Search for Efficient Large Language Models

Xuan Shen[1], **Pu Zhao**[1], **Yifan Gong**[1], **Zhenglun Kong**[2], **Zheng Zhan**[1],
Yushu Wu[1], **Ming Lin**[3], **Chao Wu**[1], **Xue Lin**[1], **Yanzhi Wang**[1]
[1]Northeastern University, [2]Harvard University, [3]Oracle
{shen.xu, yanz.wang}@northeastern.edu

Abstract

Large Language Models (LLMs) have long held sway in the realms of artificial intelligence research. Numerous efficient techniques, including weight pruning, quantization, and distillation, have been embraced to compress LLMs, targeting memory reduction and inference acceleration, which underscore the redundancy in LLMs. However, most model compression techniques concentrate on weight optimization, overlooking the exploration of optimal architectures. Besides, traditional architecture search methods, limited by the elevated complexity with extensive parameters, struggle to demonstrate their effectiveness on LLMs. In this paper, we propose a training-free architecture search framework to identify optimal subnets that preserve the fundamental strengths of the original LLMs while achieving inference acceleration. Furthermore, after generating subnets that inherit specific weights from the original LLMs, we introduce a reformation algorithm that utilizes the omitted weights to rectify the inherited weights with a small amount of calibration data. Compared with SOTA training-free structured pruning works that can generate smaller networks, our method demonstrates superior performance across standard benchmarks. Furthermore, our generated subnets can directly reduce the usage of GPU memory and achieve inference acceleration. Code: https://github.com/shawnricecake/search-llm

1 Introduction

Large Language Models (LLMs) [1, 2, 3, 4, 5, 6] are renowned for their exceptional performance across various domains of artificial intelligence research. There is a growing demand for constructing LLMs for extensive applications across a multitude of popular platforms. However, the computational and storage costs have restricted LLMs from deployment on various devices for wide applications. Take the GPT-3 model as an example, with its 175 billion parameters [6], it requires more than 326GB of memory in FP16 format. This exceeds the memory capabilities of even the most sophisticated GPUs, far surpassing available memory on resource-constrained devices. To address these challenges, a variety of compression techniques focusing on weight optimization have been developed, including weight pruning [7, 8, 8, 9, 10, 11, 12, 13, 14, 15, 16], quantization [17, 18, 19], and knowledge distillation [20, 21, 22]. The extensive research in the compression direction indicates the substantial redundancy within LLMs.

Besides optimizing model weights, improving the model architecture is another crucial direction in achieving both high efficiency and superior performance. Numerous works [23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36] have studied the Neural Architecture Search (NAS) problem for representative model designs such as Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs). However, the realm of architecture search for LLMs remains unexplored. Though enjoying the potential benefits of discovering highly efficient and well-performing LLM architectures compared with manual designs, searching with traditional NAS methods for LLMs faces significant challenges due to the immense complexity and extensive model size. Furthermore, the convergence

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-4421

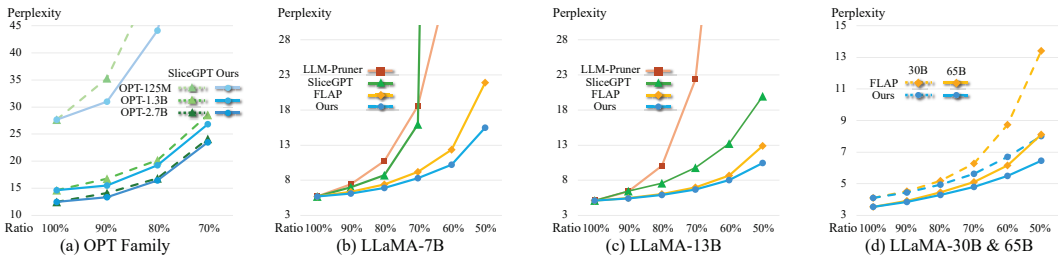

Figure 1: Experiment results of perplexity ↓ on WikiText2 dataset with 2048 sequence length.

of a randomly initialized architecture to the searched optimal state takes substantial training efforts and resources, intensifying the challenges of searching for efficient LLM architectures.

To tackle the challenges, we propose a training-free architecture search framework that discovers efficient LLM architectures within the original well-trained LLMs. Specifically, to reduce the search cost, we first identify an appropriate initial architecture by computing the importance of weights. Subsequently, an evolution-based algorithm is applied to globally search an efficient subnet starting from the initial subnet. In each generation, mutation and crossover are adopted to generate candidate architectures within the search space. The candidates are evaluated efficiently with a small amount of training samples to assess their effectiveness and select for the next generation. As we start our search from a well-trained LLM instead of randomly initialized models, we propose a mask mutation algorithm to identify the detailed channel indices, rather than just the number of channels in the mutation of traditional NAS [29, 24, 25, 26, 32]. After a few generations with the identified promising LLM architecture, we adopt the reformation algorithm based on the alternating direction method of multipliers (ADMM) [37, 38, 39, 40] to rectify weights in inherited efficient LLM architecture with omitted weights (i.e., non-inherited weights) by leveraging only 128 calibration samples.

As shown in Figure 1, our extensive experiments demonstrate that our method can achieve superior performance than SOTA structured pruning baselines in terms of perplexity and zero-shot accuracy on multiple datasets across various LLM families and model sizes. Particularly, as in Figure 1 (a), only SliceGPT and our method support the OPT model family, and our method outperforms SliceGPT. Additionally, with a 60% inheriting ratio for the LLaMA-7B model on the WikiText2 dataset, our method achieves the best performance with a perplexity of 10.21, compared to 38.27 by LLM-Pruner and 279.52 by SliceGPT, as illustrated in Figure 1 (b). Furthermore, when scaling to LLaMA-13B, both SliceGPT and LLM-Pruner fail, as in Figure 1 (c). Lastly, as in Figure 1 (d), only FLAP and our method support the LLaMA-30B and 65B models, and our method achieves better performance than FLAP. Besides, our implementations on GPUs demonstrate significant memory reduction and inference acceleration. Meanwhile, our approach eliminates the retraining process, relying solely on forward pass for both searching and reformation processes, which maintains a low memory overhead.

Our contributions are summarized below,

1. We propose a training-free search framework to identify subnets within LLMs, featuring an importance-aware initialization that significantly reduces the time cost of searching, and an evolution architecture search with special mask mutation and efficient candidate evaluation.

2. We propose a reformation algorithm that reconstructs weights by calibrating with only 128 training samples, thereby enhancing the effectiveness of the subnets.

3. Experiments indicate that the subnets generated by our method outperform SOTA structured pruning works in terms of perplexity and accuracy on multiple datasets across various LLM families and sizes. The searched subnets can effectively reduce GPU memory and accelerate inference.

2 Related Work

2.1 Compression of LLMs

Various compression techniques have been developed to reduce the model size or inference cost of LLMs, including model pruning, quantization, and distillation. Among these, quantization and

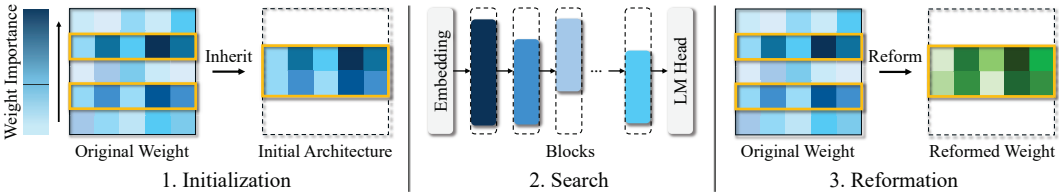

Figure 2: Framework Overview.

structured pruning methods are prevalent due to their efficacy in inference acceleration while preserving task performance. Quantization approaches, as explored in works [19, 18, 17], compress models by converting weights to lower bit representations. Besides, structured pruning techniques, including the works [8, 9, 10, 41], remove redundant weights in a structured manner to reduce the total weight count. Specifically, LLM-Pruner [8] eliminates non-critical coupled structures based on gradient information, while SliceGPT [9] substitutes each weight matrix with a smaller, dense matrix and reduces the embedding dimension of the network. FLAP [10] employs structured metrics to prune LLMs and globally optimizes the sparsity ratio with the output feature maps. Despite the advancements, most pruning methods indiscriminately remove heads within the self-attention modules, leading to more significant performance loss due to the inherent input-dependent nature of transformer architectures based on all heads.

