLM*, which addresses these issues. Specifically, the search starts from a pruned "parent" model, generated by pruning the original model using second-order information. In each step, it generates "offspring" candidate models in a runtime-aware manner. Specifically, we use restricted group-wise and in-group mutations in order to reduce the search space. Empirically, our method scales to LLMs, and achieves state-of-the-art performance on both one-shot and post-training settings compared with previous pruning methods.

For instance, we obtain smaller and better sparse models on both Llama-2 and Llama-3.1 compared with the state-of-the-art ZipLM pruning method; furthermore, on Llama-2-7B, our model guaranteeing 2.5× speedup (2.6B parameters) outperforms the Shearedllama-2.7B structured pruned model, while using 5× *less data* for post-compression training (50B Tokens vs. 10B Tokens). We also compare our method with a line of structured pruning methods including ShortGPT (Men et al., 2024), Shortened-Llama (Kim et al., 2024), and EvoPress (Anonymous, 2024) in a one-shot setting, showing that *DarwinLM* provides better performance across compression rates.

We summarize our contributions as follows:

- We propose a new structured pruning method that unifies second-order pruning and evolutionary search, with the goal of producing the optimal non-uniform sparse model given a predetermined hardware acceleration constraint.
- To reduce the large search space induced by large language models, we utilize new group-wise and in-group mutations to accelerate the evolutionary search process.
- We introduce regularization terms to the fitness environment that penalize excessive sparsity deviations and the pruning of early layers. Empirical results show that this approach enhances performance on downstream tasks.
- Comprehensive experiments validate the effectiveness of *DarwinLM*. Our method compresses Llama-2-7B to 2.6B parameters, achieving a 2.5× speedup, and Llama-3.1-8B to 4.4B parameters with 2.0× acceleration, with lower performance degradation compared to previous methods. Moreover, our approach outperforms ShearedLlama while using 5× fewer data tokens for post-training.

2 Related Work

Structured Pruning Methods. Structured pruning methods for LLMs (Wang et al., 2020; Ma et al., 2023; Xia et al., 2024; Kurtić et al., 2024; Men et al., 2024; Kim et al., 2024; An et al., 2024; Muralidharan et al., 2024) typically focus on pruning along the depth dimension (dropping layers) or on pruning width (such as attention heads, and MLP intermediate dimensions), and in some cases, both depth and width. Among recent advances, the state-of-the-art is provided by ShearedLLaMA (Xia et al., 2024), which utilizes targeted structured pruning, which reduces a larger model to a specified target shape by pruning layers, attention heads, and intermediate or hidden dimensions in an end-to-end process that is split into regularized fine-tuning, pruning, and further fine-tuning. In addition, it implements *dynamic batching*, which adjusts the composition of sampled data in each training batch dynamically, based on varying loss proxies across different evaluation domains. By contrast, *DarwinLM* provides a much more accurate method for structured pruning, combining evolutionary search and second-order information, requiring only a fraction of the training data to recover accuracy.

The recent work on MINITRON (Muralidharan et al., 2024) established a set of effective compression practices for LLMs by integrating depth, width, attention, and MLP pruning with knowledge distillation (KD)-based retraining. These practices are derived from an in-depth empirical exploration of pruning strategies across each axis, methods for combining different axes, distillation approaches, and search techniques to identify optimal compressed architectures. Our contributions are orthogonal to MINITRON, as we mainly investigate more accurate pruning techniques—many of their findings should also transfer to our setting. Unfortunately, MINITRON uses a closed fine-tuning dataset, and did not release pre-fine-tuning checkpoints which would allow us to compare pruning techniques and end models. Thus, we are unable to perform a direct comparison with MINITRON.

Non-uniform Pruning Methods. The distribution of importance across depth, attention heads, and width in the model varies between layers and is not uniform. Low-importance modules tend to be concentrated in specific locations and positions within the model. In the LLM domain, Klein et al. (2023) utilized multi-objective NAS to compress LLMs while optimizing their performance for fine-tuning downstream tasks. SIMPLE (Tao et al., 2023) detects redundant network structures by applying learnable masks to all compressible components, followed by sparse training. Unlike these prior methods, we propose a novel non-uniform, speedup-aware structured pruning framework that integrates group-wise and in-group evolutionary search with speed-level substructure components to generate non-uniform sparse models.

From the technical standpoint, our work builds upon the recent EvoPress approach (Anonymous, 2024). EvoPress was designed for *non-uniform unstructured pruning*, *non-uniform quantization*, and *layer dropping*, with a focus on achieving a

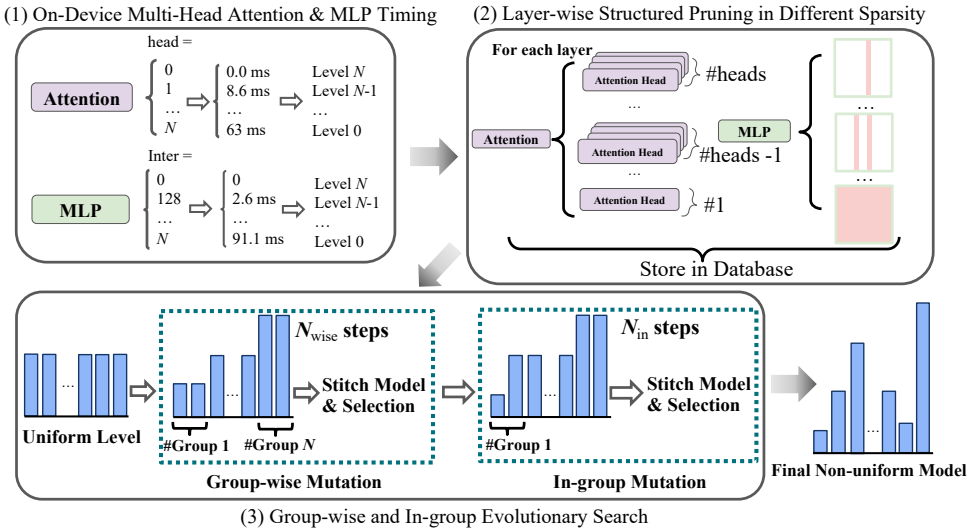


Figure 1: Visual illustration of *DarwinLM* pipeline. 1) profiling the time for attention and MLP modules. 2) generate speedup level database with different sparsities by second-order structured pruning. 3) group-wise and in-group evolutionary search based on the speedup level database.

target model size in a *one-shot setting*. By contrast, *DarwinLM* differs by building a hardware specific acceleration database and focusing on fine-grained structured pruning (at the level of rows/columns). This allows us to generate compressed models with explicit speedup guarantees through a speedup-preserving mutation operator. Additionally, we introduce group level mutation and incorporate regularization into the fitness environment, taking into account the finetuning process following the model compression.

