Pruner-Zero: Evolving Symbolic Pruning Metric from scratch for Large Language Models

Peijie Dong [1] [*] Lujun Li [2] [*] Zhenheng Tang [3] [1] Xiang Liu [1] Xinglin Pan [1] Qiang Wang [4] Xiaowen Chu [1] [2]

Abstract

Despite the remarkable capabilities, Large Language Models (LLMs) face deployment challenges due to their extensive size. Pruning methods drop a subset of weights to accelerate, but many of them require retraining, which is prohibitively expensive and computationally demanding. Recently, post-training pruning approaches introduced novel metrics, enabling the pruning of LLMs without retraining. However, these metrics require the involvement of human experts and tedious trial and error. To efficiently identify superior pruning metrics, we develop an automatic framework for searching symbolic pruning metrics using genetic programming. In particular, we devise an elaborate search space encompassing the existing pruning metrics to discover the potential symbolic pruning metric. We propose an opposing operation simplification strategy to increase the diversity of the population. In this way, Pruner-Zero allows auto-generation of symbolic pruning metrics. Based on the searched results, we explore the correlation between pruning metrics and performance after pruning and summarize some principles. Extensive experiments on LLaMA and LLaMA-2 on language modeling and zero-shot tasks demonstrate that our Pruner-Zero obtains superior performance than SOTA post-training pruning methods. Code at: https://github.com/pprp/Pruner-Zero.

1. Introduction

Recent breakthroughs in Large Language Models (LLMs) (OpenAI, 2023; Touvron et al., 2023a;b; Abdin et al., 2024) have revolutionized the field of Natural Language Processing (NLP) tasks, enabling remarkable progress across a wide spectrum of tasks, including both Natural Language Understanding and Natural Language Generation. A key differentiator of LLMs from prior convolution-based models is their unprecedented scale. With billions of parameters, LLMs can effectively manage complex tasks and have exhibited exceptional performance. However, the vast number of parameters that these models possess necessitates substantial computational resources, highlighting a critical challenge in their deployment. For instance, deploying models like GPT-3, with its 175 billion parameters, underscores the intense computational demands, necessitating advanced GPU technologies.

Several model compression techniques are developed to solve these problems, such as model quantization (Frantar et al., 2023; Liu et al., 2023b; Yao et al., 2022), model sparsing (Frantar & Alistarh, 2023; Sun et al., 2024), and knowledge distillation (Xiaolong et al., 2023; Li & Jin, 2022; Li, 2022). Model sparsing compresses the model by identifying and eliminating redundant elements in the weight matrix. Model sparsing is one of the most promising techniques for model deployment due to its flexibility. Previous research works on model sparsing require training from random initialization (Hoang et al., 2023; Sreenivasan et al., 2022; Louizos et al., 2018), retraining process (Liu et al., 2019; Chen et al., 2023) or extensive iterative pruning (Chijiwa et al., 2021; Tanaka et al., 2020). However, the inherent complexities, along with the substantial computational and data requirements of LLMs, present significant challenges that render these conventional sparse strategies less feasible.

Given the extensive data corpus and substantial model dimensions required by LLMs, post-training pruning has become an increasingly crucial methodology. Due to its minimal resource demands, this approach is highly advantageous, offering a cost-efficient alternative for optimizing LLMs. The development of the post-training pruning method, as highlighted in recent studies (Lu et al., 2022; Frantar & Alistarh, 2023; Sun et al., 2024), marks a significant advancement in this field. These methods streamline the pruning process, further reducing the resource requirements and making LLMs more democratized and accessible. SparseGPT (Frantar & Alistarh, 2023) is proposed to con-

---

[*]Equal contribution [1]The Hong Kong University of Science and Technology (Guangzhou) [2]The Hong Kong University of Science and Technology [3]Hong Kong Baptist Univeristy [4]Harbin Institude of Technology, Shenzhen. Correspondence to: Qiang Wang <qiang.wang@hit.edu.cn>, Xiaowen Chu <xwchu@ust.hk>.

Table 1. The existing pruning metrics tailored for LLMs. "W" denotes weight update, and "C" denotes the calibration data. "UOP" denotes unary operations, and "BOP" denotes binary operations. σ denotes the min-max scaling operation.

Method	W	C	Pruning Metric $\mathcal{S}$	UOP	BOP
Magnitude (Han et al., 2016a)	✗	✗	$\|\mathbf{W}\|$	$\| \cdot \|$	$\emptyset$
SparseGPT (Frantar & Alistarh, 2023)	✓	✓	$\left[\|\mathbf{W}\|^2/\mathrm{diag}\left[\mathbf{H}^{-1}\right]\right]_{ij}$	$\| \cdot \|, (\cdot)^2$	$\div$
Wanda (Sun et al., 2024)	✗	✓	$\|\mathbf{W}_{ij}\| \cdot \|\mathbf{X}_j\|_2$	$\| \cdot \|, \|\| \cdot \|\|_2$	$\times$
GBLM-Pruner$_{l1}$ (Das et al., 2023)	✗	✓	$\|\mathbf{W}\| \cdot \|\|\mathbf{G}\|\|_1$	$\| \cdot \|, \|\| \cdot \|\|_1$	$\times$
GBLM-Pruner$_{l2}$ (Das et al., 2023)	✗	✓	$\|\mathbf{W}\| \cdot \|\|\mathbf{G}\|\|_2$	$\| \cdot \|, \|\| \cdot \|\|_2$	$\times$
Pruner-Zero	✗	✓	$\|\|\mathbf{W}\| \times \|\mathbf{W}\|\| \times \sigma(\|\mathbf{G}\|)$	$\| \cdot \|, \sigma(\cdot)$	$\times$

duct post-training pruning for LLMs, which allows for a significant reduction in the size of these models to prune the GPT family to 50% sparsity. Wanda (Sun et al., 2024) does not require retraining or weight updates, which makes the pruned LLM immediately ready for inference tasks. GBLM-Pruner (Das et al., 2023) further dives into the design of the pruning metric by emphasizing the importance of first-order information, a.k.a. gradient. As shown in Table 9, we identify two key challenges of the existing pruning metrics: (1) **Human-Dependence**: existing methods heavily rely on domain knowledge and thus require massive efforts of trial and error. (2) **Format-Sensitivity**: As illustrated in Table 9, pruning metrics demonstrate a pronounced sensitivity to their format, demanding a rigorous and systematic approach to experimentation. These points of contention raise pivotal inquiries:

*(1) How to **formulate and devise comprehensive pruning metrics** that encapsulate the strengths of existing ones?*

*(2) How to find the **optimal pruning metric** tailored for Large Language Models?*

To address the **first question**, we present **a comprehensive search space** through an exhaustive review of the existing pruning metrics, as detailed in Table 1. We meticulously dissect the structure of these metrics by analyzing their constituent inputs and operations. The inputs considered are Weight (W), Gradient (G), Hessian (H), and Activation (X), while the operations are categorized into unary or binary ones. Given the computational complexity of the Hessian metric, which scales quadratically with the dimensionality of the hidden layer, $(O(d_{hidden}^2))$, we opt to exclude it from our considerations within this paper. For symbolic operations, we collect the operation utilized in the existing pruning metrics to build the operation vocabulary in Table 10. Inspired by Symbolic Regression (SR), we observe that the pruning metrics are composed of a series of symbols, and thus can be represented as an expression tree, here we call it a symbolic pruning metric. We formulate these pruning metrics as **Symbolic Pruning Metrics (SPM)**, denoting them as a combination of symbolics. However, SR

is a complex combinatorial problem, as the search space for equations grows exponentially with the number of its operations, which comes to the second question.

In addressing the **second question**, we introduce **the Pruner-Zero framework**, which employs Genetic Programming (GP) to devise a symbolic pruning metric (SPM) encapsulated within a tree structure, as depicted in Figure 1. The terminal nodes represent variables such as Weight (W), Gradient (G), and Activation (X), while its internal nodes represent mathematical operations. This innovative method facilitates the organic growth of an expression tree that emulates the principles of biological evolution, complete with node mutations and subtree crossovers that effectively probe the diversity of potential pruning metrics. We conduct fast post-training pruning evaluations for each SPM for LLaMA-7B on the WikiText2 dataset to get the perplexity of its fitness in less than 5 minutes. However, we recognize the existence of correlations among certain operations, particularly those that are inversely related, which complicates the search space with multiple symbolic trees that, despite their differences, are mathematically equivalent. To counteract this challenge, we propose the Opposing Operation Simplification (OOS) strategy. This strategy is designed to pinpoint and streamline opposing patterns, thereby curbing the redundancy within the search space and enhancing the efficiency of the metric discovery process.

Upon identifying the most promising SPM candidate, we adapt it across a wider spectrum of LLM families, including LLaMA (Touvron et al., 2023a), LLaMA-2 (Touvron et al., 2023b), Tiny-LLaMA (Zhang et al., 2024a), and OPT (Zhang et al., 2022b). Utilizing this framework, we delve deeper into the relationship between low perplexity and the design of pruning metrics. This exploration involves a comprehensive analysis of the correlations among different operations. From this analysis, we can distill and summarize key principles for developing effective pruning metrics. These principles are in harmony with the insights gained from our previously identified SPM, providing a cohesive and informed approach to pruning metric design. Our contributions are summarized as follows:

1. We formulate the pruning metric discovery as a Symbolic Regression problem and propose a unified and comprehensive search space that encompasses existing pruning metrics (Sec. 3.1).

2. We introduce Pruner-Zero, the first Symbolic Pruning Metric search framework for the automated discovery of optimal pruning metrics tailored for LLMs utilizing Genetic Programming. We propose the Opposing Operation Simplification (OOS) strategy to reduce the redundancy in the search space (Sec. 3.2).

3. Our comprehensive experiments on language modeling and zero-shot tasks demonstrate that our Pruner-Zero can achieve SOTA performance without retraining or weight update for LLMs (Sec. 4).

4. We further investigate the correlation between the performance and the SPM design and uncover some principles in designing pruning metrics (Sec. 4.6).

2. Related Work

2.1. Language Model Pruning

Network pruning can significantly reduce model complexity while maintaining performance, albeit typically necessitating extensive retraining. Given the substantial parameter sizes and extensive datasets of LLMs, traditional pruning methods (Hoang et al., 2023; Sreenivasan et al., 2022; Liu et al., 2019; Chen et al., 2023; Chijiwa et al., 2021) become impractical. In this paper, we focus on the post-training pruning techniques.

Problem Statement. Post-training pruning serves as a practical scenario where we are given a well-optimized W and a symbolic pruning metric $\mathcal{S}$. Post-training compression is conducted by splitting the full-model compression problem into layer-wise sub-problems. Specifically, for post-training pruning, our objective is to find a sparsity mask (binary mask) M_l for each layer l with a certain target sparsity ratio to minimize the $l2$-error, formulated as:

$$\textbf{argmin}_{M_l}||W_l X_l - (M_l \odot W_l)X_l||_2^2 \qquad (1)$$

Here, the mask M_l is determined by symbolic pruning metric $\mathcal{S}$ through $M_l = f(\mathcal{S}, W_l, \phi)$. The function f ranks the weights in W_l according to their importance as per $\mathcal{S}(W_l, X_l, G_l)$, and then selects the most significant weights up to the sparsity ratio ϕ. For example, a basic pruning metric is magnitude (Han et al., 2016b), where $\mathcal{S}(W_l) = |W_l|$ employs the element-wise absolute value to assess weight significance. Thus, the key to post-training pruning lies in the symbolic pruning metric.

Post-training Pruning for LLMs. Deep Compression (Han et al., 2016a) proposed magnitude-based pruning, which eliminates weights with the smallest absolute

values, based on their minimal impact on network output. SparseGPT (Frantar & Alistarh, 2023) proposed a framework for post-training pruning that obviates the need for retraining by using Hessian matrices and calibration data to update weights. Complementing this, Wanda (Sun et al., 2024) streamlines the process by simplifying SparseGPT's methodology. Additionally, GBLM-Pruner (Das et al., 2023) employs the first-order term of the Taylor expansion, emphasizing the significance of gradients. Structured pruning methods, such as activation pruning and neuron/filter output statistics, are used for GPU acceleration. LLM-Pruner (Ma et al., 2023) examines model dependencies, incorporating both first-order and approximated Hessian information. LLM Surgeon (van der Ouderaa et al., 2024) adapts Kronecker-factored curvature approximations to LLMs, targeting 20%-25% low sparsity. In this paper, our focus is on the post-training pruning of language models without retraining or weight updates.

Efficient and Low Resource Compression. Due to the large size of language models, there is an increasing demand for efficient LLM compression without using the original training data. As for efficient compression, (Kwon et al., 2022) accelerates the post-training by defining the reconstruction error as a linear least squares problem. SparseGPT (Frantar & Alistarh, 2023) and GPTQ (Frantar et al., 2023) propose the layer-wise optimal brain surgeon. Due to the constraint of availability of the training corpus, data-free methods (Srinivas & Babu, 2015; Yvinec et al., 2021) prune the neural network by measuring the similarity of neurons. Most related to our approach is pruning with limited data, which requires no modification to the original training procedure and no retraining of the pruned network on the full training datasets. To mitigate the accuracy drop, a layer-wise reconstruction problem is involved to minimize the change of output evaluated on the calibration data.

2.2. Symbolic Regression

Symbolic Regression (SR) is a distinctive approach within genetic programming, aiming at discovering the analytical expression that accurately models a complex dataset without the need for pre-specified functional forms. This adaptive nature of SR allows it to dynamically evolve when fitting data, making it especially useful in scenarios where the underlying relationships are complex or unknown. Originating from genetic programming (Koza, 1994), which applies principles of natural selection to evolve computer programs, SR has evolved into an essential tool for modeling and prediction. It stands out for its flexibility in handling complex modeling tasks, surpassing traditional regression methods in adaptability and interpretability.

The integration of deep learning techniques with SR, particularly through approaches like Equation Learner (EQL) (Mar-
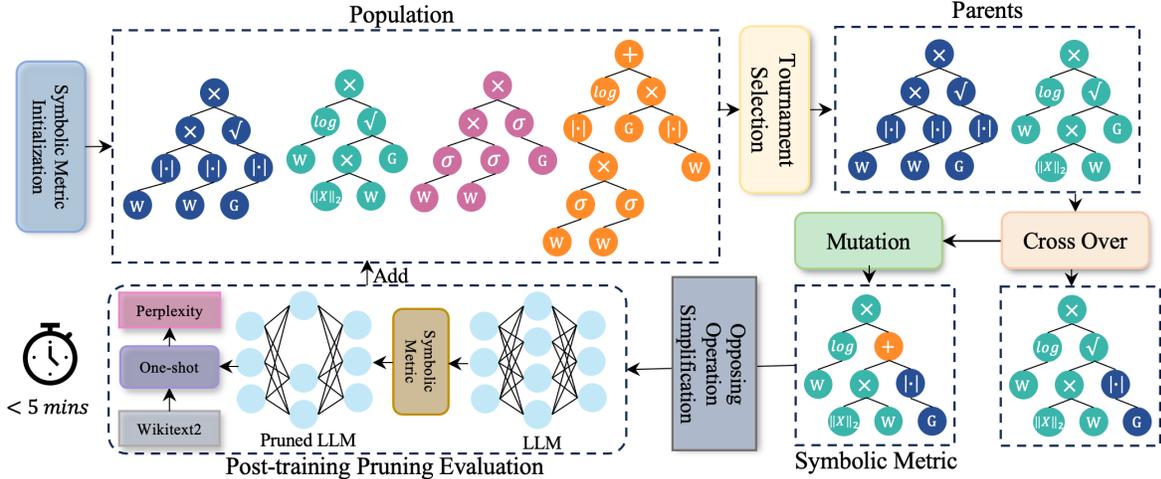
Figure 1. Overview of the Automatic Symbolic Pruning Metric Discovery Process in our Pruner-Zero framework. This process employs genetic programming to iteratively generate and refine symbolic pruning metrics via tournament selection, subtree crossover, and node mutation. Upon generating offspring, the Opposing Operation Simplification (OOS) strategy is applied to diminish repetition. Subsequently, evaluation is conducted on the LLaMA-2-7B using the WikiText2 dataset, with perplexity serving as the fitness metric. Note that it only takes less than 5 minutes to perform the post-training pruning evaluation.

tius & Lampert, 2016; Sahoo et al., 2018; Werner et al., 2021), marks a significant advancement in the field. EQL and similar methods employ neural networks, utilizing basic mathematical operations as activation functions to construct and train networks until a sparse and efficient structure is achieved. The resulting structure is then unrolled into a final symbolic expression. This synthesis of neural network capabilities with symbolic regression illustrates the potential for scalable and flexible model generation, enhancing the applicability of SR in various complex scenarios. Notably, in this paper, we pioneer the conceptualization of the pruning metric problem within the framework of symbolic regression.