2.2 Search for Transformers

NAS has emerged as a pivotal technique for identifying efficient architectures in CNNs (exemplified by EfficientNet [42]) and transformer-based models (such as BERT [43] and Vision Transformer [44]). To mitigate the typical high training costs associated with NAS, innovations such as one-shot and zero-shot NAS methods [26, 29, 25, 24] have been developed, enhancing the efficiency of generating high-performance architectures. In contrast to zero-shot NAS methods, which utilize accuracy predictors to derive optimal architectures, one-shot NAS methods streamline the process by pretraining a comprehensive supernet from which optimal subnets are subsequently selected. Specifically, in the context of transformer-based models, the one-shot NAS approach, as implemented in AutoFormer [29], involves multiple rounds of supernet training, strategically extending weights along certain dimensions to optimize performance. NASViT [26] leverages gradient information during supernet training to refine subnet selection and mitigate gradient conflicts, thereby enhancing the effectiveness of generated architectures. The proven efficacy of one-shot NAS for transformer architectures provides a compelling rationale for its application to LLMs, considering that pretrained LLMs can function analogously as supernets. This adaptation holds the potential to significantly advance the development and optimization of LLM architectures, motivating us to refine and enhance the capabilities of these complex models.

3 Methodology

3.1 Framework Overview

We show the overview of our search framework in Figure 2. It comprises three key components: search initialization, search pipeline, and weight reformation. First, an initial efficient architecture is constructed layer by layer with a uniform inheriting ratio based on the weight importance. Subsequently, based on the initialization, we conduct a comprehensive search process for the globally efficient architecture with the evolution-based search method. Finally, a reformation method is introduced to enhance the performance of the resulting subnets in LLMs without retraining.

3.2 Search Initialization

Global search with uniform initialization. Unlike prior efficient LLM research efforts such as SparseGPT [7] with a uniform sparsity ratio across all layers, our method leverages a global search approach, such that different layers in our searched architecture may inherit different percentages of parameters (inheriting ratios) from the original LLM. To reduce the search cost and promote the search performance, we initialize our search with the same inheriting ratio for all layers. Through our search process, we iteratively refine the architecture, yielding subnets with varied inheriting

https://doi.org/10.52202/079017-4421

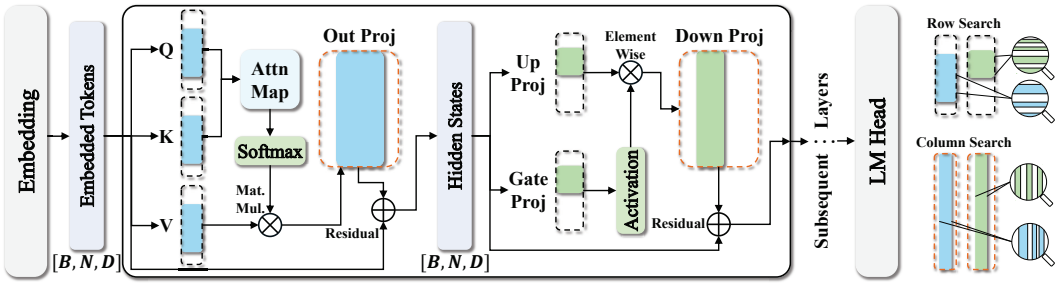

Figure 3: Visualization of the subnets generation for LLaMA family based on the selections masks $\mathbf{S}_{attn}$ for the self-attention module colored in blue and $\mathbf{S}_{mlp}$ for the MLP module colored in green.

ratios across layers. We demonstrate the pivotal role of the initialized architecture in driving search efficiency and effectiveness in Figure 5 and Section 3.3.3.

Structural subnets. To enable efficient inference, we search structural subnets from the original LLMs, i.e., certain rows or columns in the original 2D weight matrix are inherited in our searched model. Take the LLaMA family as an example. In each attention block of LLaMA models, there are query, key, value, and output linear layers in the self-attention module with weights denoted by $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$, and $\mathbf{W}_O$, respectively, and other three linear layers $\mathbf{W}_U$, $\mathbf{W}_G$, and $\mathbf{W}_D$ in the MLP module. To ensure the consistent hidden size in LLaMA models, based on the computation patterns in each block, we select rows in $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$, $\mathbf{W}_U$, and $\mathbf{W}_G$, and columns in $\mathbf{W}_O$ and $\mathbf{W}_D$. More details are presented in Figure 3 and Appendix A.

Initialization based on importance score. We construct the initial building blocks by inheriting appropriate rows/columns from the original LLM layer. To determine which row/column to be inherited, we compute the importance score for each row and column as below,

$$[\Phi^r_{\mathbf{W}}]_i = \sum_j [\mathbf{\Phi}]_{i,j}, \quad [\Phi^c_{\mathbf{W}}]_j = \sum_i [\mathbf{\Phi}]_{i,j}, \quad [\mathbf{\Phi}]_{i,j} = \frac{[\mathbf{W}]^2_{i,j}}{[(2\mathbf{X}\mathbf{X}^T)^{-1}]_{j,j}}, \tag{1}$$

where $[\Phi^r_{\mathbf{W}}]_i$ represents the row score for the i^{th} row of $\mathbf{W}$ and $[\Phi^c_{\mathbf{W}}]_j$ denotes the column score of the j^{th} column. $[\mathbf{\Phi}]_{i,j}$ is the importance value of the element in the i^{th} row and j^{th} column of $\mathbf{W}$, and $\mathbf{X}$ is the layer input. Following WoodFisher [45] and SparseGPT [7], the importance score reflects the minimum error of the layer-wise outputs (in terms of ℓ_2 norm) caused by removing a single weight. Note that the minimum error is evaluated by removing a single element from the weight matrix and it is not optimal in the case of simultaneously removing multiple weights.

Mask sharing. Given the column and row scores, we encode the architecture information by two masks: $\mathbf{S}_{attn} \in \mathbb{R}^M$ for the self-attention module and $\mathbf{S}_{mlp} \in \mathbb{R}^P$ for the MLP module for the layers in each building block. Different layers in the same module (self-attention or MLP) share the same mask to align the internal computations. We consider minimizing the collective importance scores for both the self-attention and MLP modules as below,

$$\min_{\mathbf{S}_{attn}} \|\mathbf{S}_{attn} \odot (\Phi^r_{\mathbf{W}_Q} + \Phi^r_{\mathbf{W}_K} + \Phi^r_{\mathbf{W}_V} + \Phi^c_{\mathbf{W}_O})\|, \tag{2}$$

$$\min_{\mathbf{S}_{mlp}} \|\mathbf{S}_{mlp} \odot (\Phi^r_{\mathbf{W}_U} + \Phi^r_{\mathbf{W}_G} + \Phi^c_{\mathbf{W}_D})\|, \tag{3}$$

where $\|\cdot\|$ denotes the ℓ_1 norm and $\odot$ means the element-vise multiplication. Given the target model size, we uniformly set the same inheriting ratio for the masks in all building blocks. To obtain the mask in each block, we perform sorting for the sum of the corresponding scores in Equation (2) and (3), and inherit/keep the subnets with larger scores as the initialized architecture with the target size following the inheriting ratio, while other rows/columns with smaller scores are omitted.