Other Compression Methods. Several approaches have been explored in the literature to reduce computational and memory requirements of LLMs without significantly degrading performance, including knowledge distillation, quantization, binarization, and sparsity. In knowledge distillation (Hinton et al., 2015; Sanh, 2019; Gu et al., 2024; Liu et al., 2024; Xu et al., 2024a), a smaller, simpler model (the "student") is trained to replicate the behavior of a larger, more complex model (the "teacher"). The goal is to transfer the knowledge from the teacher to the student while retaining most of the performance benefits of the larger model. Quantization (Xiao et al., 2023; Lin et al., 2024; Li et al., 2024) reduces the precision of model weights and activations, often converting floating-point numbers (such as 32-bit or 16-bit) into lower-bit representations (such as 8-bit or even lower). This reduces the memory footprint and computational complexity of the model. Binarization (Wang et al., 2023; Huang et al., 2024; Xu et al., 2024b; Ma et al., 2024) compresses models by restricting weights and/or activations to binary values (e.g., -1 or 1). While this can dramatically reduce the model size and computation, the challenge lies in maintaining accuracy, which often degrades with aggressive binarization.

3 Method

We begin with outlining the problem formulation in Section 3.1, followed by a description of the pruning method in Section 3.2. We discuss the generation of the speedup level database, which is used for the search, in Section 3.3. We describe the fitness environment in Section 3.4 and the evolutionary algorithm in Section 3.5. An overview of the pipeline is provided in Figure 1.

3.1 Problem Setup

Given a target speedup, *DarwinLM* aims to find the model with the best sparsity allocation adhering to this constraint. Formally, let $\boldsymbol{S}$ denote the target speedup factor, let $\boldsymbol{M}_{base}$ be the base model, and let $\boldsymbol{T}_{base}$ be the runtime of the base model. Then, our problem is reduced to identifying

$$\hat{\boldsymbol{M}} = \arg\max_{\boldsymbol{M}} f(\boldsymbol{M}) \tag{1}$$

under the constraint $\boldsymbol{T}(\boldsymbol{M}) \leq \boldsymbol{T}_{base}/\boldsymbol{S}$, where $f(\cdot)$ is a function that evaluates the performance of a model and $\boldsymbol{T}(\cdot)$ measures the inference time of a given model. However, Equation 1 presents a non-differentiable optimization problem. To address this, we will employ an evolutionary algorithm.

3.2 Second-Order Structured Pruning

We follow ZipLM (Kurtić et al., 2024), which itself is an extension of Optimal Brain Surgeon (OBS)

(Hassibi and Stork, 1992; Kurtic et al., 2022), for *layer-wise structured pruning* of a pre-trained LLM. Specifically, for each layer, given a calibration dataset $\mathbf{X}$ of layer inputs and the original layer weights $\mathbf{W}$, we aim to find

$$\arg\min_{\hat{\mathbf{W}}} ||\mathbf{W}\mathbf{X} - \hat{\mathbf{W}}_{:,\mathbf{M}}\mathbf{X}||_2 \quad (2)$$

subject to $\hat{\mathbf{W}}_{:,\mathbf{M}} \in \mathcal{C}$, where $\mathbf{M}$ refers to a *column mask* and $\mathcal{C}$ is the compression constraint. To ensure that the sparse weights $\hat{\mathbf{W}}$ produce outputs similar to those of the original weights $\mathbf{W}$, we must not only identify the less significant structures for pruning, but also compute an update δ for the remaining weights to compensate for the error introduced by pruning. For this purpose, denote by $\mathbf{H} = \mathbf{X}\mathbf{X}^T$ the Hessian matrix for the ℓ_2-minimization problem in Equation 2. Define $\mathbf{W}_{\mathrm{i},\mathbf{M}}$ as the weight in row i masked by $\mathbf{M}$ and let $(\mathbf{H}^{-1})_{\mathbf{M},\mathbf{M}}$ be the submatrix of the inverse Hessian corresponding to the entries under the mask $\mathbf{M}$. Now, we can compute the optimal structured mask with corresponding weight updates δ by:

$$\arg\min_{\mathbf{M}} \sum_{i=1}^{d_{row}} \mathbf{W}_{\mathrm{i},\mathbf{M}} \cdot (\mathbf{H}^{-\mathbf{1}}_{\mathbf{M},\mathbf{M}})^{-\mathbf{1}} \cdot \mathbf{W}^{\mathrm{T}}_{\mathrm{i},\mathbf{M}} \quad (3)$$

$$\delta = -\mathbf{W}_{:,\mathbf{M}} \cdot (\mathbf{H}^{-1}_{\mathbf{M},\mathbf{M}})^{-\mathbf{1}} \cdot \mathbf{H}^{-1}{}_{\mathbf{M},:} \quad (4)$$

This formulation extends the derivation of OBS to account for all rows d_{row}. In our context, we focus on three types of pruned structures: (1) head pruning in multi-head self-attention, (2) pruning of the intermediate dimension of MLP modules, and (3) removing the entire attention or MLP module.