3. Pruner-Zero Framework

In this section, we first introduce the search space design of pruning metrics and then present the details of the Pruner-Zero framework to search for symbolic pruning metrics.

3.1. Pruner-Zero Search Space Design

To ensure the effectiveness of Pruner-Zero, we devise a unified framework that encompasses and extends beyond existing pruning metrics. This framework's search space includes three types of inputs and 17 primitive operations, enabling us to reconstruct existing pruning metrics as detailed in Table 1 and Table 9.

LLM Statistics as Inputs. Our search space comprises three specific input types: activations (X), gradients (G), and weights (W). We exclude the Hessian matrix in SparseGPT (Frantar & Alistarh, 2023) due to its quadratic complexity. Given the substantial size of LLMs, we col-

lect and preprocess gradient information using 128 calibration samples and archive it locally. Consequently, during the search phase, only activation and weight are required, thereby expediting the search process.

Primitive Operations. Inspired by AutoML-Zero (Real et al., 2020) and EMQ (Dong et al., 2023b), we utilize a comprehensive suite of primitive operations that include unary and binary ones. These operations are informed by existing symbolic pruning metrics, as detailed in Table 1. The complete primitive operations are available in Appendix C.2, with an in-depth analysis provided in Sec. 4.6 and Appendix C.4.

Pruning Metric as Expression Tree. We define the pruning metrics, as outlined in Table 1, through a structured combination of operations and Large Language Model (LLM) statistics. In this framework, LLM statistics are situated at the leaf nodes, while primitive operations occupy the internal nodes. Binary operations are characterized by two child nodes, whereas unary operations possess a single child node, with the second node represented by a "#" placeholder. The root node, designated as a primitive operation, yields a singular output. Within this expression tree, the output's dimensions are required to align with those of the weights, thereby indicating the saliency of each weight.

3.2. Genetic Programming Framework

We conceptualize the identification of symbolic pruning metrics as a challenge within Symbolic Regression (SR), recognized as a complex combinatorial NP-hard problem (Virgolin & Pissis, 2022). This complexity is attributed to the search space for pruning metrics, which expands exponen-

Algorithm 1 Evolution Search for Pruner-Zero

Input: Search space $\mathcal{S}$, population size $|\mathcal{P}|$, sample ratio r, top-k k, selection ratio ρ, max iteration $\mathcal{N}$, tree depth d.
Output: Best symbolic tree with lowest Perplexity (PPL) after post-training pruning evaluation.

1: Initialize promising candidates $\mathcal{C}$ to $\emptyset$
2: $\mathcal{P}_0 \leftarrow SymbolicMetricInitialization(P_i)$
3: **for** $i = 1 \cdots \mathcal{N}$ **do**
4:      Clear promising candidates $\mathcal{C}$ to $\emptyset$
5:      Randomly select $r \times |\mathcal{P}|$ subnets $\hat{P}_i \in \mathcal{P}$ to get $\mathcal{C}$
6:      Candidates $\{\mathcal{S}_i\}_k \leftarrow GetTopk(\mathcal{C}, k)$
7:      Parent $\mathcal{S}_i^1, \mathcal{S}_i^2 \leftarrow RandomSample(\{\mathcal{S}_i\}_k)$
8:      $\mathcal{S}_i^c \leftarrow CROSSOVER(\mathcal{S}_i^1, \mathcal{S}_i^2)$
9:      $\mathcal{S}_i^m \leftarrow MUTATE(\mathcal{S}_i^c)$ with probability p
10:     $\mathcal{S}_i^s \leftarrow OOS(\mathcal{S}_i^m)$
11:     **if** $IsEquivalent(\mathcal{S}_i^s, \mathcal{S}_i^1)$ OR $IsEquivalent(\mathcal{S}_i^s, \mathcal{S}_i^2)$ **then**
12:        $\mathcal{S}_i^s \leftarrow RandomSample(\mathcal{S}, d)$
13:     **end if**
14:     Append offspring $\mathcal{S}_i^s$ to population with its PPL
15:     Remove the symbolic tree with the highest PPL
16: **end for**
17: **Procedure** $OOS(\mathcal{S}_i^m)$
18: **for** each node in $\mathcal{S}_i^m$ **do**
19:      Determine if the current node is opposing operations compared to $\mathcal{S}_i^m$.
20:      **if** opposing operation is found **then**
21:        Remove the opposing operation pair.
22:      **end if**
23: **end for**
24: **end Procedure**

Figure 2. Comparison between Evolution Search and Random Search Processes. Notably, the individual perplexity is lower in the evolution search method, leading to significant improvements in the search efficiency and overall stability.
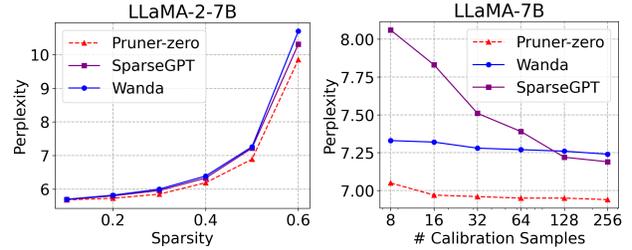
Figure 3. Left: Perplexity under Various Sparsity Ratio; Right: Perplexity with Different Calibration Samples.

tially with the increase in the number of primitive operations. Genetic Programming (GP), being the predominant methodology for addressing SR problems, guides our approach. Accordingly, we leverage GP to iteratively search for the optimal symbolic pruning metric.

An overview of our framework is depicted in Figure 1 and elaborated upon in Algorithm 1. Initially, we populate $\mathcal{P}$ with symbolic trees of varying depths. In each iteration, two parent symbolic pruning metrics are selected via tournament selection from the top-k candidates $(\{\mathcal{S}_i\}_k)$ with *GetTopk*$(\mathcal{C}, k)$ in line 6 of Algorithm 1. These parents then undergo subtree crossover and node mutation at a probability p, generating offspring symbolic pruning metrics. To minimize population redundancy, we implement the Opposing Operation Simplification (OOS) strategy, ensuring uniqueness among the metrics within the population.

Symbolic Metric Initialization. Each symbolic tree $\mathcal{S}_i$ represents an analytical equation aimed at quantifying the saliency of weights. To ensure balanced complexity through-

out the population, we initialize symbolic metrics with depths varying from three to five. Given the necessity for shape conformity between the symbolic tree and the weights, we impose specific restrictions on the activation because the shape of activation X differs from that of weight W. Typically, the norm of activation $||X||_2$ is utilized for computation to address this discrepancy. To resolve the shape-matching issue, we stipulate that if $||X||_2$ serves as the child node of a unary operation, it is excluded from the search space, and a new node is generated in its stead. This process is illustrated in the *SymbolicMetricInitialization* function in line 2 of Algorithm 1.

Crossover and Mutation. As shown in Figure 1, the process of subtree crossover involves the selection of two subtrees from the parent symbolic trees at random, followed by their subsequent interchange, as shown in the lines 8 and 9 of Algorithm 1. This crossover technique facilitates the combination of genetic symbols from parents, potentially leading to offspring with enhanced performance metrics. Following the crossover, node mutation is conducted with a probability of p. This mutation process is constrained to altering nodes within the same operation type, ensuring that unary operations can only mutate into other unary operations, and similarly for binary operations. This constraint maintains the structural integrity of the symbolic trees.

Opposing Operation Simplification (OOS). During our exploration of the search space, we observe that equivalent symbolic pruning metrics frequently encompass pairs of antagonistic operations, such as *exp* and *log*, or *sub* and *neg*,

Table 2. Perplexity of pruned LLaMA and LLaMA-2 models on WikiText2. Our Pruner-Zero outperforms SparseGPT and Wanda, achieving lower Perplexity without weight updates.

Method	Weight Update	Sparsity	LLaMA				LLaMA-2		
			7B	13B	30B	65B	7B	13B	70B
Dense	-	0%	5.68	5.09	4.77	3.56	5.12	4.57	3.12
Magnitude (Han et al., 2016b)	✗	50%	17.29	20.21	7.54	5.90	14.89	6.37	4.98
SparseGPT (Frantar & Alistarh, 2023)	✓	50%	7.22	6.21	5.31	4.57	6.51	5.63	3.98
Wanda (Sun et al., 2024)	✗	50%	7.26	6.15	5.24	4.57	6.42	5.56	3.98
Pruner-Zero	✗	50%	**6.95**	**5.94**	**5.01**	**4.33**	**6.26**	**5.36**	**3.82**
Magnitude (Han et al., 2016b)	✗	4:8	16.84	13.84	7.62	6.36	16.48	6.76	5.54
SparseGPT (Frantar & Alistarh, 2023)	✓	4:8	8.61	7.40	6.17	5.38	8.12	6.60	4.59
Wanda (Sun et al., 2024)	✗	4:8	8.57	7.40	5.97	5.30	7.97	6.55	4.47
Pruner-Zero	✗	4:8	**8.12**	**6.81**	**5.65**	**4.92**	**7.67**	**6.10**	**4.31**
Magnitude (Han et al., 2016b)	✗	2:4	42.13	18.37	9.10	7.11	54.59	8.33	6.33
SparseGPT (Frantar & Alistarh, 2023)	✓	2:4	11.00	9.11	7.16	6.28	**10.17**	8.32	5.40
Wanda (Sun et al., 2024)	✗	2:4	11.53	9.58	6.90	6.25	11.02	8.27	5.16
Pruner-Zero	✗	2:4	**10.61**	**8.11**	**6.51**	**5.67**	10.52	**7.41**	**4.81**

leading to an unnecessary increase in complexity. This redundancy detracts from the efficiency of the search process and obfuscates the interpretability of the expression trees. The OOS methodology is designed to refine the search space by eliminating these pairs of opposing operations, thereby streamlining the genetic representation and facilitating the identification of more concise and potent symbolic expressions. To implement this strategy, we meticulously catalog all opposing operations as detailed in Appendix E.1. Upon detecting operations within the same symbolic pruning metric that are opposed, we proceed to eliminate them as in line 17 of Algorithm 1.

Perplexity as Fitness. Upon generating offspring metrics, we undertake a post-training pruning evaluation using the LLaMA-2-7B model on the WikiText2 dataset (Merity et al., 2017) to determine their perplexity, which serves as the measure of fitness. In this context, a lower perplexity score indicates superior fitness, reflecting a more effective pruning metric in terms of model performance.

Searched Symbolic Pruning Metric. Here is the formula of the searched Symbolic Pruning Metric:

$$\text{Pruner-Zero} = ||W| \times |W|| \times \sigma(|G|) \quad (2)$$

where (σ) represents the min-max scaling function. This scaling function is pivotal in normalizing the weights ($|W|$) and gradients ($|G|$). By applying the min-max scaling, the weights and gradients are transformed to a common scale without distorting differences in the ranges of values. The squaring of the normalized weights and the subsequent multiplication with the normalized gradients, all under a square root, suggests a geometric mean approach. This form of averaging is beneficial for pruning metrics as it tends to

dampen the effect of extreme values.

4. Experiments

4.1. Models and Implementation Details

Implementation Details.

In this section, we primarily assess the effectiveness of our Pruner-Zero on two of the most extensively used LLM families: LLaMA 7B/13B/30B/65B (Touvron et al., 2023a) and LLaMA-2-7B/13B/70B (Touvron et al., 2023b). To further explore the generalizability of Pruner-Zero, we apply the developed symbolic pruning metric to earlier LLM families, such as OPT (Zhang et al., 2022b) and Tiny-LLaMA (Zhang et al., 2024a), as detailed in Appendix E.4. The performance of the pruned models is evaluated in terms of language modeling and zero-shot tasks. For language modeling, we follow the established protocols in LLM compression research (Sun et al., 2024; Frantar & Alistarh, 2023) to assess the perplexity on the WikiText2 (Merity et al., 2017) validation set. For zero-shot tasks, we utilize seven tasks from the EleutherAI LM Harness (Gao et al., 2021).

Counterparts. In comparing Pruner-Zero with established Post-training Pruning methods, magnitude pruning (Han et al., 2015) serves as a baseline, removing weights based on the absolute value of magnitude. SparseGPT (Frantar & Alistarh, 2023) introduces second-order pruning for LLMs, incorporating layer-wise reconstruction, while Wanda (Sun et al., 2024) simplifies this by using a diagonal approximation, avoiding matrix inverse calculations. Our approach avoids the computationally expensive Hessian matrix and aligns with magnitude pruning and Wanda in forgoing retraining or weight updates during pruning. Unlike

Table 3. Mean zero-shot accuracies (%) of pruned LLaMA and LLaMA-2 models on the BoolQ, RTE, HellaSwag, WinoGrande, ARC, and OBQA datasets. Wanda performs competitively against prior best method SparseGPT, without introducing any weight update.

Method	Weight Update	Sparsity	LLaMA				LLaMA-2		
			7B	13B	30B	65B	7B	13B	70B
Dense	-	0%	59.99	62.59	65.38	66.97	59.71	63.03	67.08
Magnitude (Han et al., 2016b)	✗	50%	46.94	47.61	53.83	62.74	51.14	52.85	60.93
SparseGPT (Frantar & Alistarh, 2023)	✓	50%	54.94	58.61	63.09	66.30	56.24	60.72	67.28
Wanda (Sun et al., 2024)	✗	50%	54.21	59.33	63.60	66.67	56.24	60.83	67.03
Pruner-Zero	✗	50%	**59.56**	**62.67**	**67.49**	**69.81**	**58.87**	**64.83**	**71.10**
Magnitude (Han et al., 2016b)	✗	4:8	46.03	50.53	53.53	62.17	50.64	52.81	60.28
SparseGPT (Frantar & Alistarh, 2023)	✓	4:8	52.80	55.99	60.79	64.87	53.80	59.15	65.84
Wanda (Sun et al., 2024)	✗	4:8	52.76	56.09	61.00	64.97	52.49	58.75	66.06
Pruner-Zero	✗	4:8	**56.24**	**59.03**	**64.04**	**68.04**	**55.82**	**61.97**	**69.94**
Magnitude (Han et al., 2016b)	✗	2:4	44.73	48.00	53.16	61.28	45.58	49.89	59.95
SparseGPT (Frantar & Alistarh, 2023)	✓	2:4	50.60	53.22	58.91	62.57	50.94	54.86	63.89
Wanda (Sun et al., 2024)	✗	2:4	48.53	52.30	59.21	62.84	48.75	55.03	64.14
Pruner-Zero	✗	2:4	**52.06**	**56.78**	**62.00**	**65.42**	**52.02**	**58.38**	**67.69**

SparseGPT and Wanda, which require calibration data, we use WikiText2 to estimate input statistics like Gradients (G) and Activations (X).