3.3 Architecture Search

In this section, we present our comprehensive training-free search framework with the visualization of the search process for one block of the LLaMA model shown in Figure 3. We first delineate the methodology for mutation based on the initialized selection masks. Next, we define the search space and present the search pipeline. Besides, we verify the effectiveness of our initialization strategy by comparing the convergence speeds with and without our initialization.

https://doi.org/10.52202/079017-4421

3.3.1 Mask Mutation

During the search, we use mask mutation to generate new masks and thus new subnets to explore the search space. The inheriting ratio for the selection mask $\mathbf{S}_{attn}$ is denoted as $\mathbf{\Gamma}_{attn} = \{\gamma^i_{attn}\}^h_{i=1}$ where h is the number of heads, and the inheriting ratio for $\mathbf{S}_{mlp}$ is γ_{mlp}. The mutation function $\mathcal{M}$ with the original mask $\mathbf{S}_{attn}$ or $\mathbf{S}_{mlp}$, mutation probability P_m, inheriting ratio requirement γ^i_{attn} or γ_{mlp}, similarity ratio α, and maximum iteration η can be represented as follows,

Algorithm 1: Mask Mutation

Input: $\mathbf{S}, P_m, \gamma, \alpha, \eta$
$P_r \leftarrow Random(0, 1)$
if $Inheriting\_Ratio(\mathbf{S}) == \gamma$ *and* $P_r > P_m$ **then**
  Output: $\mathbf{S}$
$N \leftarrow len(\mathbf{S}), iter \leftarrow 0$
$Idx_1 \leftarrow \{\mathbf{S} == 1\}, Idx_2 \leftarrow \phi$
while $len(Idx_1 \cap Idx_2) < \alpha * N$ *and* $iter < \eta$ **do**
  $Idx_2 \leftarrow Random\_Subset(\{0, 1, \cdots, N-1\}|\gamma)$
  $iter \leftarrow iter + 1$
end
$\mathbf{S}' = \mathbb{O}^N;\ \mathbf{S}'[Idx_2] \leftarrow 1$
Output: $\mathbf{S}'$ if $iter < \eta$ else $\mathbf{S}$

$$\mathbf{S}'_{attn} = \{\mathcal{M}(\mathbf{S}^i_{attn}, P_m, \gamma^i_{attn}, \alpha, \eta)\}^h_{i=1}, \tag{4}$$

$$\mathbf{S}'_{mlp} = \mathcal{M}(\mathbf{S}_{mlp}, P_m, \gamma_{mlp}, \alpha, \eta), \tag{5}$$

where $\mathbf{S}^i_{attn} \in \mathbb{R}^{h_m}$ denotes the selection mask for the i^{th} head and h_m is the head dimension. In details, we show the mask mutation process with Algorithm 1. If the inheriting ratio of input $\mathbf{S}$ already satisfies the requirement γ and the mutation is unnecessary based on the random generated P_r (i.e., $P_r > P_m$), we do not mutate and simply return $\mathbf{S}$. Otherwise, given $\mathbf{S}$ and thus the set of indices Idx_1 for the inherited rows or columns, we try to generate a new set of indices Idx_2 through random sampling between 0 to len($\mathbf{S}$) $-$ 1, such that (i) Idx_2 follows the required inheriting ratio requirement γ, and (2) the similarity of Idx_1 and Idx_2 (intersection set) is larger than threshold α.

3.3.2 Search Space

We define the LLM search space with three variables for each transformer building block below: the model depth d, inheriting ratios $\mathbf{\Gamma}_{attn} = \{\gamma^i_{attn}\}^h_{i=1}$ for $\mathbf{S}_{attn}$, and γ_{mlp} for $\mathbf{S}_{mlp}$. The specifications of this search space, including the range for each factor, are detailed in Table 1.

γ_{mlp} has a larger search space than $\{\gamma^i_{attn}\}^h_{i=1}$ according to our ablation study illustrated in Figure 4. Results are evaluated using LLaMA-7B on the WikiText2 dataset with a sequence length of 2048. Specifically, we apply the same local inheriting ratio for three cases, (i) the attention module only, (ii) the MLP module only, and (iii) both modules. Note that in case (i) or (ii), the global inheriting ratio is larger than case (iii) since the MLP in case (i) or the attention in case (ii) directly uses the original layers with 100% inheriting ratio. From Figure 4, we observe that case (ii) achieves a better perplexity with a lower global inheriting ratio than case (i), demonstrating that the MLP exhibits greater redundancy and is less sensitive to parameter reduction than the self-attention module. Therefore, we set a larger search space of inheriting ratios for MLP than the self-attention module.

Different from other transformer-based search works [29, 26, 46], we do not search the number of heads in self-attention. It stems from the nature of transformers that all heads are essential for representing the input data in the attention mechanism. Moreover, we refrain from conducting searches on the embedding and output layers of LLMs, as their weights constitute only a minor fraction of the total parameters yet are vital for the precise representation of tokens.

3.3.3 Search Pipeline

We implement our evolutionary search across the OPT and LLaMA model families with varying model sizes to derive efficient LLM architectures/subnets. The pipeline is shown below.

Table 1: Search space for different model sizes of OPT model family and LLaMA model family, where the notation $[a, b, c]$ specifies a range from a to b with an interval of c.

Model	OPT Family			LLaMA Family				General	
Space	Model Depth			Model Depth				Inheriting Ratio	
# Params.	125M	1.3B	2.7B	7B	13B	30B	65B	$\{\gamma^i_{attn}\}^h_{i=1}$	γ_{mlp}
90%	$[12, 12, 1]$	$[24, 24, 1]$	$[32, 32, 1]$	$[32, 32, 1]$	$[40, 40, 1]$	$[60, 60, 1]$	$[80, 80, 1]$	$[0.9, 1, 0.01]$	$[0.6, 1, 0.05]$
80%	$[12, 12, 1]$	$[24, 24, 1]$	$[30, 32, 1]$	$[32, 32, 1]$	$[40, 40, 1]$	$[60, 60, 1]$	$[80, 80, 1]$	$[0.8, 1, 0.01]$	$[0.4, 1, 0.05]$
70%	$[10, 12, 1]$	$[20, 24, 1]$	$[28, 32, 1]$	$[30, 32, 1]$	$[36, 40, 1]$	$[56, 60, 1]$	$[76, 80, 1]$	$[0.3, 1, 0.01]$	$[0.2, 1, 0.05]$
60%	$[10, 12, 1]$	$[20, 24, 1]$	$[28, 32, 1]$	$[28, 32, 1]$	$[32, 40, 1]$	$[52, 60, 1]$	$[72, 80, 1]$	$[0.6, 1, 0.01]$	$[0.1, 1, 0.05]$
50%	$[8, 12, 1]$	$[16, 24, 1]$	$[24, 32, 1]$	$[28, 32, 1]$	$[32, 40, 1]$	$[52, 60, 1]$	$[72, 80, 1]$	$[0.6, 1, 0.01]$	$[0.1, 1, 0.05]$

https://doi.org/10.52202/079017-4421

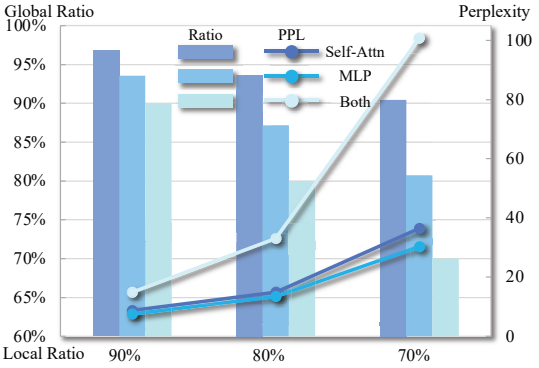

Figure 4: Ablation analysis of the inheriting ratios applied to the self-attention, MLP, or both.