Granularity. To obtain a consistent speedup after pruning, we enforce the number of dimensions pruned in the MLP module to be a multiple of a pruning granularity constant m. Specifically, for NVIDIA GPUs, we set $m = 32$ to obtain consistent time profiling. For each module, we only consider identifying the pruned columns of the output matrix. Once the pruned structure of the output matrix is determined, the corresponding rows are pruned in the other matrices (i.e., the K, Q, and V matrices in the attention module, and the up and gate matrices in the MLP). However, if the model applies group-query attention (GQA) (Ainslie et al., 2023), such as in Llama-3.1, we avoid pruning the K and V matrices. During the forward pass, we remove the corresponding heads in the repeated K and V matrices to obtain computationally compatible structures and reduce computation.

3.3 Speedup Level Database

We first generate a speedup level database, which will be used in the evolutionary search, where candidate models are stitched together from their respective unitwise speedup levels. However, we must ensure that all considered models in this search process adhere to the targeted speedup factor. We will do so by initializing the search with a valid model and then applying a speedup-preserving mutation operator. For this purpose, we generate the speedup level database in such a way that the difference in terms of (absolute) speedup between two levels is consistent across all levels and modules. (In our implementation, all attention / MLPs employ the same speedup step size, but the step size for attention differs from that of MLPs.) This way, we can mutate a model while maintaining the targeted speedup by simply increasing the same number of levels as we decrease.

To achieve this, we first prune a single attention and MLP module to different sparsities and measure the required forward pass time for the respective sparsity on a fixed device. Here, we make the simplifying assumption that the runtime for modules in different layers coincides. For the attention module, we measure the runtime of the attention module with $0, 1, \ldots, \boldsymbol{N}_{head}$ heads pruned and obtain a runtime-sparsity table:

$$\mathbf{T}_{attn} = [(\boldsymbol{t}_0, \boldsymbol{s}_0), \ldots, (\boldsymbol{t}_{\boldsymbol{N}_{head}}, \boldsymbol{s}_{\boldsymbol{N}_{head}})] \quad (5)$$

Given a number of levels $\boldsymbol{N}_{level}$, we use $\frac{\boldsymbol{t}_0}{\boldsymbol{N}_{level}}$ as the fixed absolute runtime difference between two levels and obtain a list of targeted absolute level speedups:

$$\mathbf{L} = [\boldsymbol{t}_0, \ldots, \boldsymbol{t}_0 - \frac{\boldsymbol{t}_0}{\boldsymbol{N}_{level}} \cdot (\boldsymbol{N}_{level} - 1)] \quad (6)$$

Next, for each of these levels, we find the closest runtime in $\mathbf{T}_{attn}$. Finally, we obtain the speedup-level table

$$\hat{\mathbf{T}}_{attn} = \{(\boldsymbol{t}_{i_0}, \boldsymbol{s}_{i_0}), \ldots, (\boldsymbol{t}_{i_{\boldsymbol{N}_{level}-1}}, \boldsymbol{s}_{i_{\boldsymbol{N}_{level}-1}})\},$$

where i_j is chosen such that the speedup corresponding to the i_j'th entry of $\mathbf{T}_{attn}$ is the best proxy of $L[j]$. We follow the same methodology to compute the speed-level table $\hat{\mathbf{T}}_{mlp}$ for the MLP module. Finally, we prune every layer to the sparsities indicated by the speedup-level table.

3.4 Fitness Environment

Finding a suitable fitness environment is fundamental to evolutionary search. Although models

Algorithm 1 *DarwinLM*: Evolutionary Search with Group-wise and In-group Mutation.

Given the number of group-wise mutation steps $\boldsymbol{N}_{wise}$, the number of in-group search step $\boldsymbol{N}_{in}$, and the number of offspring of per generation λ:

Speed Level Database Generation
$D_s \leftarrow$ databaseGen()

Evolutionary Search
Initialization:
parent $\leftarrow$ uniform()

Optimization:
for $t \leftarrow 1$ to $N_{\text{wise}} + N_{\text{in}}$ **do**
  candidates $\leftarrow$ [parent]
  for $i \leftarrow 1$ to λ **do**
    if $t \leq N_{\text{wise}}$ **then**
      offspring $\leftarrow$ groupMutation(parent)
    else
      offspring $\leftarrow$ inGroupMutation(parent)
    end if
    candidates.append(offspring)
  end for
  candidateModels $\leftarrow$ stitch(candidates, D_s)
  parent $\leftarrow$ selectFittest(candidateModels)
end for
return: parent

are typically evaluated based on their performance on downstream tasks, this approach is impractical in our context due to the lengthy evaluation times and the risk of overfitting. As an alternative, we adapt perplexity on a small-size calibration data as a metric to evaluate the performance of a candidate. Given a speedup-level database as described in the previous section, every possible model in our search space is defined by its module-wise speedup levels. Therefore, by assuming $\boldsymbol{M}$ to be an $\boldsymbol{n}$-tuple over the set of speedup levels $[\boldsymbol{N}_{level}]$, we can rewrite our objective function as

$$\hat{\boldsymbol{M}} = \arg\min_{\boldsymbol{M}} \text{Perplexity}(\boldsymbol{M}) + \theta(\boldsymbol{M}) \quad (7)$$

subject to $\boldsymbol{T}(\hat{\boldsymbol{M}}) \leq \boldsymbol{T}_{base}/\boldsymbol{S}$, where $\boldsymbol{\theta}(\cdot)$ is a regularization term for the candidate. In our experiments, we tried 1) penalizing extremely high sparsities of each element in $[\boldsymbol{N}_{level}]^{\boldsymbol{n}}$, where $\boldsymbol{\theta} = \boldsymbol{\beta} \sum_i^{\boldsymbol{N}} \exp^{\boldsymbol{\alpha} \cdot \boldsymbol{l}_i}$ and 2) penalizing the sparsity increase at the front layers, where $\boldsymbol{\theta} = \boldsymbol{\gamma} \sum_i^{\boldsymbol{N}} \boldsymbol{l}_i \times \exp^{\epsilon \cdot i}$. We provide an ablation study for different regularizer choices in Sec 4.3.

3.5 Evolutionary Search Algorithm

We approach the problem in Equation (1) via evolutionary search. The algorithm, described in Algorithm 1, includes initialization, group-wise and in-group mutation, and offspring selection.