Sparsity. We adopt a uniform sparsity across all linear layers in unstructured pruning settings, maintaining consistency with the approach used by Wanda. Our focus for all pruning methods is exclusively on the linear layers of the model. To ensure a fair comparison with SparseGPT and Wanda, we assess three types of sparsity: unstructured, structured 4:8, and structured 2:4. These sparsity types are chosen to reflect different levels of granularity and constraints in pruning. All our implementations are based on Wanda's framework, providing a robust and consistent basis for evaluating the effectiveness of the various pruning strategies.

Evolution Settings. The evolutionary search starts with an initial population of 50 and 300 iterations. The depth of symbolic trees ranges from 3 to 5. Tournament selection utilizes a top-K parameter of 10, selecting two parent symbolic pruning metrics from the 10 best-performing candidates. The mutation probability is set to 0.5. This search, focused on identifying an optimal symbolic pruning metric, is executed using the LLaMA-2-7B model. Perplexity is evaluated under unstructured pruning with 50% sparsity. The searched metric, detailed in Table 1, is applied to various model sizes. The genetic programming tasks are executed on two NVIDIA 4090 GPUs, while the generalization experiments involving zero-shot tasks and language modeling on the LLaMA-2-70B are conducted using 8 A100 GPUs.

4.2. Language Modeling

As shown in Table 2, this study reports on the perplexity of pruned LLaMA and LLaMA-2 models under 50% unstruc-

tured and 2:4, 4:8 structured pruning scenarios. We annotate the necessity of weight updates and the corresponding sparsity levels for both unstructured and structured pruning methods. Remarkably, Pruner-Zero outperforms all established pruning techniques without requiring any weight updates. Notably, it surpasses the magnitude baseline by a significant margin. A trend observed is that Pruner-Zero demonstrates a lesser performance drop in larger ones. For instance, in unstructured pruning, the LLaMA-7B model shows a decrease of 1.27 in perplexity, whereas the LLaMA-30B model only exhibits a 0.24 drop. Similarly, the LLaMA-2-7B model experiences a 1.14 drop in perplexity, while the drop for the LLaMA-2-70B model is only 0.7. This indicates that Pruner-Zero is particularly advantageous for larger models, such as LLaMA-30B and LLaMA-2-70B. In comparison, Pruner-Zero consistently outperforms Wanda and mostly surpasses SparseGPT, except for the LLaMA-2 model in the 2:4 structured pruning setting. For robustness analysis under different seeds, refer to Table 25. For experiments on previous pruning methods applied to BERT (Devlin et al., 2019), see Appendix E.3. For comparison with GBLM-Pruner (Das et al., 2023), see Table 13.

4.3. Zero-shot Tasks

To assess the generalizability of Pruner-Zero, we evaluate its performance on seven common-sense tasks from the Eleuther AI lm-evaluation-harness benchmark (Gao et al., 2021) in a zero-shot setting. The selected tasks encompass BoolQ (Clark et al., 2019), RTE (Wang et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), ARC (Clark et al., 2018), and OBQA (Mihaylov et al., 2018). Although zero-shot evaluation on individual tasks can exhibit variability, Table 3 presents the average per-

Table 4. Perplexity of the pruned OPT family models with various sizes on WikiText2. † denotes updating weights using 128 samples. §
denotes we utilize 128 samples using iterative update.

			OPT					
Method	Weight Update	Sparsity	125m	350m	1.3B	2.7B	6.7B	13B
Dense	-	0%	27.66	22.00	14.62	12.47	10.86	10.13
Magnitude	✗	50%	7e3	6e3	1e4	9e3	9e4	2e4
Wanda	✗	50%	38.96	35.92	19.12	14.28	11.94	11.42
Pruner-Zero	✗	50%	**37.69**	**35.91**	**18.19**	**13.85**	**11.86**	**11.32**
SparseGPT	✓	50%	37.07	34.76	**17.44**	13.48	11.57	11.19
Pruner-Zero†	✓	50%	**34.04**	30.47	18.57	13.31	**11.52**	11.54
Pruner-Zero§	✓	50%	35.51	**29.75**	18.60	**13.24**	**11.52**	**10.86**

Table 5. Perplexity and Searched SPM w.r.t. OOS

Method	Perplexity	Searched SPM Expression
w OOS	6.7079	$\|\|W\|\| \times \|\|W\|\| \times \sigma(G)$
w/o OOS	6.8395	$\frac{\|\|W\|\|+G}{\sqrt{\|\|W\|\|}} + \sigma(\zeta(\|\|W\|\|))$

Table 6. Fine-tuning can mitigate the perplexity gap to dense LLM.

Evaluation	Dense	Fine-tuning	50%	4:8	2:4
LLaMa-7B	5.68	✗	6.95	8.12	10.61
		LoRA	6.74	7.40	8.24
LLaMa-2-7B	5.12	✗	6.26	7.67	10.52
		LoRA	6.41	7.01	7.73

Table 7. In-Context Learning Accuracy on the GSM8K Dataset

Dataset	GSM8K
Dense	0.287
Magnitude	0.0607
SparseGPT (Frantar & Alistarh, 2023)	0.1152
Wanda (Sun et al., 2024)	0.1312
Pruner-Zero	0.1403

formance of Pruner-Zero across all seven tasks to enhance the interpretability of the model's overall performance.

The results demonstrate that our Pruner-Zero significantly outperforms the magnitude baseline and competes effectively with previous top-performing approaches, such as SparseGPT and Wanda. Additionally, we observe that with increasing model size, the accuracy gap in zero-shot tasks decreases. The larger models like LLaMA-2-70B even surpass the dense model baselines. Detailed performance metrics for each task are provided in Appendix E.2.

4.4. Evaluation of In-Context Learning

In-context learning, a pivotal capability of large language models (LLMs), is crucial for tasks requiring adaptability and reasoning without explicit retraining. To assess this capability within various models, we focused our evaluation on the GSM8K dataset, comprised of diverse and complex grade school math problems that effectively challenge the models' reasoning abilities. Utilizing the LightLLM framework (ModelTC, 2023), a Python-based tool optimized for efficient and scalable inference with LLMs, we tested the LLaMA2 13B model's in-context learning performance using a fixed set of eight demonstrations aligned with the Chain-of-Thought (Wei et al., 2022) approach. The

comparative analysis presented in Table 7 indicates that Pruner-Zero outperforms other pruning approaches such as SparseGPT (Frantar & Alistarh, 2023) and Wanda (Sun et al., 2024) in the in-context learning tasks, despite the general trend of decreased performance post-pruning. This observation, consistent with findings from recent studies (Li et al., 2024c) documenting a reduction in in-context learning capabilities of LLMs following quantization, highlights a significant challenge in the field of model pruning and optimization: preserving the nuanced capabilities of LLMs while reducing their computational overhead. Although Pruner-Zero shows a marked improvement over other methods, the noticeable decline in the performance of the unpruned (Dense) model underscores the delicate balance between model size and functionality, particularly in tasks requiring high cognitive functions such as reasoning and comprehension.

4.5. Ablation Study

Evolution Search vs. Random Search. To evaluate the efficacy of Evolution Search, ablation studies were conducted, the results of which are presented in Figure 2. It was observed that Evolution Search achieves convergence in fewer than 100 iterations, in contrast to Random Search, which requires approximately 300 iterations to converge.

Robustness across Different Sparsity Ratios. The left side of Figure 3 presents the perplexity results for LLaMA-2-7B on WikiText2 across varying sparsity ratios, ranging

from 0.1 to 0.6. These results demonstrate that Pruner-Zero consistently outperforms SparseGPT and Wanda across all tested sparsity levels. This confirms the robustness and effectiveness of our symbolic pruning metric under diverse sparsity ratios, ensuring reliable performance even as the level of pruning increases.

Robustness to Calibration Sample Size. The right side of Figure 2 illustrates the impact of varying calibration sample sizes, ranging from 8 to 256 samples. A distinct trend emerges as the calibration sample size changes: all pruning methods demonstrate increased robustness and lower perplexity. Our Pruner-Zero demonstrates robustness to calibration samples; specifically, when the sample size exceeds 16, the perplexity value is below 7. Finally, we chose 128 as the calibration size to ensure alignment with Wanda.

Robustness to OPT family. Table 4 showcases the pruning performance on OPT models ranging from 125 million to 13 billion parameters. We find that Pruner-Zero outperforms both Magnitude and Wanda in post-training pruning without weight updates across various model sizes. Our Pruner-Zero also outperforms SparseGPT with weight updates following Wanda (Sun et al., 2024). A sequential update (†) means that at each layer, the full pruned mask is first computed and a weight update is performed on the remaining weights. An iterative update (§) means that the pruning and weight update steps proceed iteratively within each layer.

Effectiveness of OSS. Opposing Operation Simplification (OOS) strategy plays a crucial role in reducing redundancy among the candidate symbolic pruning metrics. Without OOS, the population may contain a large number of mathematically equivalent metrics, which is undesirable. The OOS strategy helps maintain diversity in the candidate solution space by eliminating such redundancies, aligning the search process with our expectations, and improving its efficiency. To quantize the influence of OOS, we conducted experiments under different depths of the symbolic tree using Effective probability. Effective probability calculates the frequency of when OOS works. The effective probability is computed based on 1,000 randomly generalized trees. And we conduct four trials to get the final result. From Table 12, we can find that the effective probability is very high denoting the redundancy of the search space. Besides, we present the searched SPM and their corresponding perplexity w.r.t OOS strategy in Table 5. The experiments were conducted for 300 iterations on the LLaMA-2-7B model with 50% unstructured pruning settings. The experiments without the OOS strategy showed deteriorated performance, likely due to redundancy in the small population size of 50, where high homogeneity in metrics prevented further performance increase.

4.6. Analysis

Correlation Analysis. During the evolutionary search, we gathered various symbolic pruning metrics and their corresponding perplexities, identifying those with perplexities below 7 as potential metrics (detailed in Table 11, Appendix C.3). The analysis of operation frequencies in these metrics, as shown in Figure 4, revealed several key insights: (1) Multiplication operations exhibit a strong correlation with perplexity, highlighting their importance. (2) Opposite operations (*sqrt* and *sqr*) show a high correlation, suggesting counteractive effects. (3) The min-max scaling (*mms*) operation is robust, displaying a low correlation with other operations, underscoring its significance. These observations support our metric development discussed in Section 3.2.

LoRA Fine-tuning. We further explore the potential of fine-tuning to mitigate the performance reduction observed in pruned Large Language Models (LLMs). Specifically, we employ LoRA (Hu et al., 2022) ($r = 8$) for fine-tuning on the C4 training dataset (Raffel et al., 2019) using 1 GPU and 12 hours, targeting the auto-regressive loss. Experiments were conducted on both the LLaMA and LLaMA-2-7B models, encompassing 50% unstructured as well as 2:4 and 4:8 structured pruning scenarios. The resulting perplexity on WikiText2 is detailed in Table 6. In most cases, fine-tuning with LoRA can restore the performance of pruned LLMs, especially for hard cases like 2:4 structured pruning.

5. Conclusion

In this paper, we present a Pruner-Zero framework for symbolic pruning metric discovery for Large Language Models (LLMs) by formulating it as a symbolic regression problem. We leverage genetic programming to efficiently search for superior symbolic pruning metrics that have lower perplexity after pruning. During the search, we find the opposing operation that affects the quality of the population and propose the Opposing Operation Simplification (OOS) strategy to enhance search efficiency. Our comprehensive experimental evaluation, conducted on the LLaMA and LLaMA-2 for both language modeling and zero-shot tasks, reveals that our Pruner-Zero framework surpasses current state-of-the-art (SOTA) methods, including Wanda and SparseGPT, in terms of both structured and unstructured pruning

Impact Statement

Pruner-Zero significantly refines the post-training pruning process of large language models, employing genetic programming to evolve symbolic pruning metrics. This method effectively identifies and eliminates non-critical parameters, thereby preserving the model's performance. The focus of this research is on technological innovation, and it does not extend to the analysis of ethical considerations or societal impacts.

References

Abdin, M., Jacobs, S. A., and etc., A. A. A. Phi-3 technical report: A highly capable language model locally on your phone. *ArXiv*, abs/2404.14219, 2024.

Akhauri, Y., Munoz, J. P., Jain, N., and Iyer, R. EZNAS: Evolving zero-cost proxies for neural architecture scoring. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *NeurIPS*, 2022.

Ashkboos, S., Croci, M. L., do Nascimento, M. G., Hoefler, T., and Hensman, J. SliceGPT: Compress large language models by deleting rows and columns. In *ICLR*, 2024.

Chen, T., Frankle, J., Chang, S., Liu, S., Zhang, Y., Wang, Z., and Carbin, M. The lottery ticket hypothesis for pre-trained bert networks. In *NeurIPS*, 2020.

Chen, T., Liang, L., DING, T., Zhu, Z., and Zharkov, I. OTOv2: Automatic, generic, user-friendly. In *ICLR*, 2023.

Chijiwa, D., Yamaguchi, S. y., Ida, Y., Umakoshi, K., and INOUE, T. Pruning randomly initialized neural networks with iterative randomization. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *NeurIPS*, volume 34, pp. 4503–4513. Curran Associates, Inc., 2021.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. Boolq: Exploring the surprising difficulty of natural yes/no questions. *ArXiv*, abs/1905.10044, 2019.

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

Das, R. J., Ma, L., and Shen, Z. Beyond size: How gradients shape pruning decisions in large language models. *ArXiv*, abs/2311.04902, 2023.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Diao, E., Wang, G., Zhang, J., Yang, Y., Ding, J., and Tarokh, V. Pruning deep neural networks from a sparsity perspective. In *ICLR*, 2023.

Dong, P., Niu, X., Li, L., Xie, L., Zou, W., Ye, T., Wei, Z., and Pan, H. Prior-guided one-shot neural architecture search. *arXiv preprint arXiv:2206.13329*, 2022.

Dong, P., Li, L., and Wei, Z. Diswot: Student architecture search for distillation without training. In *CVPR*, 2023a.

Dong, P., Li, L., Wei, Z., Niu, X., Tian, Z., and Pan, H. Emq: Evolving training-free proxies for automated mixed precision quantization. In *ICCV*, pp. 17076–17086, 2023b.

Dong, P., Li, L., Pan, X., Wei, Z., Liu, X., Wang, Q., and Chu, X. Parzc: Parametric zero-cost proxies for efficient nas. *arXiv preprint arXiv:2402.02105*, 2024.

Fang, G., Ma, X., Song, M., Mi, M. B., and Wang, X. Depgraph: Towards any structural pruning. *CVPR*, pp. 16091–16101, 2023.