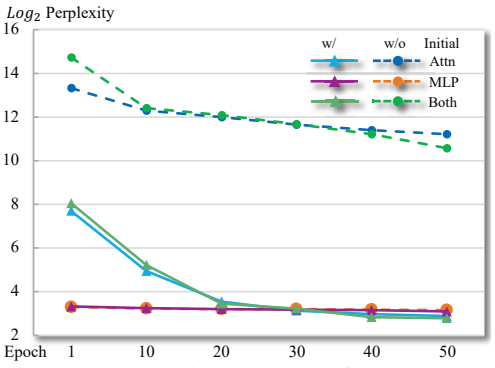

Figure 5: Ablation analysis of convergence speed with or without our initialization.

Initial generation. Given the single initialized subnet (Section 3.2), multiple candidates (N subnets in total) are generated by mutation of the inheriting ratios with probability P_s^0 and then mask mutation (Section 3.3.1) with probability P_m^0. The depth mutation is not involved at this initial step. The top k subnets are preserved as the initial generation.

Following generation. With the k subnets as parental candidates, a new population with N candidates are generated through mutation and crossover. We select a random parental candidate for mutation until the number of mutation candidates reaches a threshold N_m. Mutation involves altering the depth with probability P_d, altering the inheriting ratios with probability $P_s < P_s^0$, and mutating the mask with probability $P_m < P_m^0$ (see Algorithm 1). The probabilities are smaller than initial generation as superior candidates should be preserved with less randomness. For the crossover, two parental candidates are randomly selected and combined to form a new candidate until there are N_c candidates. With the population from the parental candidates, top k subnets are preserved as the next generation.

Candidate evaluation. For each generated candidate, if its parameter number does not fall in the range of the target model size, it is unsatisfying and we simply drop it. To compare candidate subnets, we evaluate them with a few random training samples from WikiText2 to compute the perplexity.

Necessity for initialization. To verify the effectiveness of our initialization in the search, we ablate the initialization for three cases, (i) self-attention only, (ii) MLP only, and (iii) both modules. LLaMA-7B is adopted with an 80% inheriting ratio for selection masks. Results are evaluated on Wikitext2 with a 2048 sequence length. As shown in Figure 5, the self-attention module complicates the identification of an effective subnet without our initialization strategy. In contrast, the MLP module exhibits less sensitivity to initialization. The search in both modules struggles to yield effective subnets without our initialization, primarily due to self-attention. The observation underscores the necessity of our initialization approach.

3.4 Reformation

After the search, we can obtain a subnet from the original LLM. To improve the subnet performance, we further reform the weights in the subnet by using the omitted weights to compensate their loss. Specifically, for each linear layer in the subnet with their original weights $\mathbf{W}$ before the search, we would like to reform the weights under the searched mask $\mathbf{M}$ and obtain $\widehat{\mathbf{W}}$, so that the layer output difference in ℓ_2 norm, i.e., $\|\widehat{\mathbf{W}}\mathbf{X} - \mathbf{W}\mathbf{X}\|_2^2$, is minimized. The problem is formulated below,

$$\begin{aligned} \min_{\widehat{\mathbf{W}}} \ & \|\widehat{\mathbf{W}}\mathbf{X} - \mathbf{W}\mathbf{X}\|_2^2, \\ \text{s.t.} \ & \widehat{\mathbf{W}} \odot \mathbf{M} = \mathbf{0}, \end{aligned} \tag{6}$$

where $\mathbf{M}$ indicates the location of pruned weights with element 1 denoting pruned and 0 denoting unpruned weights. Here we only reform inherited columns based on omitted columns in $\mathbf{W}$ rather than reforming rows with omitted rows, since the output corresponding to omitted rows are always zeros which are unavailable for any compensations by modifications in other rows. To solve this problem, we propose a solution based on alternating direction method of multipliers (ADMM) [38, 37, 47] with the following theorem. The detailed proof is shown in Appendix B.

Table 2: Results of the compressed LLaMA-7B and LLaMA-13B on the WikiText2 dataset, PTB dataset, and other common sense reasoning datasets. The perplexity on the WikiText2 and PTB is calculated with the 2048 sequence length. The accuracy results are evaluated with the same pipeline as LLM-Pruner [8] to ensure a fair comparison. The average is computed across seven classification datasets. LLM-Pruner (v), (e2). and (e1) denote the vector-wise and element-wise importance, (c) and (b) denote the channel and block strategies.

Method	Inheriting Ratio	Wiki PPL↓	PTB PPL↓	BoolQ	PIQA	Hella Swag	Wino Grande	ARC-e	ARC-c	OBQA	Average Acc.↑
LLaMA-7B	100%	5.68	27.34	73.18	78.35	72.99	67.01	67.45	41.38	42.40	63.25
LLM-Pruner(v)	90%	7.73	38.94	67.95	77.42	69.31	63.54	66.33	39.85	41.20	60.80
LLM-Pruner(e2)		7.46	36.87	68.29	76.88	70.25	64.33	65.28	40.10	39.60	60.68
LLM-Pruner(e1)		7.42	36.73	66.97	77.26	70.30	64.33	65.24	40.19	41.00	60.76
SliceGPT	90%	7.00	133.80	57.68	69.80	59.32	68.11	62.75	36.01	38.00	55.95
FLAP	90%	6.34	32.39	74.43	75.41	68.68	67.01	65.78	38.48	41.00	61.54
Ours	90%	**6.10**	**32.05**	74.37	76.88	70.71	67.56	68.39	40.10	39.20	**62.46**
LLM-Pruner(v)	80%	10.73	59.73	61.44	71.71	57.27	54.22	55.77	33.96	38.40	53.25
LLM-Pruner(e2)		11.97	55.68	59.39	75.57	65.34	61.33	59.18	37.12	39.80	56.82
LLM-Pruner(e1)		10.73	59.73	57.06	75.68	66.80	59.83	60.94	36.52	40.00	56.69
SliceGPT	80%	8.71	143.89	37.89	64.09	45.67	62.75	53.62	31.74	33.20	46.99
FLAP	80%	7.40	36.77	68.59	74.21	64.98	64.40	59.89	37.80	40.20	58.58
Ours	80%	**6.89**	**36.06**	70.98	74.92	67.29	64.64	64.23	36.52	39.40	**59.71**
LLaMA-13B	100%	5.09	19.23	68.47	78.89	76.24	70.09	74.58	44.54	42.00	64.97
LLM-Pruner(c)	90%	7.70	35.32	68.47	74.76	66.99	66.38	66.58	35.24	38.20	59.52
LLM-Pruner(b)		6.38	31.85	70.64	78.40	75.00	69.46	*72.82*	41.47	41.40	64.17
SliceGPT	90%	6.43	86.09	61.74	69.97	60.74	69.38	66.79	40.70	41.80	58.73
FLAP	90%	5.45	20.98	63.76	78.07	73.69	69.61	69.53	39.93	41.60	62.31
Ours	90%	**5.39**	**20.63**	71.65	78.18	75.04	69.61	69.70	43.09	42.60	**64.23**
LLM-Pruner(c)	80%	48.76	218.49	62.39	66.87	49.17	58.96	49.62	31.83	33.20	50.29
LLM-Pruner(b)		10.05	55.46	67.68	77.15	73.41	65.11	68.35	38.40	42.40	61.79
SliceGPT	80%	7.55	117.94	50.34	66.00	53.37	68.11	60.56	36.35	38.20	53.27
FLAP	80%	6.03	23.33	62.23	76.50	70.59	68.35	65.66	38.99	41.60	60.56
Ours	80%	**5.90**	**22.66**	68.53	77.09	72.60	69.22	66.25	40.02	41.00	**62.10**