Initialization. We first generate the speedup level database composed of pruned layers for different sparsities as discussed in Sections 3.2 and 3.3. Initially, our search algorithm starts with uniform compression. For example, if the max level is 10 and the target speedup is 2.0×, then the initial level for all of the attention and MLP modules is 5.

Group-wise and In-group Mutation. In each generation, λ offspring are generated by applying a speedup-preserving mutation. We propose a group-wise and in-group mutation strategy to enhance the offspring generation. Given a predefined group number $\boldsymbol{n}_g$, we partition the $\boldsymbol{n}$-tuples describing a compression configuration into $\boldsymbol{n}_g$ groups with $\frac{\boldsymbol{N}}{\boldsymbol{n}_g}$ elements in each group. For the *group-wise mutation*, one group is chosen uniformly at random and all speedup levels across this group are increased. Then, another group is chosen uniformly at random and all speedup levels across this group are decreased. This way, capacities can be shifted quickly towards different depths of the model. After $\boldsymbol{N}_{wise}$ generations, we switch the mutation operator to *in-group mutation*. Here, in each generation, we restrict the exchange of compression levels to within these groups.

Selection Process. After generating a set of offspring candidates, we construct a model for each candidate by stitching together components from the speedup level database. All candidate models are then evaluated on our fitness function. To ensure a robust and efficient selection, we employ a multi-step selection process, following the approach outlined in prior work (Anonymous, 2024). Specifically, the selection is performed iteratively over three rounds. In each round, a progressively smaller subset of offspring is retained, while the number of samples for fitness evaluation is increased. The final surviving candidate is selected as the starting point for the next generation.

4 Experiments

4.1 Setup

Models and Datasets. Given a target speedup and a set of pre-trained weights, our method searches for combinations of per-layer sparsity levels under the speedup constraint, based on a small generic calibration set. In our experiments, we consider Llama-2-7B (Touvron et al., 2023) and Llama-3.1-8B (Dubey et al., 2024). We utilize the open-source dataset Fineweb-Edu (Lozhkov et al., 2024) for both calibration and post-training. The dataset is filtered according to the sample score provided with the dataset. All samples with a lower score than 0.9 are removed from the dataset, resulting in a dataset

Table 1: Comparison of main results for *DarwinLM* and baseline methods on LLaMA2-7B. Our method achieves the best average performance on benchmarks compared to the baseline methods. With only 10B tokens of fine-tuning, our method beats ShearedLlama, which is fine-tuned with 50B tokens. (†) refers to training on the same 10B tokens we use. We omit MMLU results in this table as they are close to random for all Llama2 models.

Method	**Speedup**	**Param.**	**SciQ**	**PIQA**	**WG**	**ArcE**	**ArcC**	**HS**	**LogiQA**	**BoolQ**	**Avg**
Dense	1x	6.7B	93.7	78.1	69.3	76.4	53.0	78.6	30.7	82.1	70.2
Uniform	2×	3.4B	44.1	57.1	53.3	33.5	32.2	27.3	25.0	49.0	40.1
ZipLM	1.5×	4.0B	87.4	64.4	58.3	53.2	33.6	50.1	25.5	63.6	54.5
ShearedLLaMA	2.24×	2.7B	84.5	66.4	53.4	49.8	28.4	47.6	**27.6**	50.9	51.0
DarwinLM (one-shot)	2.5×	2.6B	**89.3**	**70.8**	**58.7**	**66.5**	**40.4**	**52.7**	25.1	**68.9**	**59.1**
ShearedLLaMA (50B)	2.24×	2.7B	90.8	75.8	64.2	67.0	41.2	70.8	28.9	73.7	64.0
ShearedLLaMA (10B†)	2.24×	2.7B	92.0	73.6	63.1	69.8	42.3	64.4	29.3	72.1	63.2
DarwinLM (10B)	2.5×	2.6B	**92.4**	73.7	63.5	**72.7**	**45.3**	67.2	28.8	72.2	**64.5**

with 80B tokens. For the search process, we use at most 16 token sequences for calibration, making this process highly lightweight.

Unless otherwise specified, we prune the models in a gradual manner. On Llama-2-7B, we first prune the model with a 2× target speedup in one-shot using 20K calibration samples and fine-tune the sparse model on 10B tokens. We continue to prune the model to 2.5× target speedup based on the fine-tuned model with 2K calibration data. We use the same procedure for Llama-3.1-8B, with the difference that Llama-3.1-8B is pruned to 1.5× and then 2.0×. The final pruned models are trained on an additional 10B Fineweb tokens.

Baselines. First, we compare our non-uniform sparse model with a uniform sparse model under a similar computational budget. Additionally, on Llama-2-7B, we conduct comparisons with ZipLM (Kurtić et al., 2024) and ShearedLlama (Xia et al., 2024). ZipLM employs dynamic programming to search for the sparse model structure, while ShearedLlama learns pruning masks for Llama-2-7B's weights and applies large-scale fine-tuning on 50B tokens. We perform evaluation using the publicly available weights after pruning and fine-tuning, as provided by the respective papers. For ZipLM, we reproduce their implementation at a larger scale, following the original paper's methodology. We limit our comparison with ShearedLlama to Llama-2-7B, as the original paper only reports results for this model, and the tuning costs for adapting it to other models are substantial. We also compare *DarwinLM* in a one-shot setting against other one-shot structured pruning methods, including EvoPress (Anonymous, 2024), ShortGPT (Men et al., 2024), Shortened Llama (Kim et al., 2024), and a block scoring approach using sliding window cosine similarity (Gromov et al., 2024). All of these methods perform structured pruning on a per-module or per-layer level.

Evaluation. We follow ShearedLlama (Xia et al., 2024) to evaluate our method on several downstream tasks including 0-shot accuracy on ARC-easy (Clark et al., 2018), LogiQA (Liu et al., 2020), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), 5-shot on MMLU (Hendrycks et al., 2020) and WinoGrande (Sakaguchi et al., 2021), 10-shot on HellaSwag (Zellers et al., 2019) and 25-shot on ARC Challenge (Clark et al., 2018). We utilize the *lm-evaluation-harness* framework (Gao et al.) to evaluate all downstream tasks.