Frantar, E. and Alistarh, D. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *ICML*, 2023.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. Gptq: Accurate post-training quantization for generative pre-trained transformers. In *ICLR*, 2023.

Gao, L., Tow, J., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., McDonell, K., Muennighoff, N., Phang, J., Reynolds, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, September 2021.

Han, S., Pool, J., Tran, J., and Dally, W. J. Learning both weights and connections for efficient neural network. In *NeurIPS*, 2015.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *ICLR*, 2016a.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *ICLR*, 2016b.

He, X., Yao, J., Wang, Y., Tang, Z., Cheung, K. C., See, S., Han, B., and Chu, X. Nas-lid: Efficient neural architecture search with local intrinsic dimension. In *AAAI*, 2022.

Hoang, D. N., Liu, S., Marculescu, R., and Wang, Z. revisiting pruning at initialization through the lens of ramanujan graph. In *ICLR*, 2023.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Hu, Y., Wang, X., Li, L., and Gu, Q. Improving one-shot nas with shrinking-and-expanding supernet. *Pattern Recognition*, 2021.

Koza, J. R. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4:87–112, 1994.

Kwon, W., Kim, S., Mahoney, M. W., Hassoun, J., Keutzer, K., and Gholami, A. A fast post-training pruning framework for transformers. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *NeurIPS*, 2022.

Lee, N., Ajanthan, T., and Torr, P. H. Snip: Single-shot network pruning based on connection sensitivity. In *ICLR*, 2019.

Li, L. Self-regulated feature learning via teacher-free feature distillation. In *ECCV*, 2022.

Li, L. and Jin, Z. Shadow knowledge distillation: Bridging offline and online knowledge transfer. In *NeuIPS*, 2022.

Li, L., Dong, P., Wei, Z., and Yang, Y. Automated knowledge distillation via monte carlo tree search. In *ICCV*, 2023.

Li, L., Dong, P., Li, A., Wei, Z., and Yang, Y. Kd-zero: Evolving knowledge distiller for any teacher-student pairs. *NeuIPS*, 2024a.

Li, S., Han, X., and Bai, J. Nuteprune: Efficient progressive pruning with numerous teachers for large language models. *ArXiv*, abs/2402.09773, 2024b.

Li, S., Ning, X., Wang, L., Liu, T., Shi, X., Yan, S., Dai, G., Yang, H., and Wang, Y. Evaluating quantized large language models. *ICML*, abs/2402.18158, 2024c.

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., and Han, S. Awq: Activation-aware weight quantization for llm compression and acceleration. In *MLSys*, 2024.

Liu, S.-y., Liu, Z., Huang, X., Dong, P., and Cheng, K.-T. LLM-FP4: 4-bit floating-point quantized transformers. In *EMNLP*, pp. 592–605, Singapore, December 2023a. Association for Computational Linguistics. URL https://aclanthology.org/2023.emnlp-main.39.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Rethinking the value of network pruning. In *ICLR*, 2019.

Liu, Z., Oguz, B., Zhao, C., Chang, E., Stock, P., Mehdad, Y., Shi, Y., Krishnamoorthi, R., and Chandra, V. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023b.

Louizos, C., Welling, M., and Kingma, D. P. Learning sparse neural networks through l_0 regularization. In *ICLR*, 2018.

Lu, L., Chen, Z., Lu, X., Rao, Y., Li, L., and Pang, S. Uniads: Universal architecture-distiller search for distillation gap. In *AAAI*, 2024.

Lu, M., Luo, X., Chen, T., Chen, W., Liu, D., and Wang, Z. Learning pruning-friendly networks via frank-wolfe: One-shot, any-sparsity, and no retraining. In *ICLR*, 2022.

Luccioni, A. S., Viguier, S., and Ligozat, A.-L. Estimating the carbon footprint of bloom, a 176b parameter language model. *Journal of Machine Learning Research*, 24(253): 1–15, 2023.

Ma, X., Fang, G., and Wang, X. Llm-pruner: On the structural pruning of large language models. In *NeurIPS*, 2023.

Martius, G. and Lampert, C. H. Extrapolation and learning equations. *ArXiv*, abs/1610.02995, 2016.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

ModelTC. Lightllm. https://github.com/ModelTC/lightllm, 2023. A Python-based LLM inference and serving framework.

OpenAI. Gpt-4 technical report, 2023.

Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., and Dean, J. Efficient neural architecture search via parameter sharing. In *ICML*, 2018.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

Real, E., Liang, C., So, D., and Le, Q. Automl-zero: Evolving machine learning algorithms from scratch. In *ICML*, 2020.

Ren, S. and Zhu, K. Pruning pre-trained language models with principled importance and self-regularization. In *ACL*, pp. 8995–9008, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.573.

Sahoo, S. S., Lampert, C. H., and Martius, G. Learning equations for extrapolation and control. In *ICML*, 2018.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande. *Communications of the ACM*, 64:99–106, 2019.

Sanh, V., Wolf, T., and Rush, A. Movement pruning: Adaptive sparsity by fine-tuning. *NeurIPS*, 33:20378–20389, 2020.

Schwartz, R., Dodge, J., Smith, N. A., and Etzioni, O. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.

Sharma, P., Ash, J. T., and Misra, D. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. In *ICLR*, 2024.

Sreenivasan, K., yong Sohn, J., Yang, L., Grinde, M., Nagle, A., Wang, H., Xing, E., Lee, K., and Papailiopoulos, D. Rare gems: Finding lottery tickets at initialization. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *NeurIPS*, 2022.

Srinivas, S. and Babu, R. V. Data-free parameter pruning for deep neural networks. In *BMVC*, 2015.

Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. In *ICLR*, 2024.

Tanaka, H., Kunin, D., Yamins, D. L., and Ganguli, S. Pruning neural networks without any data by iteratively conserving synaptic flow. *NeurIPS*, 33:6377–6389, 2020.

Tang, Z., Wang, Y., Wang, Q., and Chu, X. The impact of gpu dvfs on the energy and performance of deep learning: An empirical study. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, e-Energy '19, pp. 315–325, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366717. doi: 10.1145/3307772.3328315.

Tang, Z., Shi, S., Chu, X., Wang, W., and Li, B. Communication-efficient distributed deep learning: A comprehensive survey. *arXiv preprint arXiv:2003.06307*, 2020.

Tang, Z., Wang, Y., He, X., Zhang, L., Pan, X., Wang, Q., Zeng, R., Zhao, K., Shi, S., He, B., et al. Fusionai: Decentralized training and deploying llms with massive consumer-level gpus. *arXiv preprint arXiv:2309.01172*, 2023.

Tang, Z., Zhang, Y., Shi, S., Tian, X., Liu, T., Han, B., and Chu, X. Fedimpro: Measuring and improving client update in federated learning. In *ICLR*, 2024.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

van der Ouderaa, T. F. A., Nagel, M., van Baalen, M., Asano, Y. M., and Blankevoort, T. The llm surgeon. In *ICLR*, 2024.

Virgolin, M. and Pissis, S. P. Symbolic regression is np-hard. *arXiv preprint arXiv:2207.01018*, 2022.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP@EMNLP*, 2018.

Wang, H., Qin, C., Zhang, Y., and Fu, Y. R. Neural pruning via growing regularization. *ICLR*, abs/2012.09243, 2020.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., brian ichter, Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Wei, Z., Pan, H., Li, L., Dong, P., Tian, Z., Niu, X., and Li, D. Tvt: Training-free vision transformer search on tiny datasets. *arXiv preprint arXiv:2311.14337*, 2023.

Wei, Z., Dong, P., Hui, Z., Li, A., Li, L., Lu, M., Pan, H., and Li, D. Auto-prox: Training-free vision transformer architecture search via automatic proxy discovery. In *AAAI*, 2024.

Werner, M., Junginger, A., Hennig, P., and Martius, G. Informed equation learning. *ArXiv*, abs/2105.06331, 2021.

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *ICML*, 2023.

Xiaolong, L., Lujun, L., Chao, L., and Yao, A. Norm: Knowledge distillation via n-to-one representation matching. In *ICLR*, 2023.

Xu, P., Shao, W., Chen, M., Tang, S., Zhang, K., Gao, P., An, F., Qiao, Y., and Luo, P. BESA: Pruning large language models with blockwise parameter-efficient sparsity allocation. In *ICLR*, 2024.

Yang, Z., Cui, Y., Yao, X., and Wang, S. Gradient-based intra-attention pruning on pre-trained language models. In *ACL*, 2022.

Yao, Z., Yazdani Aminabadi, R., Zhang, M., Wu, X., Li, C., and He, Y. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *NeurIPS*, volume 35, pp. 27168–27183. Curran Associates, Inc., 2022.

Yin, L., Wu, Y., Zhang, Z., Hsieh, C.-Y., Wang, Y., Jia, Y., Pechenizkiy, M., Liang, Y., Wang, Z., and Liu, S. Outlier weighed layerwise sparsity (OWL): A missing secret sauce for pruning LLMs to high sparsity. In *ICML*, 2024.

Yvinec, E., Dapogny, A., Cord, M., and Bailly, K. Red++ : Data-free pruning of deep neural networks via input splitting and output merging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45:3664–3676, 2021.

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

Zhang, P., Zeng, G., Wang, T., and Lu, W. Tinyllama: An open-source small language model. *ArXiv*, abs/2401.02385, 2024a.

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

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M. T., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T., and Zettlemoyer, L. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068, 2022b.

Zhang, Y., Bai, H., Lin, H., Zhao, J., Hou, L., and Cannistraci, C. V. Plug-and-play: An efficient post-training pruning method for large language models. In *ICLR*, 2024b.

Zhu, C., Li, L., Wu, Y., and Sun, Z. Saswot: Real-time semantic segmentation architecture search without training. In *AAAI*, 2024.

Zimmer, M., Andoni, M., Spiegel, C., and Pokutta, S. Perp: Rethinking the prune-retrain paradigm in the era of llms. *ArXiv*, abs/2312.15230, 2023.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018.

A. Related Work

The development of Large Language Models (LLMs) has witnessed a surge in model and dataset sizes, necessitating distributed training across numerous devices (Tang et al., 2020; 2023; 2024). This distributed approach, while effective, demands substantial computational and storage resources, with LLMs incurring higher energy costs compared to their smaller counterparts (Luccioni et al., 2023; Schwartz et al., 2020; Tang et al., 2019). Consequently, energy-efficient LLM training and inference are crucial for green computing, with LLM pruning emerging as a key technique for achieving this goal. Post-training pruning, in particular, has gained prominence due to its minimal resource requirements, making it a cost-effective approach for democratizing access to LLMs (Lu et al., 2022; Frantar & Alistarh, 2023; Sun et al., 2024). This method's efficiency and accessibility contribute significantly to the broader impact and applicability of LLMs.

Network Pruning   Network pruning is an effective technique for reducing model complexity while preserving performance, although it often requires extensive retraining. However, traditional pruning methods (Hoang et al., 2023; Sreenivasan et al., 2022; Liu et al., 2019; Chen et al., 2023; Chijiwa et al., 2021) become impractical when dealing with the substantial parameter sizes and vast datasets of Large Language Models (LLMs). Deep Compression (Han et al., 2016a) popularized magnitude-based pruning for deep neural networks, which removes the weights with the smallest absolute values, assuming that they have the least impact on the network's output. Network pruning techniques can be broadly categorized into two main approaches: unstructured pruning and structured pruning.

(1) Unstructured Pruning involves removing individual weights or connections based on certain criteria. SparseGPT (Frantar & Alistarh, 2023) is the first post-training quantization method that performs unstructured pruning using an approximated Hessian matrix. Wanda (Sun et al., 2024) further simplifies the Hessian matrix by using just the weight and l_2 norm of activation. GBLM-Pruner (Das et al., 2023) further introduces the gradient to boost the performance. Plug-and-play (Zhang et al., 2024b) The paper presents a plug-and-play post-training pruning method for large language models (LLMs) that introduces two innovative components: Relative Importance and Activations (RIA), a new pruning metric, and Channel Permutation, a technique to maximize the preservation of important weights under N:M sparsity constraints. The proposed method, named plug-and-play, outperforms existing pruning techniques and achieves practical speed-up on specific hardware without the need for additional fine-tuning or retraining. PERP (Zimmer et al., 2023) uses Low-rank adaptation to mitigate the expense of the retraining process in the original prune-retrain paradigm. NutePrune (Li et al., 2024b) combines structure pruning with progressive knowledge distillation by utilizing the unpruned model as a teacher and the pruned model as a student. OWL (Yin et al., 2024) proposed outliers metric to re-assign the sparsity of different layers. BESA (Xu et al., 2024) proposes to use parameter-efficient sparsity learning to learn the sparsity ratio in a differentiable manner. GRAIN (Yang et al., 2022) utilizes gradient information to prune intra-attention structures,

Table 8. Comparison of Pruner-Zero with its counterparts.

Description	AutoML-Zero (Real et al., 2020)	EZNAS (Akhauri et al., 2022)	Auto-Prox(Wei et al., 2024)	EMQ (Dong et al., 2023b)	Pruner-Zero
Task	Machine Learning Program Discovery	Zero-shot NAS	Zero-shot NAS	Mixed-precision Quantization	Symbolic Pruning Metric
Targets	-	CNN	ViT	CNN	LLMs
Params	-	0.3-1.5MB	2-25MB	13.4-44.6MB	7B-70B
Retrain	No	Yes	Yes	Yes	No
Input	Multiple inputs	One or two inputs	Two inputs	Two inputs	Unlimited inputs
Output	Machine Learning Task (accuracy)	One scalar	One scalar	One scalar	Matrix with the same shape as weight
Strategy	Evolutionary Algorithm	Distributed Evolutionary Algorithm in Python (DEAP)	Elitism-Preserve Strategy	Diversity Prompting Selection	Opposing Operation Simplification strategy
Objective	Find machine learning algorithms from scratch	Find optimal proxy that can measure the convolution-based architectures	Find the optimal proxy that can measure the vit-based architectures	Find the optimal metric that can better rank candidate bit-width configurations	Find the optimal symbolic pruning metric that can measure the importance of different weights

Table 9. Impact of Pruning Metrics on GBLM-Pruner Performance. This figure illustrates the performance variations under 50% sparsity ratio in GBLM-Pruner (Das et al., 2023) when utilizing weight, gradient, and activation-based pruning metrics. The observed sensitivity of the results to seemingly minor format changes highlights the challenges of manual metric selection and underscores the need for an automated pruning framework, which serves as the motivation for this work.