Theorem 3.1. *Problem (6) can be solved in iterations. In the k^{th} iteration, it performs the updates:*

$$\widehat{\mathbf{W}}^{k+1} = (\mathbf{X}\mathbf{X}^T + \rho\mathbf{I})^{-1}(\mathbf{X}\mathbf{X}^T\mathbf{W}^T + \rho(\mathbf{Z}^k - \mathbf{U}^k)^T), \tag{7}$$

$$\mathbf{Z}^{k+1} = \left(\widehat{\mathbf{W}}^{k+1} + \mathbf{U}^k\right) \odot \mathbf{M}, \tag{8}$$

$$\mathbf{U}^{k+1} = \mathbf{U}^k + \mathbf{W}^{k+1} - \mathbf{Z}^{k+1}, \tag{9}$$

where $\rho > 0$ is the penalty parameter. The initial variable values at $k = 0$ follow the configurations that $\widehat{\mathbf{W}}^0 = \mathbf{W}$, $\mathbf{Z}^0 = \widehat{\mathbf{W}}^0$, and $\mathbf{U}^0 = \mathbf{0}$.

In practice, we find that after a few tens iterations (such as 20 or 30), the loss in Problem (6) converges. The complexity is determined by the inversion operation, which is the same as SparseGPT [7]. However, as it can converge fast within 30 iterations, our ADMM-based solution typically requires less computation time than SparseGPT which needs to iterate over all rows in the weight matrix.

3.5 Efficient Inference

After searching and reformation, we can get optimal efficient subnets with the selection masks $\mathbf{S}_{attn} \in \mathbb{R}^M$ and $\mathbf{S}_{mlp} \in \mathbb{R}^P$ for each block of the LLMs. We further convert the subnets into small-dense models following the masks for efficient inference. Thus, the dimension of the weight is actually reduced with faster inference speed. More details can be found in Appendix A.

4 Experiments

4.1 Experiment Settings

Hyper-parameter setting. For the evolutionary search, we adopt specific hyper-parameters as follows: the population size (N), the number of mutations (N_m), and the number of crossovers (N_c)

https://doi.org/10.52202/079017-4421

Table 3: Results of compressed Vicuna-7B.

Method	Inheriting Ratio	Wiki PPL↓	PTB PPL↓	BoolQ	PIQA	Hella Swag	Wino Grande	ARC-e	ARC-c	OBQA	Average Acc.↑
Vicuna-7B	100%	6.78	26.78	76.57	77.75	70.64	67.40	65.11	41.21	40.80	62.78
LLM-Pruner(e2)	90%	8.74	30.69	61.68	75.95	69.35	64.72	68.86	39.16	40.00	59.96
SliceGPT		8.23	61.83	67.00	69.64	58.65	63.77	54.59	37.71	38.40	55.68
FLAP		7.64	29.95	74.43	75.41	68.68	67.01	65.78	38.48	41.00	61.54
Ours		**7.18**	**28.87**	75.10	75.90	69.97	66.92	67.11	40.61	40.40	**62.29**
LLM-Pruner(e2)	80%	12.97	46.34	62.87	75.41	64.00	58.41	60.98	37.12	39.00	56.83
SliceGPT		10.13	94.93	53.82	64.42	49.14	60.38	52.27	34.56	33.40	49.71
FLAP		8.90	34.04	69.14	72.52	63.01	64.48	61.11	34.56	39.00	57.69
Ours		**8.20**	**33.59**	66.30	74.92	65.05	62.90	64.63	38.91	39.80	**58.93**

Table 4: Results of compressed OPT.

Ratio	90%		80%		70%	
Method	Wiki ↓	PTB ↓	Wiki ↓	PTB ↓	Wiki ↓	PTB ↓
OPT-125M Wiki: 27.65 PTB: 38.99						
SliceGPT	35.31	75.59	54.88	149.17	84.16	245.18
Ours	**30.97**	**40.14**	**44.12**	**66.55**	**80.84**	**124.27**
OPT-1.3B Wiki: 14.63 PTB: 20.29						
SliceGPT	16.74	35.31	20.17	61.30	28.53	113.42
Ours	**15.51**	**20.19**	**19.23**	**32.81**	**26.82**	**69.42**
OPT-2.7B Wiki: 12.47 PTB: 17.97						
SliceGPT	14.10	37.01	16.81	65.09	24.12	132.13
Ours	**13.32**	**17.24**	**16.44**	**23.66**	**23.48**	**58.46**

Table 5: Results with lower inheriting ratio.

Ratio	70%		60%		50%	
Method	Wiki ↓	PTB ↓	Wiki ↓	PTB ↓	Wiki ↓	PTB ↓
LLaMA-7B						
LLM-Pruner(e2)	18.58	93.24	38.27	238.09	125.96	460.73
SliceGPT	15.95	583.58	279.52	5186.06	1830.43	15333.66
FLAP	9.18	47.35	12.34	65.54	21.89	135.84
Ours	**8.28**	**45.26**	**10.21**	**62.07**	**15.48**	**117.06**
LLaMA-13B						
LLM-Pruner(e2)	22.36	112.03	66.38	278.36	3827.63	1287.11
SliceGPT	9.79	167.27	13.21	247.71	19.95	408.68
FLAP	6.97	27.38	8.67	35.91	12.88	53.54
Ours	**6.67**	**26.37**	**8.00**	**33.23**	**10.44**	**45.73**

are set to 100, 50, and 30, respectively. In each generation, the top 10 subnets are selected as parental candidates to produce offspring networks through the mechanisms of mutation and crossover. The rest subnets in the population are generated with mutation with larger randomness (i.e., same as initial mutation). The initial mutation probabilities (P_m^0 and P_s^0) are set at 0.6 and 0.3 to promote variability early in the search process. Subsequently, for ongoing generation processes, the mutation probabilities (P_m and P_s) are adjusted to 0.3 and 0.1, while the probability for depth (P_d) is maintained at 0.1. The similarity ratio α and maximum iteration η are set at 0.8 and 1000 in mask mutation. The total evolution epoch is 50. For the reformation, we adopt ρ as 1.0 and the iteration number as 30.

Datasets and metrics. We compare the perplexity of the models on the WikiText2 [48] and PTB [49] datasets with the 2048 sequence length. We also compare the zero-shot accuracy on common reasoning zero-shot classification datasets including BoolQ [50], PIQA [51], HellaSwag [52], WinoGrande [53], ARC-easy [54], ARC-challenge [54], and OpenbookQA [55].

Models. We evaluate on multiple LLM families including LLaMA [1], Vicuna [56] and OPT [2].

Baselines and pipeline. We compare with SOTA baselines including LLM-pruner [8], SliceGPT [9] and FLAP [10]. We adhere to the exact evaluation pipeline from the well-known LLM-Pruner [8], which is applied for all approaches to ensure a fair comparison. For the reformation, we randomly select 128 samples from training split of WikiText2, with the same random seed and thus same data for the calibration of other works, including SliceGPT and FLAP, ensuring a fair comparison.

Search Cost. We leverage the evolution search on NVIDIA A100 40G GPUs. Specifically, to explore the subnets of LLaMA-7B, we finish the search on one GPU with around 5 hours.