Implementation Details. For evolutionary search, we apply group-wise search with $\boldsymbol{n}_g = 4$ for $\boldsymbol{N}_{wise} = 20$ steps and $\boldsymbol{n}_g = 8$ for $\boldsymbol{N}_{wise} = 80$ steps, respectively. After that, in-group search is conducted for $\boldsymbol{N}_{in} = 200$ steps. For each generation, we generate $\boldsymbol{\lambda} = 16$ offspring for selection. We set $\boldsymbol{\alpha} = 0.5, \boldsymbol{\beta} = 0.01, \boldsymbol{\gamma} = 0.2, \epsilon = -5$, respectively. The pruning and search process is conducted on a 4× L40s GPU workstation. Our training code is based on the LLM-Foundry repository. Our batch size is 1,024 for Llama-2 and 2,304 for Llama-3.1. The base learning rate is 1e-4 with a cosine decay strategy. The time profiling utilized to generate speedup level database is conducted on a single A6000 machine. The post-training with 10B token is operated on a 40× H100 cluster for 13 hours. More details can be found in the Appendix.

4.2 Main Results

Results on Llama-2-7B. We prune the Llama-2-7B model down to 2.6B with a 2.5× speedup. The main results are shown in Table 1. For the pruned models, our method achieves the highest performance on all downstream tasks, except for LogiQA, where the difference falls within the noise margin. Our method also attains the highest average score. In contrast, the uniform pruning method results in

Table 2: Comparison of results for *DarwinLM* and baseline models on Llama-3.1-8B. With similar speedup, our method achieves the best performance on all benchmarks compared with uniform pruning and ZipLM. After post-training with 10B tokens, the performance recovers from 54.0 to 58.1.

Method	Speedup	Param.	SciQ	PIQA	WG	ArcE	ArcC	HS	LogiQA	BoolQ	MMLU	Avg
Dense	1x	8B	96.3	81.2	74.3	81.4	58.2	81.7	31.1	84.0	65.2	72.8
Uniform	2×	4.4B	29.1	53.6	51.7	26.0	23.6	27.1	25.5	62.1	25.7	36.1
ZipLM	1.5×	6B	65.5	60.6	56.0	40.2	36.2	34.4	28.1	63.0	27.9	45.7
DarwinLM (one-shot)	2×	4.4B	**89.2**	**69.2**	**56.4**	**63.4**	**36.4**	**49.1**	**28.2**	**64.9**	**29.5**	**54.0**
DarwinLM (10B)	2×	4.4B	91.8	73.0	62.5	67.3	44.8	60.6	29.0	63.8	30.1	58.1

a significant performance drop, with an average accuracy of only 38.3 and a 2× speedup, essentially a performance collapse compared to the dense model. Specifically, the uniformly-pruned model generates nearly random results on benchmarks such as WinoGrande, HellaSwag, LogiQA, BoolQ, and MMLU. By contrast, *DarwinLM* achieves an average score of 59.1 with a 2.5× speedup, outperforming ZipLM (54.5 with a 1.5× speedup) and ShearedLlama (51.0 with a 2.24× speedup). This comparison highlights the effectiveness of non-uniform structured pruning, particularly at high sparsity.

After post-compression training, the pruned models see a significant recovery in performance. Notably, with only 10B tokens for training, *DarwinLM* reaches an average score of 64.5, surpassing the 64.0 reported by ShearedLlama, which was trained with 50B tokens. Furthermore, when we train the pruned model released by ShearedLlama under the same conditions and with 10B tokens, it achieves an average score of 63.2, which is considerably lower than *DarwinLM*. Figure 2 presents the training performance curve of *DarwinLM* and ShearedLlama, showing the average performance on benchmarks evaluated every 2B tokens. More detailed results for individual tasks can be found in the Appendix. Overall, with the same amount of training data, *DarwinLM* consistently outperforms ShearedLlama. Interestingly, with just 2B tokens, *DarwinLM* produces comparable results to ShearedLlama trained on 10B tokens. The curve clearly demonstrates that *DarwinLM* achieves superior performance to ShearedLlama regardless of the number of training tokens used.

Results on Llama-3.1-8B. We also pruned the Llama-3.1 8B model to 4.4B parameters with a target speedup of 2×. The comparison results are shown in Table 2. Similar to Llama-2-7B, the uniformly pruned Llama-3.1-8B model suffers catastrophic degradation. For example, the uniformly pruned model achieves 26.0, 23.6, and 27.1 on ARC-E, ARC-C, and HellaSwag, respectively, close to randomly generated results

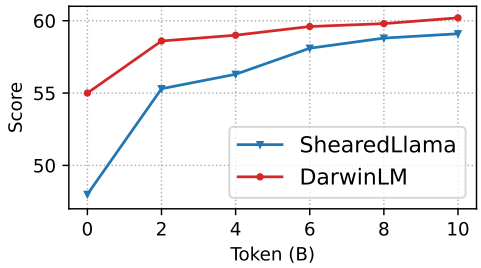


Figure 2: Performance comparison of *DarwinLM* and ShearedLlama with different training token numbers. *DarwinLM* achieves better performance than ShearedLlama on all training token number settings.

Table 3: One-shot result comparison of the uniformly pruned model and *DarwinLM* on Llama-2-7B and Llama-3.1-8B. *DarwinLM* outperforms uniform models on both pre-trained models, demonstrating the importance of non-uniform sparsity allocation.

Model	Method	Param.	PIQA	WG	ArcC	HS
LLaMA-2-7B	Uniform	3.3B	57.1	53.3	32.2	27.3
	DarwinLM	3.3B	**68.4**	**59.1**	**36.5**	**51.6**
LLaMA-3.1-8B	Uniform	5.9B	54.0	51.8	25.0	27.8
	DarwinLM	5.9B	**76.1**	**64.2**	**45.7**	**64.4**

(25.0%). In contrast, *DarwinLM* significantly improves performance, achieving 63.4, 36.4, and 49.1 on these datasets. Overall, *DarwinLM* shows the best average performance compared to both the uniformly pruned and ZipLM models. After the post-compression training, *DarwinLM* recovers performance across all benchmarks, with an average score of 58.1. An interesting observation is that Llama-3.1-8B is more challenging to prune at higher sparsity than Llama-2-7B. For example, with the same speedup, the uniformly pruned Llama-2-7B outperforms Llama-3.1-8B, with scores of 38.3 vs. 36.1. Similarly, the pruned Llama-2 model with a 2.5× speedup outperforms the performance of Llama-3.1 with a 2× speedup. We attribute this to the compression constraint imposed by GQA (Ainslie et al., 2023), as we leave the K and V matrices unpruned, leading to higher sparsity in the remaining matrices.