Metric	Perplexity	Metric	Perplexity
$|\mathbf{W}| \cdot ||G||_1$	7.17	$(|\mathbf{W}| \cdot ||X||_2)^2 + \lambda \cdot |\mathbf{W}| \cdot ||G||_1$	180490.19
$|\mathbf{W}| \cdot ||G||_2$	7.09	$(|\mathbf{W}| \cdot ||X||_2)^2 + \lambda \cdot |\mathbf{W}| \cdot ||G||_2$	91781.49
$|\mathbf{W}| \cdot ||X||_2 \cdot ||G||_1$	7.31	$(|\mathbf{W}| \cdot ||X||_2)^2 - \lambda \cdot |\mathbf{W}| \cdot ||G||_1$	248646.28
$|\mathbf{W}| \cdot ||X||_2 \cdot ||G||_2$	7.31	$(|\mathbf{W}| \cdot ||X||_2)^2 - \lambda \cdot |\mathbf{W}| \cdot ||G||_2$	283620.75
$|\mathbf{W}| \cdot ||X||_2^2 + \lambda \cdot |\mathbf{W}| \cdot ||G||_1$	6.86	$(|\mathbf{W}| \cdot ||X||_2)^2 + \lambda \cdot |\mathbf{W}| \cdot ||G||_1$	6.90
$|\mathbf{W}| \cdot ||X||_2^2 + \lambda \cdot |\mathbf{W}| \cdot ||G||_2$	6.89	$(|\mathbf{W}| \cdot ||X||_2)^2 + \lambda \cdot |\mathbf{W}| \cdot ||G||_2$	6.88
$|\mathbf{W}| \cdot ||X||_2^2 - \lambda \cdot |\mathbf{W}| \cdot ||G||_1$	1180.67	$(|\mathbf{W}| \cdot ||X||_2)^2 - \lambda \cdot |\mathbf{W}| \cdot ||G||_1$	6.94
$|\mathbf{W}| \cdot ||X||_2^2 - \lambda \cdot |\mathbf{W}| \cdot ||G||_2$	7.10	$(|\mathbf{W}| \cdot ||X||_2)^2 - \lambda \cdot |\mathbf{W}| \cdot ||G||_2$	9377.00

incorporating knowledge distillation to enhance performance.

(2) Structured Pruning , also known as N:M structured pruning, is a specialized technique designed to enhance computational efficiency during the inference phase in deep learning models. This method is particularly tailored for compatibility with NVIDIA's Ampere architecture and its Sparse Tensor Cores. In the context of N:M structured pruning, for each block of M parameters, only N parameters are actively retained while the remainder is pruned, effectively set to zero or eliminated. Common configurations such as 2:4 and 4:8 structured pruning maintain a fixed sparsity ratio of 50%, indicating that only half of the parameters are preserved. Notably, the 4:8 structured pruning format offers less restrictive conditions than the 2:4 format, permitting a more adaptable parameter reduction. This flexibility in 4:8 structured pruning is demonstrated by superior performance metrics, as detailed in Tables 2 and 3, where 4:8 pruning consistently outperforms 2:4 configurations under identical experimental conditions. Furthermore, NVIDIA's A100 GPU represents a significant milestone as the first mainstream hardware to incorporate sparse capabilities directly into its architecture. This allows the Sparse Tensor Cores of the A100 to support a wide array of operations prevalent in modern neural networks, including linear transformations, convolutional layers, recurrent neural networks, and transformer architectures. This integration heralds a new era of efficiency, enabling more rapid and energy-efficient computation across diverse deep-learning applications. SliceGPT (Ashkboos et al., 2024), focuses on compressing LLMs by eliminating rows and columns from weight matrices, though it is mainly effective at lower sparsity ratios. LLM-Pruner (Ma et al., 2023) is inspired by DepGraph (Fang et al., 2023) to detect the dependency lies in the model, which also considers both first-order information and approximated Hessian information.

Neural Architecture Search (Zoph et al., 2018; Pham et al., 2018) aims to automate the design of neural network architectures. Traditional methods (Hu et al., 2021; Dong et al., 2022; Li et al., 2023; 2024a) for designing neural networks rely heavily on human expertise and extensive trial-and-error, which can be both time-consuming and resource-intensive. NAS methods (Dong et al., 2023b; Lu et al., 2024; Dong et al., 2024; He et al., 2022) seek to alleviate this burden by leveraging algorithmic strategies to discover optimal network architectures. Our method is partly related to Zero-Shot Neural Architecture Search (Zero-Shot NAS) (Wei et al., 2023; Dong et al., 2023a; Zhu et al., 2024), an emerging subfield within NAS. Zero-Shot NAS aims to predict the performance of neural network architectures without the need for extensive training and evaluation. The key to Zero-Shot NAS is the use of proxy metrics. Some Zero-Shot NAS methods propose to formulate the proxy as an automated discovery process (Akhauri et al., 2022; Wei et al., 2024). For example, EZNAS (Akhauri et al., 2022) and Auto-Prox (Wei et al., 2024) represent significant advancements in the field of zero-shot NAS by focusing on the automatic identification of zero-cost proxies tailored to CNN-based and ViT-based architectures, respectively. These methodologies align with the broader domain of program synthesis, which is dedicated to the automated generation of programs that meet specific user-defined constraints. An illustrative example within this domain is AutoML-Zero (Real et al., 2020), which innovatively evolves complete machine learning algorithms from a foundational level. Similarly, EMQ (Dong et al., 2023b) addresses the challenges of mixed-precision quantization. It aims to create a proxy capable of effectively ranking candidate bit-width configurations, thereby enhancing computational efficiency and model performance without extensive manual tuning.

Table 10. Operations vocabulary. Summary of Unary and Binary Operations in Computational Functions. This table includes operation identifiers, code examples, input and output types, and detailed descriptions for each operation.

Op ID	Code Example	Symbols	Input / Types	Output / Type	Description
U01	s_c=sqr(s_a)	$(\cdot)^2$	a / matrix	c / matrix	$s_c = s_a^2$
U02	s_c=neg(s_a)	$-(\cdot)$	a / matrix	c / matrix	$s_c = -s_a$
U03	s_c=abs(s_a)	$\lvert \cdot \rvert$	a / matrix	c / matrix	$s_c = \lvert s_a \rvert$
U04	s_c=log(s_a)	log	a / matrix	c / matrix	$s_c = \log(s_a)$
U05	s_c=exp(s_a)	$e^{(\cdot)}$	a / matrix	c / matrix	$s_c = e^{s_a}$
U06	s_c=sqrt(s_a)	$\sqrt{\cdot}$	a / matrix	c / matrix	$s_c = \sqrt{s_a}$
U07	s_c=tanh(s_a)	tanh	a / matrix	c / matrix	$s_c = \tanh(s_a)$
U08	s_c=pow(s_a, s_b)	$(\cdot)^{(\cdot)}$	a,b / matrixs	c / matrix	$s_c = s_a^{s_b}$
U09	s_c=skp(s_a)	$\emptyset$	a / matrix	c / matrix	$s_c = s_a$ (skip)
U10	s_c=mms(s_a)	$\sigma(\cdot)$	a / matrix	c / matrix	$s_c = $ min-max scale(s_a)
U11	s_c=zsn(s_a)	$\zeta(\cdot)$	a / matrix	c / matrix	$s_c = $ z-score scale(s_a)
U12	s_c=norm$_2$(s_a)	$\lvert\lvert \cdot \rvert\rvert_2$	a / matrix	c / matrix	$s_c = \lvert\lvert s_a \rvert\rvert_2$
U13	s_c=norm$_1$(s_a)	$\lvert\lvert \cdot \rvert\rvert_1$	a / matrix	c / matrix	$s_c = \lvert\lvert s_a \rvert\rvert_1$
B01	s_c=add(s_a, s_b)	$(\cdot) + (\cdot)$	a,b / matrixs	c / matrix	$s_c = s_a + s_b$
B02	s_c=sub(s_a, s_b)	$(\cdot) - (\cdot)$	a,b / matrixs	c / matrix	$s_c = s_a - s_b$
B03	s_c=mul(s_a, s_b)	$(\cdot) \times (\cdot)$	a,b / matrixs	c / matrix	$s_c = s_a \cdot s_b$
B04	s_c=div(s_a, s_b)	$(\cdot)/(\cdot)$	a,b / matrixs	c / matrix	$s_c = \frac{s_a}{s_b}$

Characteristics and Innovations:

- **Model Scale:** Pruner-Zero is designed to manage exceptionally large-scale language models (LLMs), with parameters ranging from 7 billion to 70 billion. This scale surpasses that of the other discussed methods, highlighting its capability to handle extensive computational loads and complex data structures.

- **Retraining Necessity:** Unlike other approaches that necessitate retraining to fulfill their objectives, Pruner-Zero operates efficiently without the requirement for further retraining. This feature significantly reduces the computational overhead and accelerates the optimization process.

- **Input/Output Paradigms:** Pruner-Zero employs a unique approach to inputs and outputs; it accepts an unlimited number of inputs and produces an output matrix that retains the same dimensions as the model weights. This distinct I/O schema facilitates direct interventions in model pruning and optimization.

- **Objective:** The primary goal of Pruner-Zero is to discover an optimal symbolic pruning metric that assesses the significance of different weights in large language models. This objective focuses on enhancing model efficiency and performance through targeted weight reduction.

These distinctions underscore the varied landscape of program synthesis, where different problems are defined and differentiated by their specific constraints, which are inherently linked to the tasks at hand. Such innovations contribute substantially to the field, pushing the boundaries of what is possible in automated machine learning and architecture search.

B. Motivation of Searching Pruning Metric

In this section, our exploration into the design of an effective pruning metric is inspired by the GBLM-Pruner (Das et al., 2023). The intricacies of pruning metrics are exemplified in Table 9, which is derived from their paper. A meticulous analysis of these metrics underscores the complexity and iterative nature of their design process. Notably, metrics that are nearly identical in their formulation can yield vastly divergent outcomes, as evidenced by the last two metrics in the second column of the table. These metrics differ only slightly in their normalization approach, yet the resulting perplexities are starkly contrasting (6.94 vs. 9377). Such findings illuminate the challenges faced in metric design and catalyze for our

development of Pruner-Zero, an automated solution aimed at refining the pruning metric formulation process. This initiative is geared towards enhancing the efficiency and efficacy of pruning within large language models.

C. Detailed Search Space

C.1. Search Space Composition

Our search space comprises three specific input types: activations (X), gradients (G), and weights (W). These inputs provide complementary perspectives on the significance and contribution of different model components to the overall performance.

Activation. Activation values from a model's layers offer rich, interpretable information about the role of individual neurons in processing input data. By considering activation magnitudes or patterns across a calibration dataset, we can identify and prune less important neurons or connections. For instance, consistently low activations might suggest redundancy, while highly correlated activations could point to opportunities for merging similar neurons.

Gradient. Gradients reflect the sensitivity of a model's output with respect to its parameters (weights). Larger gradient magnitudes indicate that small changes in the corresponding weights would significantly impact the model's predictions. Therefore, gradients can guide pruning by highlighting weights crucial for the model's performance. Pruning strategies might target weights with consistently small gradients, as their impact on the output is likely less significant.

Weight. Weights represent the learned parameters of a neural network and are fundamental to its ability to make predictions. The importance of weights can be assessed by evaluating their magnitudes and the roles they play within the network. Weights with larger magnitudes typically have a greater influence on the network's output, while smaller weights might contribute less significantly. By analyzing the distribution and significance of weights, we can identify which ones are essential for maintaining model performance. Pruning methods often focus on removing weights that have minimal impact, thereby simplifying the network and reducing its complexity without substantially degrading its accuracy.

Hessian Matrix, a fundamental concept in multivariable calculus, is instrumental in understanding the curvature of scalar fields and optimizing complex functions. While the utility of the Hessian matrix in various mathematical and engineering disciplines is well-documented, its specific application in the pruning of large language models (LLMs) merits particular attention due to its potential for enhancing model efficiency. The Hessian is a square matrix consisting of second-order partial derivatives of a scalar-valued function. For a function $f(x_1, x_2, \ldots, x_n)$ of n variables, the Hessian matrix H is given by:

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

This matrix is crucial for determining the local curvature of a function, which in turn helps in identifying maxima, minima, and saddle points — essential for optimization algorithms.

In the context of LLM pruning, the Hessian matrix offers a theoretically sound approach to evaluating the importance of weights within a neural network. Pruning methods aim to reduce the computational complexity of LLMs by removing weights that contribute minimally to the model's output. By assessing the impact of weight removal through the lens of the Hessian's curvature effects, one can theoretically predict and mitigate potential losses in model performance more effectively.

However, the practical application of the Hessian matrix in LLM pruning is not without challenges. The computation of the Hessian matrix is notably resource-intensive, particularly due to the necessity of calculating second-order derivatives for large matrices and potentially inverting these matrices to analyze their characteristics. These operations entail significant computational costs and can be prohibitive in terms of time, especially for models as large as those used in contemporary LLMs.

Given these considerations, our investigation into Hessian-based methods for LLM pruning not only enhances our theoretical understanding but also challenges us to develop more computationally feasible approaches. Future work could explore

Table 11. Pruning Metrics Discovered by Pruner-Zero with Corresponding Perplexity (PPL) Scores

Equation	PPL
$\left(\sigma\left(\left(|W|+|W|\right)+\sqrt{|G|}\right)\right)$	6.9175
$\left(\log\left(\left|\left(|W|\times|W|\right)\right|+\left(G\times|W|\right)\right)\right)$	6.7483
$\left(\left(\left(|W|+|W|\right)+\sqrt{|G|}\right)+\left(\frac{G}{\sqrt{\sqrt{|W|}}}\right)\right)$	6.9182
$\left(\left(|W|+|W|\right)+\sqrt{|G|}\right)$	6.9175
$\left(\tanh\left(\|W|\times|W\|\right)+\left(\frac{G}{\sqrt{|W|}}\right)\right)$	6.9814
$\left(\left|\left(|W|\times|W|\right)\right|\times\sqrt{|G|}\right)$	6.7701
$\left(\sigma\left(|W|\right)\times\sigma(|W|)\times\sigma(|G|)\right)$	6.7079

approximate methods to compute the Hessian or its relevant characteristics, such as using randomized algorithms or leveraging sparsity within the matrix to reduce the computational burden. By refining these methods, we aim to make Hessian-based pruning a practical tool for optimizing large-scale neural networks, ultimately contributing to more efficient and effective deployments of LLMs in real-world applications.

C.2. Operation Vocabulary

As presented in Table 10, we provide operations vocabulary for describing common unary and binary functions used in mathematical and machine learning operations. The vocabulary is organized into two sections for unary and binary operations respectively. For the unary operations section, this Table lists 13 operations denoted by unique operation IDs from U01 to U13. For each operation, it provides a code example, a description of input and output addresses/types, and a mathematical definition of the operation. Some common unary functions included are square, negative value, absolute value, logarithm, exponent, square root, and tanh. The binary operations section similarly lists four common operations - addition, subtraction, multiplication, and division. These binary operations take two matrix inputs and produce a matrix output. Overall, this operations vocabulary provides a standardized indexing and definition of commonly used mathematical and machine learning building blocks to facilitate the description and replication of computational models and algorithms.

C.3. Searched Metrics with Equations.

In the pursuit of optimizing neural network structures, Pruner-Zero has identified a set of novel pruning metrics, as illustrated in Table 11. These metrics are critical in the context of model simplification while maintaining, and in some instances, enhancing the performance of the neural network models. Each equation presented is accompanied by its Perplexity (PPL) score, which serves as an indicator of the model's predictive performance post-pruning. The scores close to the lower bound of this range suggest that the corresponding pruning metric has led to a model that balances the complexity and predictive capability effectively.