4.2 Main Results

Superior performance compared with SOTA baselines. We show our results of LLaMA-7B and LLaMA-13B in Table 2 and Figure 1 (b) and (c). We observe that our method outperforms all baselines in terms of perplexity on WikiText2 and PTB, and average zero-shot accuracy (over multiple zero-shot datasets). Take LLaMA-13B on WikiText2 as an example, our method improves the perplexity by 4.15, 1.65, and 0.13 compared to LLM-Pruner(b), SliceGPT, and FLAP, respectively, with a 80% inheriting ratio. Meanwhile, it achieves a higher average accuracy on seven classification datasets than baselines. For instance, under a 80% inheriting ratio, our method on LLaMA-7B improves the average accuracy by 2.89%, 12.72%, and 1.13% compared to LLM-Pruner(e1), SliceGPT, and FLAP, respectively. As the SliceGPT is sensitive to the calibration dataset, we further show the results with PTB calibration in Appendix C. Besides, we ablate the search with PTB in Appendix D.

Table 6: Results of extra large LLaMA models.

Model		LLaMA-30B						LLaMA-65B					
Dataset	Ratio	100%	90%	80%	70%	60%	50%	100%	90%	80%	70%	60%	50%
Wiki↓	FLAP	4.10	4.52	5.18	6.28	8.73	13.41	3.53	3.91	4.45	5.10	6.16	8.11
	Ours		**4.44**	**4.94**	**5.63**	**6.70**	**8.01**		**3.84**	**4.29**	**4.80**	**5.49**	**6.45**
PTB↓	FLAP	16.29	17.29	19.30	21.88	29.11	47.30	17.61	19.35	21.01	22.45	25.97	33.86
	Ours		**17.15**	**18.80**	**20.70**	**24.22**	**30.51**		**19.22**	**20.40**	**21.39**	**23.41**	**30.08**

The comparisons on other LLM families are demonstrated in Table 3 for Vicuna-7B [56] and Table 4 with Figure 1 (a) for OPT family [2]. As LLM-Pruner and FLAP do not implement their methods on OPT, we only compare with SliceGPT for OPT models. We can make similar observations that our method performs the best compared with SOTA baselines, demonstrating a superior generalization performance across various datasets/tasks, model structures, and inheriting ratios.

Scaling to small inheriting ratios and large models. Furthermore, our method consistently performs the best when scaling to larger model sizes or smaller inheriting ratios, indicating the great potential of our method for the ever-increasing LLM model size. Specifically, we show the results of LLaMA-7B and LLaMA-13B with lower inheriting ratios and thus more efficient models in Table 5 and Figure 1 (b) and (c). Our method consistently performs the best with more significant improvements under smaller inheriting ratios such as 50%. Besides, we deliver results on large models including LLaMA-30B and LLaMA-65B in Table 6 and Figure 1 (d). Our method achieves superior performance than FLAP under various settings, verifying our effectiveness and generalization.

4.3 Ablation Study

We show the results of LLaMA-7B with 128 sequence length in Table 7. Our method performs the best across different inheriting ratios, indicating the effectiveness on short sequences. Results of LLaMA-13B are in Appendix E.

Table 7: LLaMA-7B perplexity (↓) results on WikiText2 dataset with 128 sequence length.

Ratio	LLM-Pruner(e1)	SliceGPT	FLAP	Ours
90%	15.22	14.18	14.15	**13.40**
80%	19.09	17.08	14.62	**14.54**
70%	30.63	24.39	17.62	**17.11**
60%	52.30	40.04	23.53	**20.24**
50%	106.07	74.09	31.80	**26.96**

Besides, to verify the influence of the number of examples for the reformation, we conduct ablation studies by varying the number from 128 to 512 and 1024. As shown in Figure 6, our reformation effectively improves the performance, and it is not sensitive to the number of samples. 128 samples already provide satisfying performance. Furthermore, we investigate the impact of the step number and ρ in the ADMM solution for the reformation, with detailed ablation results presented in Appendix F.

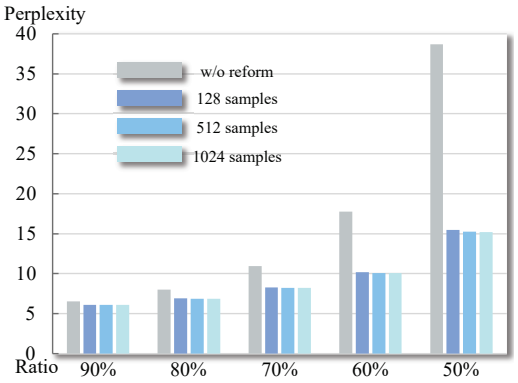

Figure 6: Ablation analysis for reformation with different numbers of samples.

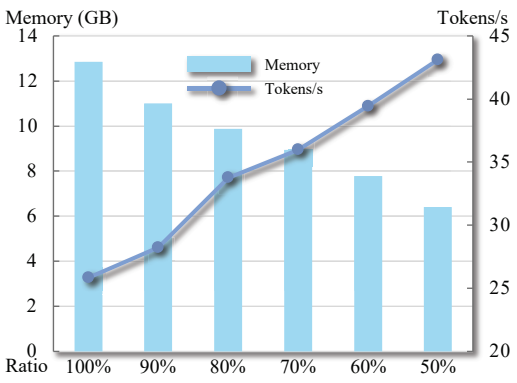

Figure 7: Analysis for memory and generation speed of LLaMA-7B on NVIDIA A100 40G.

4.4 Generation Acceleration

To demonstrate our acceleration performance, we report the memory consumption and inference speed with our searched LLaMA-7B models on NVIDIA A100 40G GPUs across different inheriting ratios. As shown in Figure 7, we can observe that with a smaller inheriting ratio, our searched efficient model consumes less memory with a faster generation speed.

https://doi.org/10.52202/079017-4421

5 Conclusion and Limitation

In this paper, we propose a training-free search framework to find the optimal subnets inside LLMs. We further propose a reformation algorithm that reconstructs the weights of subnets to enhance the task performance. The experiments show the effectiveness of our proposed method compared to SOTA structured pruning methods. Additionally, we achieve memory reduction and practical inference acceleration on GPUs, which shows the efficiency of our method. The search cost required by our method can increase with the model size, which takes more time for large models.

Acknowledgment

We would like to express our sincere gratitude to Professor Lin and Professor Wang for their invaluable guidance throughout the development of this paper. We are also deeply grateful to Pu, Yifan, and Zhenglun for their dedicated contributions, thoughtful insights, and collaborative efforts, which were essential to the completion of this research. Meanwhile, This research is funded in whole or in part by the National Science Foundation CNS-2312158 to Northeastern University. Any errors and opinions are not those of the National Science Foundation and are attributable solely to the author(s).

References

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

[2] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv*, 2022.

[3] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv*, 2022.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 33:1877–1901, 2020.

[5] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020.

[7] Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*, 2023.

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

[9] Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*, 2024.

[10] Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models, 2023.

[11] Zheng Zhan, Zhenglun Kong, Yifan Gong, Yushu Wu, Zichong Meng, Hangyu Zheng, Xuan Shen, Stratis Ioannidis, Wei Niu, Pu Zhao, and Yanzhi Wang. Exploring token pruning in vision state space models. *arXiv preprint arXiv:2409.18962*, 2024.

[12] Xuan Shen, Zhenglun Kong, Changdi Yang, Zhaoyang Han, Lei Lu, Peiyan Dong, et al. Edgeqat: Entropy and distribution guided quantization-aware training for the acceleration of lightweight llms on the edge. *arXiv preprint arXiv:2402.10787*, 2024.

[13] Changdi Yang, Pu Zhao, Yanyu Li, Wei Niu, Jiexiong Guan, Hao Tang, Minghai Qin, Bin Ren, Xue Lin, and Yanzhi Wang. Pruning parameterization with bi-level optimization for efficient semantic segmentation on the edge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15402–15412, 2023.

[14] Yihua Zhang, Yuguang Yao, Parikshit Ram, Pu Zhao, Tianlong Chen, Mingyi Hong, Yanzhi Wang, and Sijia Liu. Advancing model pruning via bi-level optimization. *Advances in Neural Information Processing Systems*, 35:18309–18326, 2022.