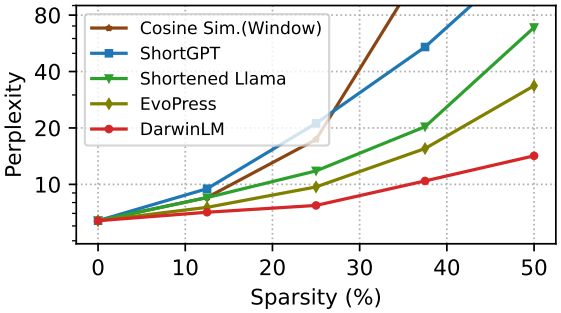


Figure 3: Comparison of *DarwinLM* and other one-shot methods that remove modules entirely. Our method consistently outperforms across all sparsity levels, demonstrating the effectiveness of our finer-grained structured pruning approach. Note that the y-axis is log-scaled.

4.3 Analysis

Uniform vs. Non-uniform Pruning. We now focus on the performance comparison of the uniformly sparse model with the non-uniform model, as shown in Table 3. The one-shot pruning results without post-training are presented for both the uniform model and *DarwinLM* on Llama-2-7B and Llama-3.1-8B. For Llama-2-7B, we apply a 2× speedup, while for Llama-3.1-8B, we target a 1.5× speedup. Across all benchmarks, the one-shot uniform sparse model performs close to random. In contrast, the non-uniform sparse model, *DarwinLM*, shows significant improvement. For example, on HellaSwag, the uniform model achieves only 27.3 on Llama-2 and 27.8 on Llama-3.1, while *DarwinLM* scores 51.6 and 64.4, respectively. On the ARC-C dataset, *DarwinLM* outperforms the uniform model with scores of 36.5 and 45.7 for Llama-2 and Llama-3.1, compared to 32.2 and 45.7 for the uniform model. These results clearly demonstrate the effectiveness of non-uniform sparsity in structured pruning for large language models. A full list of results is available in the Appendix.

Comparison with One-shot Methods. We further compare *DarwinLM* with several current one-shot structured pruning (layer dropping) methods including EvoPress(Anonymous, 2024), ShortGPT (Men et al., 2024), Shortened Llama (Kim et al., 2024), and a method bases on sliding window cosine similarity (Gromov et al., 2024) on Llama-2-7B. We select 40 samples with 4,096 tokens from Fineweb-Edu as the test set and compute the perplexity of each model under different sparsity. For *DarwinLM*, we choose the target speedups with the closest sparsity settings for comparison. The comparison results are shown in Figure 3. First, we can observe that even if all pruning methods can preserve well performance under the sparsity of 25%, *DarwinLM* still achieves lower perplexity compared with other one-shot pruning methods. Moreover, the performance of ShortGPT and Sliding Window Cosine Similarity shows a dramatic degradation after 25% sparsity while the perplexity of Shortened Llama and *DarwinLM* increases slightly before 40%. However, EvoPress also degrades to more than 30 perplexity while *DarwinLM* shows a minor increase for 50% sparsity. Generally, *DarwinLM* outperforms all other one-shot methods under different sparsity settings and keeps stable performance with the increase of sparsity, demonstrating the effectiveness of our method. This is natural also since our method benefits from higher compression granularity.

Table 4: Ablation study of our proposed group-wise (GW) and in-group (IG) search, and different regualizers on Llama-2-7B with 2× speedup.

Model	PIQA	ArcC	ArcE
Uniform	57.1	33.5	32.2
DarwinLM	**68.4**	**59.1**	36.5
DarwinLM w/o GW and IG search	60.5	37.5	33.0
DarwinLM w/o Both Penalties	61.2	39.6	36.1
DarwinLM w/o Extreme Penalty	61.9	41.1	**37.0**
DarwinLM w/o Front Layer Penalty	62.5	39.5	36.2

Ablation Study. Finally, we examine the impact of group-wise and in-group search as well as the influence of different regularizers. The results, presented in Table 4, show a clear trend: without group-wise and in-group search, performance drops sharply; for example, the ARC-E score falls from 59.1 to 37.5. Additionally, we assess the effect of each regularizer individually and in combination. Overall, performance is highest when both regularizers are applied together. Each regularizer contributes to improved results on downstream tasks, and their combination achieves the best performance across all benchmarks.

5 Conclusion

In this paper, we introduced a novel non-uniform, speed-aware structured pruning method called *DarwinLM*, which guarantees speedup by producing speed-aware offspring as part of the evolutionary search and generating non-uniform sparse models. We propose novel group-wise and in-group mutation strategy to further accelerate the search process. Experiments on Llama-2-7B and Llama-3.1-8B demonstrate that our approach achieves state-of-the-art performance even with higher sparsity compared to previous structured pruning techniques.

6 Limitations

While our non-uniform pruning of LLMs using a speed-aware evolutionary search algorithm offers several advantages, it also has certain limitations: 1) The speed-aware algorithm focuses on optimizing both model performance and inference speed, but this dual-objective approach may result in suboptimal performance for certain tasks. Focusing on speed can sometimes compromise the accuracy or robustness of the model. 2) The fine-tuning process still requires substantial computational resources, especially when applied to LLMs with billions of parameters.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2024. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Anonymous. 2024. Evopress: Towards optimal dynamic model compression via evolutionary search. In *Submitted to The Thirteenth International Conference on Learning Representations*. Under review.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

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 preprint arXiv:1803.05457*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2023. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Elias Frantar and Dan Alistarh. 2022. Spdy: Accurate pruning with speedup guarantees. In *International Conference on Machine Learning*, pages 6726–6743. PMLR.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

L Gao, J Tow, B Abbasi, S Biderman, S Black, A DiPofi, C Foster, L Golding, J Hsu, A Le Noac'h, et al. A framework for few-shot language model evaluation, 12 2023. *URL https://zenodo. org/records/10256836*, 7.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. 2024. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*.