C.4. More Discussion over the Discovered SPM

We provide a concise theoretical analysis below and shed light on the insights gained from the investigated metric. Given an L-layer Large Language Model (LLM) denoted as $W = \{w_0, \ldots, w_L\}$ and a dataset $\mathcal{D} = \{(x_0, y_0), \ldots, (x_k, y_k)\}$, the task is to minimize the error E by optimizing the model parameters:

$$\min_{W} E(\mathcal{D}, W) = \min_{W} E(y|x, W) \tag{3}$$

In the context of pruning, the importance of a parameter is quantified by the error induced when the parameter is removed. The induced error can be measured as the squared difference with respect to a specific parameter w_m:

$$I_m = \left(E(\mathcal{D}, W) - E(\mathcal{D}, W_{w_m=0})\right)^2 \tag{4}$$

To approximate this error, we can use a second-order Taylor expansion:

$$I_m^{(2)}(W) = \left(g_m w_m - \frac{1}{2} w_m H_m W \right)^2 \tag{5}$$

where g_m is the gradient and H_m is the Hessian matrix. However, computing the Hessian matrix is computationally expensive, thus we employ a first-order expansion to obtain a more tractable approximation:

$$I_m^{(1)}(W) = (g_m w_m)^2 \tag{6}$$

This can be further simplified to:

$$I(W) = (W \times G)^2 \tag{7}$$

where W is the weight vector and G is the gradient vector.

Insight 1: Measurement of Kurtosis. We measure the kurtosis of the weights and gradients for the LLaMA-2-7B model, obtaining values of 0.7734 and 5.8203, respectively. This indicates that both weights and gradients significantly contribute to the Sparsity Promoting Metric (SPM) and influence the relative importance of different weights. To balance their contributions, we employed an evolutionary algorithm to automatically find the optimal scaling factors.

Insight 2: Inspired by Quantization Methods. Inspired by quantization methods such as AWQ (Lin et al., 2024), SmoothQuant (Xiao et al., 2023), and LLM-FP4 (Liu et al., 2023a), we employ scaling factors to balance weights and activations. This step is critical in reducing computational requirements without significantly compromising model accuracy. This approach underscores the importance of balancing weights and gradients when evaluating weight significance.

Based on these insights, we simplify Equation 3.2 to:

$$I(W) = (s_1(W) \times s_2(G))^2 \tag{8}$$

Here, s_1 and s_2 are scaling functions that control the contributions of weights and gradients to the final importance metric. Our goal is to identify these functions. Specifically, s_1 is a square operation that penalizes minor weights, and s_2 is a mean-median scaling (mms) operation that moderates the influence of gradients, preventing excessive dominance due to scaling disparities.

In Appendix C.2, we analyze the impact of these operations on final performance by using a correlation matrix to identify the most influential operations. Our findings highlight the critical role of the mms operation in optimizing the balance between weight and gradient contributions.

By systematically investigating and refining these scaling operations, we achieve a more nuanced understanding of parameter importance in LLM pruning. This approach not only enhances computational efficiency but also preserves model performance, demonstrating the efficacy of our proposed metric in practical applications.

D. Genetic Programming in SPM Optimization

In our study, we utilize genetic programming as outlined in Algorithm 1 to optimize the search space for pruning metrics in large language models. Genetic programming is a type of evolutionary algorithm that automates the generation of computer programs to solve specific computational problems. Our application of this methodology focuses on the development of symbolic pruning metrics for neural networks.

Implementation Details: Each individual in the genetic programming population represents a potential pruning metric, where the structure of each individual is modeled as a symbolic tree. Nodes within these trees correspond to either unary or binary operations, crucial for defining the computational logic of the pruning metric.

Evolutionary Process: The evolutionary process unfolds through several distinct stages, ensuring the continuous improvement of the population with respect to the pruning task:
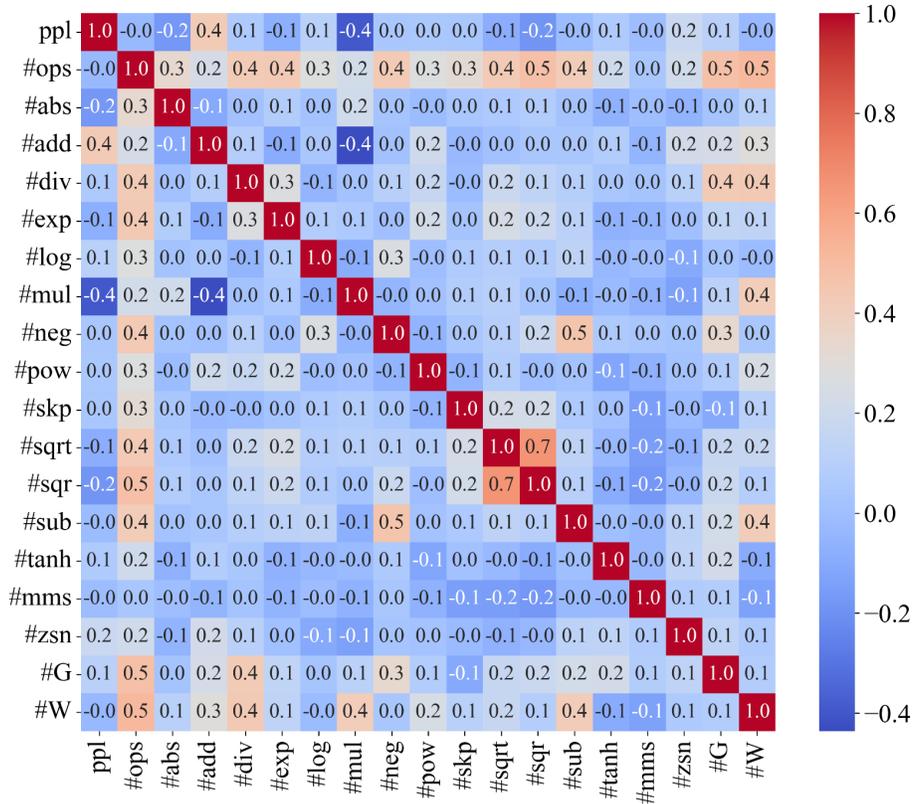
Figure 4. Correlation Matrix of Primitive Operations with Perplexity.

1. **Initialization:** The process begins with the generation of a random population of symbolic trees. This initial population forms the basis from which better-suited solutions can evolve over successive generations.

2. **Selection:** We implement a selection strategy where the top-k symbolic pruning metrics are chosen based on their performance. From this subset, two candidates are randomly sampled to serve as parents for the next generation. This selective approach ensures that only the most promising metrics contribute to the gene pool.

3. **Crossover:** The crossover operation is vital for introducing variability and combining beneficial traits from two parents. It involves scanning each parent using Depth-First Search (DFS) to identify a node. A node from each parent is then randomly selected, and their subtrees are exchanged to produce offspring. This method of recombination allows offspring to inherit and reconfigure traits from both parents, potentially leading to more effective pruning metrics.

4. **Mutation:** After crossover, each node in the offspring undergoes a mutation with a probability of 50%. This mutation can alter the node's operation, encouraging diversity within the population and aiding in the exploration of the search space.

Evaluation of Offspring: The newly generated offspring, now representing different symbolic pruning metrics, are evaluated based on their fitness. Fitness is measured by the perplexity of the pruned model, specifically the LLaMA-2-7B model under 50% sparsity. This measure helps determine how well the pruning metric performs in reducing the model's complexity without significantly impacting its effectiveness.

Iterative Process: The selection, crossover, and mutation processes are repeated for $\mathcal{N}$ iterations, or until a satisfactory pruning metric is found. Each iteration refines the population, ideally leading to increasingly effective pruning strategies over time.

By employing genetic programming, we leverage a robust evolutionary framework to explore a diverse array of pruning metrics systematically. This approach not only aids in discovering highly effective pruning techniques but also contributes to

Table 12. Data from trials at various depths

Depth	Trial1	Trial2	Trial3	Trial4	Average	Standard deviation
3	0.34	0.25	0.33	0.32	0.31	0.035
5	0.33	0.34	0.26	0.24	0.29	0.043
7	0.27	0.26	0.15	0.24	0.23	0.047
9	0.26	0.31	0.31	0.26	0.28	0.027

the broader understanding of how symbolic operations can impact the performance of large neural networks under sparsity constraints.

E. Expanding the Results

E.1. Detailed Correlation Analysis

Methodology. During the evolutionary search phase, a comprehensive dataset of symbolic pruning metrics was established alongside their associated perplexity scores. Metrics yielding a perplexity score below 7 were earmarked as potential candidates for effective pruning strategies. These metrics are systematically enumerated in Table 11 within Appendix C.3.

Visualization and Interpretation. Figure 4 visualizes the correlation coefficients between the frequency of various operations within the potential metrics and their corresponding perplexity scores. A heatmap representation facilitates an intuitive understanding of these relationships. The following points summarize the major findings from this analysis:

- Multiplication Operations (#mul): A strong positive correlation with perplexity was observed for multiplication operations, denoted by #mul in the heatmap. The frequency of multiplication within a metric tends to coincide with lower perplexity scores, suggesting that incorporating multiplication may be beneficial for pruning effectiveness.

- Opposite Operations (#sqrt and #sqr): A noteworthy correlation was detected between square roots (#sqrt) and squaring (#sqr) operations, indicating that they often occur in tandem and may have compensatory dynamics within the metrics.

- Min-Max Scaling (#mms): The min-max scaling operation, referred to as #mms in the heatmap, showed minimal correlation with perplexity and other operations. This implies that mms contributes a stabilizing effect on the metric's performance, reinforcing its utility in the pruning process.

Conclusion and Implications for Metric Design. The insights gleaned from this correlation analysis provide a data-driven foundation for refining symbolic pruning metrics. Specifically, the importance of multiplication and the potential balancing act between opposite operations offer avenues for enhancing metric sophistication. Meanwhile, the robust nature of min-max scaling underscores its value as a consistent component in pruning metrics. These analytical observations have been instrumental in the formulation of the symbolic pruning metrics presented in Section 3.2, and may inform future developments in pruning methodology.

E.2. Expanding the Zero-Shot Tasks

Detailed Results of Zero-shot Tasks. In the context of zero-shot learning, our evaluation encompasses a diverse set of tasks, as presented in Table 3. These tasks include BoolQ (Clark et al., 2019), RTE (Wang et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), ARC Easy and Challenge (Clark et al., 2018), and OpenbookQA (Mihaylov et al., 2018). To ensure the reproducibility of our results, we adhere to the settings and methodologies outlined in the Wanda study (Sun et al., 2024). Detailed task-wise performance metrics are systematically presented across several tables: Table 14 demonstrates the unstructured evaluation, while Tables 15 and 16 provide insights into structured evaluations. Furthermore, the nuanced performances under the LLaMA framework are captured in Tables 17, 18, and 19, offering a comprehensive understanding of the zero-shot capabilities of the models in question.

Discussion. Pruning involves removing redundant or less important parameters from the model, which can act as a form of regularization or denoising. (1) Regularization: Pruning can act as a form of regularization (Wang et al., 2020), reducing

Table 13. Perplexity of pruned LLaMA models on WikiText2. Our Pruner-Zero outperforms SparseGPT (Frantar & Alistarh, 2023), Wanda (Sun et al., 2024) and GBLM-Pruner (Das et al., 2023), achieving lower Perplexity without weight updates.

Method	Sparsity	LLaMA-7B	LLaMA-13B	LLaMA-30B
Dense	0	5.68	5.09	4.10
Magnitude (Han et al., 2016b)	0.5	17.29	20.21	7.54
SparseGPT (Frantar & Alistarh, 2023)	0.5	7.22	6.19	5.32
Wanda (Sun et al., 2024)	0.5	7.26	6.15	5.24
GBLM-Pruner2 (Das et al., 2023)	0.5	7.19	6.14	5.23
GBLM-Pruner1 (Das et al., 2023)	0.5	7.15	6.11	5.18
Pruner-Zero(Ours)	0.5	**6.95**	**5.94**	**5.01**
Magnitude (Han et al., 2016b)	2:4	42.54	18.36	9.11
SparseGPT (Frantar & Alistarh, 2023)	2:4	10.88	9.0	7.12
Wanda (Sun et al., 2024)	2:4	11.53	9.59	6.90
GBLM-Pruner2 (Das et al., 2023)	2:4	11.36	9.45	6.90
GBLM-Pruner1 (Das et al., 2023)	2:4	11.33	9.16	6.87
Pruner-Zero(Ours)	2:4	**10.61**	**8.11**	**6.51**
Magnitude (Han et al., 2016b)	4:8	16.83	13.87	7.62
SparseGPT (Frantar & Alistarh, 2023)	4:8	8.45	7.44	6.18
Wanda (Sun et al., 2024)	4:8	8.57	7.41	5.97
GBLM-Pruner2 (Das et al., 2023)	4:8	8.50	7.38	5.94
GBLM-Pruner1 (Das et al., 2023)	4:8	8.48	7.26	5.89
Pruner-Zero(Ours)	4:8	**8.12**	**6.81**	**5.65**

overfitting by applying dropout to the model. Dense models might overfit the training data or learn spurious correlations that do not generalize well to unseen tasks. In contrast, the pruned model might generalize better to new tasks due to this implicit regularization. (2) Denoising: In dense models, some parameters act as noise (Diao et al., 2023), contributing little to the model's performance or even detracting from it. Pruning eliminates much of this noise and thus leads to a clearer signal within the model. This denoising effect can make weakly learned facts or knowledge more accessible to the model, leading to improved performance on zero-shot tasks. For instance, LASER (Sharma et al., 2024) finds that even under extreme reductions, the performance of LLM on natural language understanding tasks continues to improve. Recent research (Li et al., 2024c) suggests a similar phenomenon in quantization, observing that quantization even brings notable accuracy gain for multi-choice zero-shot tasks. They conducted experiments to verify that there is less uncertainty for the quantized model, resulting in the phenomenon that quantization brings accuracy gain. Additionally, previous methods like Wanda (Sun et al., 2024) and SparseGPT (Frantar & Alistarh, 2023) have also demonstrated improved performance on zero-shot tasks after pruning. This suggests that these post-training pruning techniques can be effective in enhancing the reasoning capabilities of language models, particularly in scenarios where they need to rely on their pre-trained knowledge without task-specific fine-tuning.

E.3. Comparison with Previous Pruning Methods

In this section, we extend the scope of pruning methods traditionally applied to BERT (Devlin et al., 2019) and evaluate their efficacy on larger language models (LLMs). Table 20 provides an overview of these pre-existing pruning methods, primarily utilized for BERT. A notable difference between these methods and our approach is their integration of pruning with the fine-tuning process. Additionally, BERT-specific pruning techniques typically focus on downstream task performance, in contrast to our aim of preserving the general language modeling capabilities of pre-trained LLMs.

Adapting these methods for LLM pruning, we employ the pre-training auto-regressive loss as the metric to guide the pruning process. Our evaluation considers two scenarios: post-training pruning and post-training pruning followed by a constrained period of fine-tuning, limited to one day. The effectiveness of the pruning is determined using the metrics outlined in Table 20. In the post-training pruning scenario, these metrics are directly applied to the LLMs. The pruned models are then subjected to fine-tuning within the stated computational constraints. The results, summarized in Table 21, reveal that the traditional pruning methods, when adapted to LLMs, do not yield effective outcomes. This observation underscores

Table 14. Accuracies (%) of LLaMA for 7 zero-shot tasks with unstructured 50% sparsity.