[15] Yanyu Li, Changdi Yang, Pu Zhao, Geng Yuan, Wei Niu, Jiexiong Guan, Hao Tang, Minghai Qin, Qing Jin, Bin Ren, Xue Lin, and Yanzhi Wang. Towards real-time segmentation on the edge. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023.

[16] Pu Zhao, Fei Sun, Xuan Shen, Pinrui Yu, Zhenglun Kong, Yanzhi Wang, and Xue Lin. Pruning foundation models for high accuracy without retraining. *arXiv preprint arXiv:2410.15567*, 2024.

[17] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training compression for generative pretrained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

[18] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[19] Xuan Shen, Peiyan Dong, Lei Lu, Zhenglun Kong, Zhengang Li, Ming Lin, Chao Wu, and Yanzhi Wang. Agile-quant: Activation-guided quantization for faster inference of llms on the edge. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):18944–18951, Mar. 2024.

[20] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*, 2019.

[21] Siqi Sun, Zhe Gan, Yuwei Fang, Yu Cheng, Shuohang Wang, and Jingjing Liu. Contrastive distillation on intermediate representations for language model compression. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 498–508, Online, November 2020. Association for Computational Linguistics.

[22] Haojie Pan, Chengyu Wang, Minghui Qiu, Yichang Zhang, Yaliang Li, and Jun Huang. Meta-KD: A meta knowledge distillation framework for language model compression across domains. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3026–3036, Online, August 2021. Association for Computational Linguistics.

[23] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019.

[24] Xuan Shen, Yaohua Wang, Ming Lin, Yilun Huang, Hao Tang, Xiuyu Sun, and Yanzhi Wang. Deepmad: Mathematical architecture design for deep convolutional neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6163–6173, June 2023.

[25] Ming Lin, Pichao Wang, Zhenhong Sun, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. Zen-nas: A zero-shot nas for high-performance deep image recognition. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021*, 2021.

 https://doi.org/10.52202/079017-4421

[26] Chengyue Gong, Dilin Wang, Meng Li, Xinlei Chen, Zhicheng Yan, Yuandong Tian, qiang liu, and Vikas Chandra. NASVit: Neural architecture search for efficient vision transformers with gradient conflict aware supernet training. In *International Conference on Learning Representations*, 2022.

[27] Xiu Su, Shan You, Jiyang Xie, Mingkai Zheng, Fei Wang, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. Vision transformer architecture search. *arXiv preprint arXiv:2106.13700*, 2021.

[28] Dongning Ma, Pengfei Zhao, and Xun Jiao. Perfhd: Efficient vit architecture performance ranking using hyperdimensional computing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2230–2237, June 2023.

[29] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12270–12280, October 2021.

[30] Peiyan Dong, Zhenglun Kong, Xin Meng, Pinrui Yu, Yifan Gong, Geng Yuan, Hao Tang, and Yanzhi Wang. Hotbev: Hardware-oriented transformer-based multi-view 3d detector for bev perception. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 2824–2836. Curran Associates, Inc., 2023.

[31] Zheng Zhan, Yifan Gong, Pu Zhao, Geng Yuan, Wei Niu, Yushu Wu, Tianyun Zhang, Malith Jayaweera, David Kaeli, Bin Ren, et al. Achieving on-mobile real-time super-resolution with neural architecture and pruning search. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4821–4831, 2021.

[32] Yushu Wu, Yifan Gong, Pu Zhao, Yanyu Li, Zheng Zhan, Wei Niu, Hao Tang, Minghai Qin, Bin Ren, and Yanzhi Wang. Compiler-aware neural architecture search for on-mobile real-time super-resolution. In *European Conference on Computer Vision*, pages 92–111. Springer, 2022.

[33] Changdi Yang, Yi Sheng, Peiyan Dong, Zhenglun Kong, Yanyu Li, Pinrui Yu, Lei Yang, Xue Lin, and Yanzhi Wang. Fast and fair medical ai on the edge through neural architecture search for hybrid vision models. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 01–09. IEEE, 2023.

[34] Changdi Yang, Yi Sheng, Peiyan Dong, Zhenglun Kong, Yanyu Li, Pinrui Yu, Lei Yang, and Xue Lin. Late breaking results: Fast fair medical applications? hybrid vision models achieve the fairness on the edge. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–2. IEEE, 2023.

[35] Peiyan Dong, Zhenglun Kong, Xin Meng, Pinrui Yu, Yifan Gong, Geng Yuan, Hao Tang, and Yanzhi Wang. Hotbev: Hardware-oriented transformer-based multi-view 3d detector for bev perception. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 2824–2836. Curran Associates, Inc., 2023.

[36] Yanyu Li, Pu Zhao, Geng Yuan, Xue Lin, Yanzhi Wang, and Xin Chen. Pruning-as-search: Efficient neural architecture search via channel pruning and structural reparameterization. *arXiv preprint arXiv:2206.01198*, 2022.

[37] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and trends® in Optimization*, 1(3):127–239, 2014.

[38] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[39] Pu Zhao, Sijia Liu, Yanzhi Wang, and Xue Lin. An admm-based universal framework for adversarial attacks on deep neural networks. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, page 1065–1073, New York, NY, USA, 2018. Association for Computing Machinery.

[40] Yifan Gong, Zheng Zhan, Zhengang Li, Wei Niu, Xiaolong Ma, Wenhao Wang, Bin Ren, Caiwen Ding, Xue Lin, Xiaolin Xu, et al. A privacy-preserving-oriented dnn pruning and mobile acceleration framework. In *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, pages 119–124, 2020.

[41] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

[42] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[43] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[44] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[45] Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems*, 33:18098–18109, 2020.

[46] Yi-Lun Liao, Sertac Karaman, and Vivienne Sze. Searching for efficient multi-stage vision transformers. *arXiv preprint arXiv:2109.00642*, 2021.

[47] Tom Goldstein, Brendan O'Donoghue, Simon Setzer, and Richard Baraniuk. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences*, 7(3):1588–1623, 2014.

[48] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv*, 2016.

[49] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, pages 313–330.

[50] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

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

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

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

[54] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[55] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[56] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

[57] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

 https://doi.org/10.52202/079017-4421

Appendix

A Efficient Inference

After searching and reformation, we can get optimal efficient subnets with the selections masks $\mathbf{S}_{attn} \in \mathbb{R}^M$ and $\mathbf{S}_{mlp} \in \mathbb{R}^P$ for each block of the LLMs. In detail, for the weights of query, key, and value denoted as $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{M\times D}$, we generate the weight subsets by extracting the selected rows from the original weights, which are denoted as $\mathbf{W}'_Q, \mathbf{W}'_K, \mathbf{W}'_V \in \mathbb{R}^{m\times D}$. For the weights of the output projection $\mathbf{W}_O \in \mathbb{R}^{D\times M}$, we extract the columns instead of rows and reform the selected weight based on the omitted ones for the weight subnets $\mathbf{W}'_O \in \mathbb{R}^{D\times m}$. Subsequently, the given input $\mathbf{X} \in \mathbb{R}^{BN\times D}$ is projected in the self-attention module as follows,

$$\mathbf{X}' = \mathbf{W}'_O\{softmax[(\mathbf{W}'_Q\mathbf{X}) \cdot (\mathbf{W}'_K\mathbf{X})^T] \cdot (\mathbf{W}'_V\mathbf{X})\} \in \mathbb{R}^{BN\times D} \tag{10}$$

For the MLP module in the LLaMA family, we denote the weights of the three linear layers with $\mathbf{W}_U, \mathbf{W}_G \in \mathbb{R}^{P\times D}$, and $\mathbf{W}_D \in \mathbb{R}^{D\times P}$ for the up, gate, and down projections, respectively. The weight subsets generated with the selection mask $\mathbf{S}_{mlp}$ for three linear layers are $\mathbf{W}'_U, \mathbf{W}'_G \in \mathbb{R}^{p\times D}$, and $\mathbf{W}'_D \in \mathbb{R}^{D\times p}$, where only $\mathbf{W}_D$ is reformed. Then, the given input $\mathbf{X} \in \mathbb{R}^{BN\times D}$ is projected in the MLP module as follows,

$$\mathbf{X}' = \mathbf{W}'_D\{(\mathbf{W}'_U\mathbf{X}) \odot activation[(\mathbf{W}'_G\mathbf{X})]\} \in \mathbb{R}^{BN\times D} \tag{11}$$

where the activation function for the LLaMA family is SiLU [57].