Babak Hassibi and David Stork. 1992. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5.

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*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *Preprint*, arXiv:1503.02531.

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*.

Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and XIAOJUAN QI. 2024. Billm: Pushing the limit of post-training quantization for llms. In *Forty-first International Conference on Machine Learning*.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. 2024. Shortened llama: A simple depth pruning for large language models. *arXiv preprint arXiv:2402.02834*.

Aaron Klein, Jacek Golebiowski, Xingchen Ma, Valerio Perrone, and Cédric Archambeau. 2023. Structural pruning of large language models via neural architecture search. In *AutoML Conference 2023*.

Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. 2022. The optimal bert surgeon: Scalable and accurate second-order pruning for large language models. *arXiv preprint arXiv:2203.07259*.

Eldar Kurtić, Elias Frantar, and Dan Alistarh. 2024. Ziplm: Inference-aware structured pruning of language models. *Advances in Neural Information Processing Systems*, 36.

Shiyao Li, Xuefei Ning, Luning Wang, Tengxuan Liu, Xiangsheng Shi, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. 2024. Evaluating quantized large language models. In *Forty-first International Conference on Machine Learning*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100.

Chengyuan Liu, Fubang Zhao, Kun Kuang, Yangyang Kang, Zhuoren Jiang, Changlong Sun, and Fei Wu. 2024. Evolving knowledge distillation with large language models and active learning. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. 2024. Fineweb-edu.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. 2024. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. LLM-pruner: On the structural pruning of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. 2024. Compact language models via pruning and knowledge distillation. *arXiv preprint arXiv:2407.14679*.

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.

V Sanh. 2019. Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Chaofan Tao, Lu Hou, Haoli Bai, Jiansheng Wei, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. 2023. Structured pruning for efficient generative pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10880–10895.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. 2023. Bitnet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2020. Structured pruning of large language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6151–6162.

Johannes Welbl, Nelson F Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*.

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

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

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

Yuzhuang Xu, Xu Han, Zonghan Yang, Shuo Wang, Qingfu Zhu, Zhiyuan Liu, Weidong Liu, and Wanxiang Che. 2024b. Onebit: Towards extremely low-bit large language models. *arXiv preprint arXiv:2402.11295*.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Mykola Pechenizkiy, Yi Liang, Zhangyang Wang, and Shiwei Liu. 2023. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. *arXiv preprint arXiv:2310.05175*.

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*.

A Appendix

A.1 More Implementation Details.

Details of On-device Time Profiling. The on-device time profiling needs to be conducted within the target inference environment to ensure accurate and relevant measurements. In our experiment, we perform the time profiling on a single A6000 machine, which serves as our target hardware for evaluation. For the Multi-head Self-Attention (MHSA) module, we record the time taken by the attention mechanism with different numbers of attention heads, varying the configuration to assess performance under different settings. For the MLP module, we adjust the intermediate size by shrinking it according to a scaling factor of $0.9^i, i \in range(0, 42)$, following the approach in (Kurtić et al., 2024). This helps evaluate how the performance changes as the size of the MLP is progressively reduced. For Llama-2-7B, we use a sequence length of 4,096 tokens to calculate the inference time, while for Llama-3.1-8B, we use a longer sequence of 8,092 tokens to account for its larger model size and complexity.

We use a maximum of 10 speedup levels, where each level corresponds to a particular running time and its associated sparsity configuration. This results in a speedup-level database with 10 different configurations. Additionally, we employ FlashAttention (Dao et al., 2022) to further optimize the time profiling process and ensure efficient attention computations.

Details of Second-Order Structured Pruning. We utilize 20K sequences with 4,096 tokens from the Fineweb-Edu dataset as calibration data for both Llama-2-7B and Llama-3.1-8B. In the attention module, we fully prune the attention heads, and in the MLP module, we prune entire columns of the output matrix. For Llama-2-7B, we prune the input matrix, as well as the Q, K, and V matrices, based on the pruned output matrix in the attention module. For Llama-3.1-8B, we ignore the K and V matrix for pruning. The input and gate matrices in the MLP module are pruned according to the output matrix. On average, pruning both Llama-2-7B and Llama-3.1-8B requires 4× 48GB GPU memory. Most of the second-order structured pruning experiments are conducted on a 4× L40S machine with 48GB GPU memory.

Details of Evolutionary Search. For group-wise mutation, we first set the group size as 4 for 20 steps and changed it to 8 groups for the next 80 generations. After the group-wise mutation for an overall 100 steps, we do the in-group mutation with 8 groups for 200 steps. During the offspring generation, the number of mutations to be done for each generation is influenced by the group size. We randomly sample the number of mutations from $min(randint(1, \frac{group_size}{2}), randint(1, \frac{group_size}{2}))$ This adapts the mutation size with the group size and makes the search more stable. The search process can be done on a single GPU with 48GB of memory.

Table 5: Hyper-parameter details for post-training on *DarwinLM*-2.6B and *DarwinLM*-4.4B.