Params	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Mean
	Dense	75.05	66.43	56.92	69.93	75.34	41.89	34.40	59.99
7B	Magnitude	54.59	54.51	45.49	59.19	58.84	33.53	22.40	46.94
	SparseGPT	**72.05**	54.15	51.43	**67.88**	**71.38**	37.71	30.00	54.94
	Wanda	71.22	55.60	51.85	66.06	69.11	36.86	28.80	54.21
	Pruner-Zero	68.87	**63.54**	**69.12**	67.80	69.19	**38.82**	**39.60**	**59.56**
	Dense	77.89	70.4	59.94	72.77	77.40	46.50	33.20	62.59
13B	Magnitude	54.89	51.26	44.16	63.14	58.80	33.79	27.20	47.61
	SparseGPT	**76.97**	61.01	54.95	71.67	72.47	41.98	31.20	58.61
	Wanda	75.90	**62.82**	55.71	**71.98**	73.19	**43.52**	32.20	59.33
	Pruner-Zero	74.77	59.57	**74.00**	71.19	**74.12**	42.06	**43.00**	**62.67**
	Dense	82.69	66.79	63.35	75.69	80.30	52.82	36.00	65.38
30B	Magnitude	64.34	50.18	50.59	66.54	72.39	43.77	29.00	53.83
	SparseGPT	**82.32**	62.45	59.15	**75.22**	78.96	48.56	35.00	63.09
	Wanda	81.90	**65.34**	60.93	73.48	**79.29**	**49.66**	34.60	63.60
	Pruner-Zero	81.04	67.15	**79.25**	73.32	78.07	48.81	**44.80**	**67.49**
	Dense	84.83	69.68	64.54	77.27	81.40	52.90	38.20	66.97
65B	Magnitude	79.15	62.45	61.90	74.74	76.40	49.57	35.00	62.74
	SparseGPT	84.60	70.76	63.90	**77.43**	79.35	50.85	37.20	66.30
	Wanda	**84.70**	**71.48**	64.55	76.87	**79.75**	50.51	38.80	66.67
	Pruner-Zero	84.45	70.04	**81.75**	75.77	79.45	**52.65**	**44.60**	**69.81**

the necessity for developing pruning techniques that are more suited to the unique characteristics of large-scale language models.

E.4. Comparative Analysis of Pruning Techniques

This section extends the discourse on pruning methodologies to the Tiny-LLaMA (Zhang et al., 2024a) and OPT (Zhang et al., 2022b), which is engineered for optimal performance with a minimal parameter set. Table 22 benchmarks Tiny-LLaMA against a suite of pruning approaches, which include unstructured pruning as well as structured pruning with specific ratios, namely 2:4 and 4:8.

We scrutinize a range of pruning methods: Magnitude-based pruning, the SparseGPT algorithm, the Wanda technique, and our proposed Pruner-Zero. The post-pruning performance metrics illuminate the interplay between model size and operational effectiveness. While Magnitude pruning serves as a ubiquitous benchmark, it demonstrates a variable efficacy across the pruning ratios, particularly underperforming at the 2:4 ratio. Contrastingly, SparseGPT and Wanda deliver enhanced capabilities over Magnitude pruning, attesting to their prowess in preserving model competence during downsizing. Our Pruner-Zero is noteworthy for its competitive performance, especially prominent at the unstructured and 4:8 pruning thresholds, which underscores its suitability as an adept pruning strategy for Tiny-LLaMA. The empirical evidence accentuates the proficiency of Pruner-Zero in closely approximating the performance of models with lower levels of pruning, a critical consideration for deployment in resource-limited settings where compact models are requisite without considerable compromise in functionality.

Subsequently, Table 4 delineates the pruning performance across the spectrum of OpenAI's OPT models, with sizes ranging from 125 million to 13 billion parameters. The methodologies under scrutiny here comprise a baseline dense model, Magnitude pruning, SparseGPT, Wanda, and Pruner-Zero, each targeting a consistent sparsity quotient of 50%.

The results conspicuously reveal that Pruner-Zero outperforms other approaches that forego a weight update phase, specifically Magnitude pruning and Wanda. This is manifested in markedly lower perplexity scores for models pruned via Pruner-Zero, indicating substantial retention of predictive capacity post-pruning.

Table 15. Accuracies (%) of LLaMA for 7 zero-shot tasks with 4:8 sparsity.

Params	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Mean
7B	Dense	75.05	66.43	56.92	69.93	75.34	41.89	34.40	59.99
	Magnitude	51.19	50.54	46.73	60.69	58.96	30.89	23.20	46.03
	SparseGPT	**73.06**	58.12	47.88	**65.98**	**66.75**	32.42	25.40	52.80
	Wanda	70.97	58.24	46.81	65.83	65.53	33.97	28.00	52.76
	Pruner-Zero	69.97	**60.65**	**63.70**	64.88	64.23	**34.64**	**35.60**	**56.24**
13B	Dense	77.89	70.40	59.94	72.77	77.40	46.50	33.20	62.59
	Magnitude	61.07	51.26	48.91	65.11	63.26	35.67	28.40	50.53
	SparseGPT	**76.61**	57.76	51.24	70.17	**71.17**	37.20	27.80	55.99
	Wanda	74.89	**57.89**	51.26	**70.56**	70.29	**37.97**	29.80	56.09
	Pruner-Zero	69.20	53.07	**70.01**	69.22	70.50	39.42	**41.80**	**59.03**
30B	Dense	82.69	66.79	63.35	75.69	80.30	52.82	36.00	65.38
	Magnitude	63.55	50.18	49.45	65.75	73.36	42.83	29.60	53.53
	SparseGPT	**78.69**	**61.73**	56.15	**74.35**	76.94	46.08	31.60	60.79
	Wanda	77.38	58.80	58.79	74.28	**77.34**	**46.46**	34.00	61.00
	Pruner-Zero	77.71	58.48	**75.76**	72.14	75.72	45.90	**42.60**	**64.04**
65B	Dense	84.83	69.68	64.54	77.27	81.40	52.90	38.20	66.97
	Magnitude	74.95	68.23	60.85	74.27	76.45	47.61	32.80	62.17
	SparseGPT	**84.35**	68.95	61.00	**77.19**	78.75	48.46	35.40	64.87
	Wanda	84.29	**70.92**	59.54	76.64	**79.00**	48.83	35.60	64.97
	Pruner-Zero	81.50	66.43	**79.15**	76.48	78.50	**50.00**	**44.20**	**68.04**

Remarkably, Pruner-Zero achieves these benchmarks sans the weight update step, which is indispensable for SparseGPT. Nevertheless, Pruner-Zero showcases results that are congruent with those of SparseGPT, underscoring its robustness and streamlined efficiency as a pruning modality. This is particularly salient for the 13B parameter model, where Pruner-Zero's performance virtually mirrors that of SparseGPT, thus solidifying its stance as an efficacious pruning strategy.

E.5. Pruning Time Comparison.

The efficiency of pruning algorithms is critical in their application to large language models, such as the LLaMa 2-7b. Pruning time, an important metric for evaluating such algorithms, depends on various computational factors, including gradients, Hessian matrices, weight adjustments, and activation computations. The sequence of time consumption for these components generally follows:

$$Time(Hessian) > Time(Gradient) > Time(Activation) > Time(Weight)$$

This hierarchy highlights the computational intensity of calculating the Hessian matrix, which requires the most time due to the complexity of computing second-order derivatives for each element in the weight matrix. To enhance the pruning process's efficiency, gradients are pre-computed offline. This preparatory step significantly speeds up the pruning phase by eliminating real-time gradient computation, thus streamlining the entire procedure.

A comparative analysis of pruning time was conducted under fair conditions, where extraneous factors such as dataset loading and evaluation procedures were systematically excluded to ensure a level comparison field. The following table presents the pruning times and the associated perplexity results of various pruning methods, including our Pruner-Zero, SparseGPT (Frantar & Alistarh, 2023), and Wanda (Sun et al., 2024):

The data reveals that Pruner-Zero is twice as fast as SparseGPT (Frantar & Alistarh, 2023) and is slightly slower compared to Wanda (Sun et al., 2024), taking only 10% more time. However, this modest increase in time is offset by a significant improvement in model performance, as evidenced by Pruner-Zero's lower perplexity score (6.95) compared to Wanda's (7.26). This improvement underscores Pruner-Zero's ability to balance efficiency with effectiveness, optimizing not just for speed but also for enhancing the model's linguistic capabilities post-pruning.

Table 16. Accuracies (%) of LLaMA for 7 zero-shot tasks with 2:4 sparsity.

Params	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Mean
7B	Dense	75.05	66.43	56.92	69.93	75.34	41.89	34.40	59.99
	Magnitude	53.09	55.60	42.30	59.91	53.28	27.13	21.80	44.73
	SparseGPT	**70.46**	**60.65**	42.99	**64.88**	**61.49**	30.12	23.60	50.60
	Wanda	69.30	51.99	42.06	62.75	60.94	28.07	24.60	48.53
	Pruner-Zero	66.54	53.79	**56.29**	62.19	60.56	30.63	34.40	**52.06**
13B	Dense	77.89	70.40	59.94	72.77	77.40	46.50	33.20	62.59
	Magnitude	60.95	49.10	45.81	62.75	58.75	31.06	27.60	48.00
	SparseGPT	**72.14**	**55.23**	48.11	**68.98**	66.71	34.98	26.40	53.22
	Wanda	70.21	53.43	46.74	68.82	65.82	33.87	27.20	52.30
	Pruner-Zero	69.20	53.07	**64.67**	68.51	**67.13**	**36.09**	**38.80**	**56.78**
30B	Dense	82.69	66.79	63.35	75.69	80.30	52.82	36.00	65.38
	Magnitude	65.11	52.35	51.72	66.22	70.88	38.23	27.60	53.16
	SparseGPT	**75.60**	62.13	53.10	72.61	**75.13**	41.98	31.80	58.91
	Wanda	74.68	**63.80**	54.41	**72.93**	74.41	42.06	32.20	59.21
	Pruner-Zero	70.92	62.09	**72.47**	70.24	73.36	**42.32**	**42.60**	**62.00**
65B	Dense	84.83	69.68	64.54	77.27	81.40	52.90	38.20	66.97
	Magnitude	77.9	64.98	58.65	72.85	75.15	45.05	34.40	61.28
	SparseGPT	83.15	65.34	57.20	**76.72**	78.20	45.18	32.20	62.57
	Wanda	**83.58**	**66.79**	56.36	75.82	**78.23**	45.56	33.60	62.84
	Pruner-Zero	79.45	60.29	**76.95**	73.48	77.20	**46.76**	**43.80**	**65.42**

E.6. Performance on Higher Sparisty Ratio

For higher sparsity at 60%, to further explore the efficacy of various pruning methods under increased constraints. The results of this analysis are encapsulated in Table 24. At this sparsity threshold, the pruning methods exhibit varied performance across different model sizes of both LLaMA and LLaMA-2. Our method, Pruner-Zero, demonstrates commendable robustness, consistently achieving lower perplexity scores across all model configurations. This is indicative of Pruner-Zero's effectiveness in maintaining model performance even under substantial reduction in parameters. Notably, at 60% sparsity, the method exhibits a competitive edge over other approaches, including Wanda and SparseGPT, especially with larger model sizes.

E.7. Robustness Assessment

To evaluate the stability of our Pruner-Zero, we conduct experiments for LLaMA and LLaMA-2 under different Seeds to test their robustness in generating consistent performance outcomes. Our results, as encapsulated in Table 25, exhibit that the variability in performance, as indicated by the standard deviation (STD), remains within an acceptable range, confirming the inherent stability of our Pruner-Zero. Specifically, the LLaMA model shows a moderate standard deviation at the 2:4 ratio, which suggests that while there is some variability, it does not significantly detract from the overall reliability of the model. In comparison, the LLaMA-2 model demonstrates an even lower standard deviation across all ratios, notably at the 50% ratio, which underlines a high level of consistency in performance despite the initialization variability.

The average (AVG) performance metrics further reinforce the conclusion that our Pruner-Zero maintains a dependable performance level. For instance, the average performance metric for LLaMA at the 50% ratio is 6.8799, whereas for LLaMA-2 it slightly improves to 6.2511, which could be indicative of model-specific optimizations or inherent architectural advantages. These findings suggest that our method not only sustains performance across various initial conditions but may also benefit from further tuning specific to the model variant being utilized.

Moreover, the systematic evaluation across multiple seeds provides a comprehensive understanding of the performance spectrum that one can anticipate when employing these models in practical scenarios. It also lays a foundation for future explorations into the causes of variability, enabling targeted enhancements to our Pruner-Zero for improved stability and performance consistency.

Table 17. Accuracies (%) of LLaMA-2 for 7 zero-shot tasks with unstructured 50% sparsity.

Params	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Mean
	Dense	77.74	62.82	57.17	68.90	76.39	43.52	31.40	59.71
7B	Magnitude	63.00	**57.04**	49.13	63.30	64.10	34.64	26.80	51.14
	SparseGPT	75.02	54.15	52.37	**69.85**	**73.27**	**39.85**	29.20	56.24
	Wanda	**75.99**	53.43	52.49	68.19	72.77	39.59	31.20	56.24
	Pruner-Zero	70.92	53.07	**69.10**	65.98	71.89	39.76	**41.40**	**58.87**
	Dense	80.52	65.34	60.06	72.22	79.42	48.46	35.20	63.03
13B	Magnitude	57.61	55.96	54.40	65.27	70.54	38.40	27.80	52.85
	SparseGPT	81.44	65.34	55.83	**72.77**	74.83	42.24	32.60	60.72
	Wanda	**81.84**	64.02	56.90	71.35	**76.18**	43.52	32.00	60.83
	Pruner-Zero	79.79	**66.06**	**74.60**	70.88	75.67	**43.60**	**43.20**	**64.83**
	Dense	83.40	67.87	66.10	78.06	82.55	54.44	37.20	67.08
70B	Magnitude	70.55	60.65	61.50	73.48	75.70	49.23	35.40	60.93
	SparseGPT	**83.55**	70.40	63.80	**78.85**	**82.40**	53.75	38.20	67.28
	Wanda	82.50	**73.65**	64.10	78.14	80.80	52.65	37.40	67.03
	Pruner-Zero	83.20	73.29	**81.70**	76.72	81.10	**55.12**	**46.60**	**71.10**

F. Searched Metrics with Expressions.

Table 26 provides a symbolic string representation of the pruning metrics, offering an alternative perspective on the structural patterns that emerged from Pruner-Zero's search algorithm. Symbol "#" denotes the left operation as a unary operation. For each string, we can define the only symbolic pruning metric exclusively. These representations are directly linked to the mathematical equations in Table 11, with the accompanying PPL values validating their efficacy. This string-based format encapsulates the operational essence of the pruning metrics, allowing for an abstract view that can be particularly useful for algorithm interpretation and further computational analysis. The variations in PPL scores across different metrics underscore the sensitivity of model performance to the pruning strategy employed, highlighting the importance of careful selection and evaluation of pruning metrics in model optimization processes.

G. Limitations and Future Works

Limitations: Our Pruner-Zero framework, while pioneering in unstructured pruning at a 50% sparsity level, encounters specific limitations in its specialization and evaluative scope. It primarily excels within its tested scenario but lacks empirical support for its effectiveness in alternative configurations such as structured pruning or higher sparsity levels, where challenges escalate due to the complexity of removing structured network components or a greater proportion of weights. It is possible to have better performance under higher sparsity levels or structured pruning. Furthermore, the evaluation of Pruner-Zero has predominantly focused on perplexity and accuracy within constrained contexts, specifically on the Wikitext2 dataset and zero-shot tasks. While these measures offer insights into language modeling and general reasoning, they fall short of capturing the full breadth of capabilities essential for Large Language Models (LLMs), such as in-context learning, commonsense reasoning, instruction following, and self-calibration. These abilities are vital for gauging the comprehensive intelligence and versatility of LLMs, particularly in assessing their operational efficacy post-pruning.