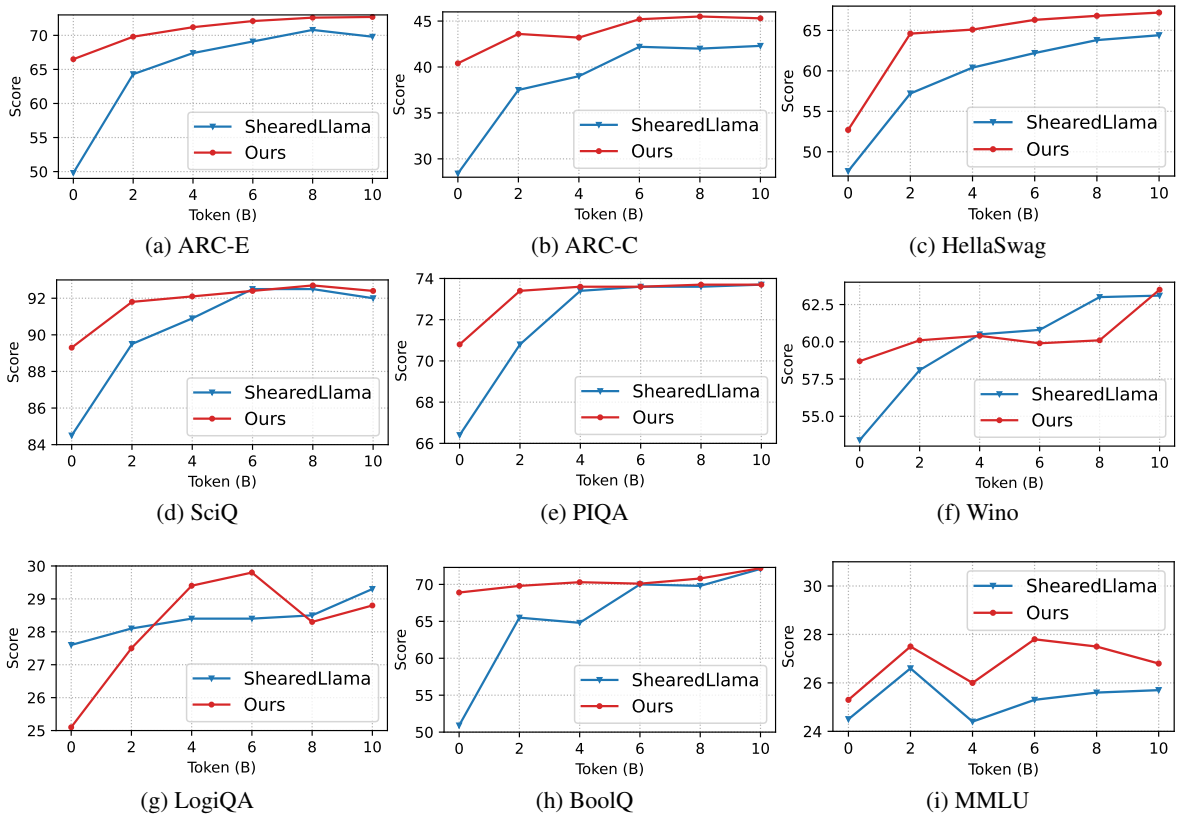
Parameter	***DarwinLM*-2.6B**	***DarwinLM*-4.4B**
Learning rate	1e-4	1e-4
Global batch size	1,024	2,304
Warm-up steps	50 steps	10 steps
LR decay scheduler	Cosine	Cosine
Context length	4,096	8,192
Overall tokens	10B	10B

Details of Post-Training. We train the final 2.6B sparse model pruned from Llama-2-7B and the 4.4B model pruned from Llama-3.1-8B using 10B tokens for each. Gradient accumulation is applied to achieve a larger global batch size. The models are trained with the Adam optimizer, using a learning rate of 1e-4, and a cosine learning rate decay scheduler to ensure gradual reductions for stable training. No weight decay is applied. The training process is conducted on a cluster of 40 H100 GPUs for 13 hours. Detailed hyperparameters for post-training can be found in Table 5.

A.2 More Experimental Results.

More results of the uniform and non-uniform sparse model. We present a full comparison of uniform and non-uniform sparse models across each benchmark in Table 7. For both Llama-2-7B and Llama-3.1-8B, *DarwinLM* consistently outperforms the uniform pruning approach. Interestingly, we observe that the uniformly pruned Llama-2-7B model performs better than *DarwinLM* on the LogiQA dataset. We conjecture this is due to the challenging nature of LogiQA, where even the dense model only achieves a score of 30.7. However, on stronger dense models like Llama-3.1-8B, *DarwinLM* surpasses the uniformly pruned model in performance on LogiQA.

More Results of Post-Training Comparison with ShearedLlama. We provide the post-training comparison of ShearedLlama across all benchmarks, with the performance trends for each dataset available in 4. In most cases, *DarwinLM* outperforms



(a) ARC-E (b) ARC-C (c) HellaSwag

(d) SciQ (e) PIQA (f) Wino

(g) LogiQA (h) BoolQ (i) MMLU

Figure 4: Post-training comparison of ShearedLlama and *DarwinLM* on each benchmark.

Table 6: Comparison of *DarwinLM* with uniform pruning on Llama-2-7B.

Method	Speedup	Param.	SciQ	PIQA	WG	ArcE	ArcC	HS	LogiQA	BoolQ	MMLU	Avg
Dense	1×	6.7B	93.7	78.1	69.3	76.4	53.0	78.6	30.7	82.1	46.6	67.6
Uniform	2×	3.3B	44.1	57.1	53.3	33.5	32.2	27.3	**25.0**	49.0	23.7	38.4
DarwinLM	2×	3.3B	**85.5**	**68.4**	**59.1**	**63.2**	**36.5**	**51.6**	23.9	**62.4**	**24.8**	**52.8**

Table 7: Comparison of *DarwinLM* with uniform pruning on Llama-3.1-8B.

Method	Speedup	Param.	SciQ	PIQA	WG	ArcE	ArcC	HS	LogiQA	BoolQ	MMLU	Avg
Dense	1×	8B	96.3	81.2	74.3	81.4	58.2	81.7	31.1	84.0	65.2	72.8
Uniform	1.5×	5.9B	38.5	54	51.8	27.4	24.9	27.8	27.5	57.9	26.5	37.3
DarwinLM	1.5×	5.9B	**92.9**	**76.1**	**64.2**	**75.6**	**45.7**	**64.4**	**29.6**	**70.8**	**45.5**	**62.7**

ShearedLlama in benchmark evaluations, including SciQ, PIQA, ARC-E, ARC-C, HellaSwag, Wino-Grande, LogiQA, BoolQ, and MMLU, demonstrating the effectiveness of our approach.