Future Work: Recognizing these gaps, future research will aim to broaden the evaluative framework of the Pruner-Zero to encompass a wider array of capabilities intrinsic to LLMs. This expansion will involve devising tests and benchmarks specifically tailored to accurately measure the impacts of pruning on aspects like in-context learning, commonsense reasoning, and other advanced functionalities. The goal is to ensure that pruning not only mitigates model size and computational demands but also preserves or enhances the model's capacity for complex tasks and exhibiting human-like intelligence.

Table 18. Accuracies (%) of LLaMA-2 for 7 zero-shot tasks with 4:8 sparsity.

Params	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Mean
	Dense	77.74	62.82	57.17	68.90	76.39	43.52	31.40	59.71
7B	Magnitude	63.00	52.35	50.08	62.43	64.73	35.92	26.00	50.64
	SparseGPT	72.69	**55.23**	48.20	**68.11**	**69.15**	35.84	27.40	53.80
	Wanda	**73.91**	53.79	46.45	66.61	66.71	34.13	25.80	52.49
	Pruner-Zero	67.52	53.43	**63.06**	64.48	66.96	**36.69**	**38.60**	**55.82**
	Dense	80.52	65.34	60.06	72.22	79.42	48.46	35.20	63.03
13B	Magnitude	63.33	57.76	53.96	64.40	68.48	35.75	26.00	52.81
	SparseGPT	79.97	**66.79**	52.01	**70.64**	73.61	41.04	30.00	59.15
	Wanda	**80.26**	65.62	52.05	69.48	73.88	41.54	28.40	58.75
	Pruner-Zero	79.45	60.65	**70.00**	67.64	**74.03**	**41.64**	**40.40**	**61.97**
	Dense	83.40	67.87	66.10	78.06	82.55	54.44	37.20	67.08
70B	Magnitude	70.95	59.21	60.05	74.11	76.25	46.76	34.60	60.28
	SparseGPT	82.20	**72.20**	61.45	**77.82**	**80.85**	51.19	35.20	65.84
	Wanda	**84.30**	71.80	61.90	76.24	80.40	51.80	36.00	66.06
	Pruner-Zero	83.55	71.12	**80.40**	75.37	80.45	**52.47**	**46.20**	**69.94**

Table 19. Accuracies (%) of LLaMA-2 for 7 zero-shot tasks with 2:4 sparsity.

Params	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Mean
	Dense	77.74	62.82	57.17	68.90	76.39	43.52	31.40	59.71
7B	Magnitude	56.23	51.35	42.27	60.93	59.18	27.31	21.80	45.58
	SparseGPT	**70.52**	**58.84**	43.26	**66.69**	**64.10**	29.97	23.20	50.94
	Wanda	67.65	53.07	40.92	62.43	61.78	31.20	24.20	48.75
	Pruner-Zero	69.14	53.43	**54.68**	60.54	61.57	**32.17**	**32.60**	**52.02**
	Dense	80.52	65.34	60.06	72.22	79.42	48.46	35.20	63.03
13B	Magnitude	65.69	54.15	50.13	62.04	62.46	31.74	23.00	49.89
	SparseGPT	76.79	59.38	46.58	**68.67**	**70.62**	36.60	25.40	54.86
	Wanda	76.80	**61.22**	47.82	66.90	69.24	36.82	26.40	55.03
	Pruner-Zero	**77.89**	56.68	**63.37**	67.72	69.70	**37.29**	**36.00**	**58.38**
	Dense	83.40	67.87	66.10	78.06	82.55	54.44	37.20	67.08
70B	Magnitude	73.20	57.04	58.40	74.27	76.15	45.22	35.40	59.95
	SparseGPT	79.50	**70.76**	59.00	**76.64**	78.95	48.55	33.80	63.89
	Wanda	**82.20**	69.85	59.34	76.23	79.30	47.26	34.80	64.14
	Pruner-Zero	81.40	67.87	**77.60**	74.11	**79.45**	**50.00**	**43.40**	**67.69**

Table 20. Summary of prior pruning methods on BERT.

Pruning Method	Pruning Type	Pruning Metric	Training Procedure
SNIP (Lee et al., 2019)	Unstructured	Loss Sensitivity	Pruning at Initialization
BERT-LTH (Chen et al., 2020)	Unstructured	Magnitude	Fine-tuning BERT
Movement (Sanh et al., 2020)	Unstructured	Loss Sensitivity	Fine-tuning BERT
Platon (Zhang et al., 2022a)	Unstructured	Loss Sensitivity	Fine-tuning BERT
PINS (Ren & Zhu, 2023)	Unstructured	Loss Sensitivity	Fine-tuning BERT

Table 21. Comparisons with prior pruning methods on BERT (unstructured 50% sparsity).

			Pruning method					
Model	**Dense**	**Fine-tuning**	**SNIP**	**BERT-LTH**	**Movement**	**Platon**	**PINS**	Pruner-Zero
LLaMA-7B	5.68	✗	231.48	17.29	349.33	124.91	89.12	6.95
		✓	102.32	12.43	168.17	102.34	72.13	6.74

Table 22. Comparative Pruning Performance on Tiny-LLaMA (Zhang et al., 2024a).

Tiny-LLaMA	**Unstructured** (50%)	**Structured (2:4)**	**Structured (4:8)**
Magnitude	21.64	64.07	23.13
SparseGPT	10.71	18.79	13.84
Wanda	11.21	27.17	16.18
Pruner-Zero	10.65	22.17	14.47

Table 23. Comparison of Pruning Times and Perplexity Across Different Methods

Pruning Method	**Pruning Time Only (s)**	**Perplexity**
Magnitude	0.92	17.29
Wanda	402.25	7.26
SparseGPT	1178.62	7.22
Pruner-Zero	444.12	6.95

Table 24. WikiText2 validation perplexity of pruned LLaMA and LLaMA-2 models with unstructured 60% sparsity.

			LLaMA				**LLaMA-2**		
Method	**Weight Update**	**Sparsity**	**7B**	**13B**	**30B**	**65B**	**7B**	**13B**	**70B**
Dense	-	0%	5.68	5.09	4.77	3.56	5.12	4.57	3.12
Magnitude	✗	60%	6e2	2e2	27.67	9.34	4e3	11.23	8.21
SparseGPT	✓	60%	10.51	8.56	6.66	5.82	**9.58**	7.80	4.98
Wanda	✗	60%	10.66	8.56	6.49	5.83	9.71	7.75	4.98
Pruner-Zero	✗	60%	**9.83**	**7.66**	**6.22**	**5.31**	**9.58**	**6.90**	**4.64**

Table 25. Performance Metrics (Perplexity) of LLaMA Models of Pruner-Zero Across Different Seeds

LLaMA	**Seed-0**	**Seed-1**	**Seed-2**	**Seed-3**	**AVG**	**STD**
50%	6.8827	6.8819	6.8959	6.8592	6.8799	0.0132
2:4	10.3582	10.3536	10.4872	10.2486	10.3997	0.0619
4:8	8.0697	8.0505	8.0171	8.0006	8.0345	0.0271
LLaMA-2	**Seed-0**	**Seed-1**	**Seed-2**	**Seed-3**	**AVG**	**STD**
50%	6.2467	6.2587	6.2449	6.2542	6.2511	0.0056
2:4	10.6715	10.5575	10.6180	10.6440	10.6227	0.0421
4:8	7.6963	7.6748	7.7072	7.6512	7.6824	0.0214

Table 26. Expressions of Searched Pruning Metrics and Their Associated PPL Values on LLaMA-7b

Expression	Perplexity
$(((W) \text{ mms } (\#)) \text{ add } (G))\text{sqrt}(\#)$	7.670979499816894
$((G)\exp(\#))\text{mul}((W)\text{mul}((G) \text{ abs } (\#)))$	6.774543285369873
$(((G) \text{ add } (W))\text{div}((W)\exp(\#)))\text{mul}(((W)\text{mul}(W))\text{sqr}(\#))$	7.059216022491455
$(((G) \text{ mms } (\#)) \text{ add } ((W)\tanh(\#)))\text{sqrt}(\#)$	7.096769332885742
$((((W) \text{ mms } (\#)) \text{ add } ((G) \text{ pow } (\#))) \text{ neg } (\#)) \text{ pow } (\#)$	6.921246528625488
$((((W)\text{sqrt}(\#)) \text{ abs } (\#)) \text{ sub } (((W) \text{ sub } (G))\tanh(\#)))\log(\#)$	6.933580875396728
$((G)\text{mul}(W))$	6.770424842834473
$((W) \text{ mms } (\#)) \text{ add } (G)$	7.671018600463867
$((G) \text{ skp } (\#))\text{mul}((W)\text{div}(W))$	6.770524024963379
$((G)\text{sqr}(\#)) \text{ add } ((W) \text{ abs } (\#))$	6.990375518798828
$((G) \text{ skp } (\#))\text{mul}(((W)\tanh(\#))\tanh(\#))$	6.770989894866943
$(((W)\tanh(\#))\text{mul}((G) \text{ mms } (\#)))\text{sqr}(\#)$	6.821769714355469
$(((G)\text{div}(G))\text{sqr}(\#)) \text{ sub } (((W)\tanh(\#)) \text{ sub } ((W) \text{ abs } (\#)))$	7.053229808807373
$(((G)\text{mul}(W)) \text{ add } ((W)\text{mul}(W)))\text{div}(((W) \text{ sub } (W)) \text{ add } ((W) \text{ pow } (\#)))$	6.748305320739746
$((((G) \text{ abs } (\#))\exp(\#))\text{mul}(W)) \text{ sub } ((((W) \text{ abs } (\#))\text{div}((W)\exp(\#)))\exp(\#))$	7.161033630371094
$(((G) \text{ pow } (\#))\exp(\#))\text{mul}(((W) \text{ skp } (\#))\text{div}(W))$	7.531239986419678
$((((W) \text{ abs } (\#))\text{mul}((G) \text{ skp } (\#))) \text{ pow } (\#))\log(\#)$	6.77089786529541
$((((W)\tanh(\#))\text{mul}((G)\text{mul}(G)))\text{div}(((G)\text{div}(W))\tanh(\#))) \text{ mms } (\#)$	7.190669059753418
$((((G) \text{ pow } (\#)) \text{ add } ((W)\exp(\#))) \text{ abs } (\#))\text{div}((((W) \text{ mms } (\#))\text{div}((G) \text{ zsn } (\#)))\exp(\#))$	6.987056732177734
$((((W)\text{mul}(W)) \text{ skp } (\#)) \text{ zsn } (\#)) \text{ sub } ((((W)\tanh(\#)) \text{ abs } (\#)) \text{ sub } (((G)\text{sqr}(\#)) \text{ zsn } (\#)))$	7.274583339691162
$((((W)\tanh(\#))\exp(\#))\text{sqrt}(\#)) \text{ sub } ((((W)\log(\#)) \text{ add } ((G)\text{sqrt}(\#))) \text{ neg } (\#))$	7.018911838531494
$(((((W) \text{ add } (G))\exp(\#))\text{sqr}(\#)) \text{ sub } ((G) \text{ sub } ((G) \text{ neg } (\#)))))\text{sqrt}(\#)$	6.949575901031494
$(((((G)\text{div}(G)) \text{ mms } (\#))\text{mul}(((W)\text{mul}(W))\text{sqrt}(\#)))\exp(\#))\tanh(\#)$	6.821941375732422
$((((((W)\exp(\#))\text{sqr}(\#))\log(\#))\exp(\#)) \text{ add } (((((G) \text{ pow } (\#))\text{sqr}(\#)) \text{ skp } (\#))\text{sqr}(\#))$	7.282495498657227
$((((W) \text{ abs } (\#))\text{mul}((G) \text{ skp } (\#))) \text{ pow } (\#)) \text{ sub } ((((W)\log(\#)) \text{ add } ((G)\text{sqrt}(\#))) \text{ neg } (\#))$	7.013281345367432
$((((W)\text{mul}(W)) \text{ abs } (\#))\text{sqrt}(\#)) \text{ sub } ((((W)\log(\#)) \text{ add } ((G)\text{sqrt}(\#))) \text{ neg } (\#))$	7.023911952972412
$((W)\text{mul}(G))\text{mul}(((W)\text{mul}(W)) \text{ mms } (\#))$	6.785792350769043
$((((W)\text{mul}(W)) \text{ abs } (\#))\text{mul}((G) \text{ mms } (\#))) \text{ sqr } (\#)$	6.707982063293457
$(((G) \text{ abs } (\#)) \text{ abs } (\#)) \text{ add } (((W)\log(\#)) \text{ mms } (\#))$	6.763171672821045
$(((W)\tanh(\#))\text{mul}((G)\text{mul}(W)))\text{div}(((W)\text{mul}\exp(\#))\exp(\#))$	6.712837696075439
$((((W)\log(\#))\text{mul}((G)\text{mul}(W)))\text{sqrt}(\#))\text{sqr}(\#)$	6.843364715576172
$(((W)\text{add}(W))\text{add}((G)\text{sqr}(\#))) \text{ mms } (\#)$	6.917529582977295
$((((W)\text{mul}(W))\text{abs}(\#))\text{add}((G)\text{mul}(W))) \text{ log } (\#)$	6.748325347900391
$(((W)\text{add}(W))\text{add}((G)\text{sqr}(\#))) \text{ add } ((((G)\text{pow}(\#))\exp(\#))\text{div}(((W)\text{sqrt}(\#))\text{sqrt}(\#)))$	6.918227195739746
$((W)\text{add}(W)) \text{ add } ((G)\text{sqr}(\#))$	6.917529582977295
$((((W)\text{mul}(W))\text{abs}(\#))\tanh(\#))\text{add}((((G)\text{pow}(\#))\exp(\#))\text{div}((W)\text{sqrt}(\#)))$	6.981475353240967
$((((W)\text{mul}(W))\text{abs}(\#))\text{mul}((G)\text{sqr}(\#)))\text{skp}(\#)$	6.770103454589844
$(((W)\text{add}(W))\text{add}((G)\text{sqr}(\#)))\log(\#)$	6.917529582977295
$(((G)\text{mul}(W))\exp(\#)) \text{ sqr } (\#)$	6.770689487457275
$((G)\text{mul}((W)\tanh(\#))) \text{ mms } (\#)$	6.770922660827637
$((W)\text{pow}(\#)) \text{ mul } (((G)\text{sqr}(\#))\text{mul}(G))$	6.982866764068603
$(((W)\text{abs}(\#))\text{pow}(\#)) \text{ mul } (((G)\text{mms}(\#))\text{mms}(\#))$	6.707982063293457
$((((W) + (G))\text{abs}(\#))\text{mul}(((W)\text{skp}(\#))\text{sqr}(\#))) \text{ sqr } (\#)$	6.735072135925293
$((((G)\tanh(\#))\text{mul}((W)\text{mms}(\#)))\tanh(\#))\tanh(\#)$	6.775103092193603
$((W)\text{pow}(\#))\text{mul}(((W)\text{sqr}(\#))\text{mul}(G))$	6.952877998352051