Therefore, the computation cost is reduced for both self-attention and MLP modules, while the configuration of the input $\mathbf{X} \in \mathbb{R}^{BN\times D}$ is preserved for preventing information loss and maintaining the representation capability of tokens.

B Proof of Theorem 3.1

Problem (6) can be reformulated as follows,

$$\begin{aligned} &\min_{\widehat{\mathbf{W}},\mathbf{Z}} \ \|\widehat{\mathbf{W}}\mathbf{X} - \mathbf{W}\mathbf{X}\|_2^2 + g(\mathbf{Z}), \\ &\text{s.t. } \widehat{\mathbf{W}} = \mathbf{Z}, \end{aligned} \tag{12}$$

where $g(\mathbf{Z})$ is a inidicator function as the following,

$$g(\mathbf{Z}) = \begin{cases} \infty, & \text{otherwise}, \\ 0, & \text{if } \ \widehat{\mathbf{W}} \odot \mathbf{M} = \mathbf{0}. \end{cases} \tag{13}$$

We can see that Problem (12) is equvilant to Problem (6).

Based on ADMM [37, 38, 47], Problem (12) can be solved with ADMM iterations. In the k^{th} iteration, it needs to address the following,

$$\widehat{\mathbf{W}}^{k+1} = \arg\min_{\widehat{\mathbf{W}}} \quad \|\widehat{\mathbf{W}}\mathbf{X} - \mathbf{W}\mathbf{X}\|_2^2 + \frac{\rho}{2}\|\widehat{\mathbf{W}} - \mathbf{Z}^k + \mathbf{U}^k\|_2^2, \tag{14}$$

$$\mathbf{Z}^{k+1} = \arg\min_{\mathbf{Z}} \quad g(\mathbf{Z}) + \frac{\rho}{2}\|\widehat{\mathbf{W}}^{k+1} - \mathbf{Z} + \mathbf{U}^k\|_2^2, \tag{15}$$

$$\mathbf{U}^{k+1} = \mathbf{U}^k + \mathbf{W}^{k+1} - \mathbf{Z}^{k+1}, \tag{16}$$

Problem (12) is split into multiple sub-problems with Problem (14) and (15).

Problem (14) is similar to Ridge regression problem. We can directly obtain its solution as

$$\widehat{\mathbf{W}}^{k+1} = (\mathbf{X}\mathbf{X}^T + \rho\mathbf{I})^{-1}(\mathbf{X}\mathbf{X}^T\mathbf{W}^T + \rho(\mathbf{Z}^k - \mathbf{U}^k)^T), \tag{17}$$

To solve Problem (15), we can set $\mathbf{Z}^{k+1} = \left(\widehat{\mathbf{W}}^{k+1} + \mathbf{U}^k\right)$ and project $\mathbf{Z}^{k+1}$ on the g function as follows,

$$\mathbf{Z}^{k+1} = \left(\widehat{\mathbf{W}}^{k+1} + \mathbf{U}^k\right) \odot \mathbf{M}, \tag{18}$$

Thus, we can obtain the solution in Theorem 3.1.

Table A1: Compare with SliceGPT using LLaMA-7B perplexity (↓) results on PTB dataset.

Calibration Dataset	Method	90%	80%	70%	60%	50%
PTB	SliceGPT	38.47	43.23	51.38	63.14	118.78
WikiText2	SliceGPT	133.80	143.89	583.58	51.86.06	15333.66
WikiText2	Ours	**32.05**	**36.06**	**45.26**	**62.07**	**117.06**

Table A2: LLaMA-7B perplexity (↓) results on different datasets of search and evaluation.

Dataset		Inheriting Ratio				
Search	Eval	90%	80%	70%	60%	50%
Wiki	Wiki	6.10	6.89	8.28	10.21	15.48
Wiki	PTB	32.05	36.06	45.26	62.07	117.06
PTB	Wiki	6.22	7.08	8.41	11.08	17.23
PTB	PTB	31.24	34.87	43.89	59.07	108.83

C SliceGPT Comparison

For SliceGPT [9], the generated results are sensitive to the calibration datasets, we further show the results of SliceGPT with the calibration of PTB training dataset and 2048 sequence length in Table A1. We can know that our method can achieve better performance with the calibration on WikiText2 instead of PTB dataset than the SliceGPT with the calibration on the same dataset.

D Search on PTB Dataset

To further verify the generalization and effectiveness of our method, we utilize the training portion of the PTB dataset in our search instead of WikiText2. The results, shown in Table A2, are evaluated with a sequence length of 2048 using the LLaMA-7B model. Our findings reveal that the models generated through searches on two different training datasets achieve similar performance, demonstrating the robustness and generalization capability of our search method across different datasets.

E Ablation for 128 Sequence Length

We also present the results for the LLaMA-13B model with a sequence length of 128 in Table A3, demonstrating that our method continues to achieve superior performance. The results are evaluated with on WikiText2 dataset.

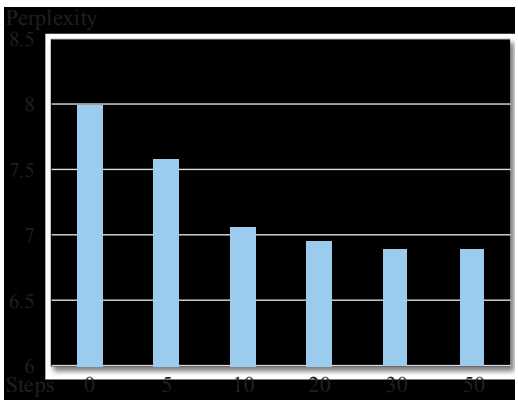

Figure A1: Ablation analysis for reformation with different numbers of steps.

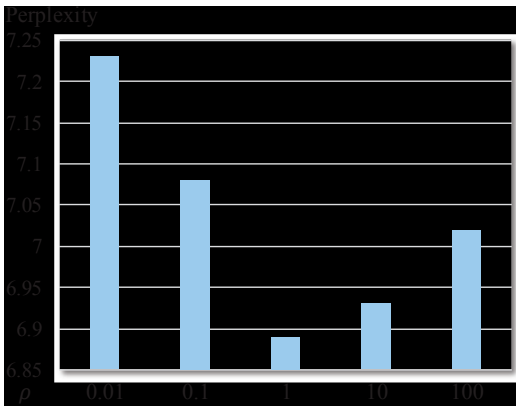

Figure A2: Ablation analysis for reformation with different ρ.

https://doi.org/10.52202/079017-4421

Table A3: LLaMA-13B perplexity (↓) results on WikiText2 dataset with 128 sequence length.

Ratio	LLM-Pruner(e1)	SliceGPT	FLAP	Ours
90%	13.23	12.68	12.25	**12.18**
80%	16.01	15.66	13.66	**13.25**
70%	21.85	21.44	15.65	**15.17**
60%	31.17	32.77	18.53	**18.14**
50%	236.24	52.92	24.20	**22.65**

F Ablation in Reformation

To explore the influence of the number of iterations on the reformation process, we conducted experiments with varying iteration counts, as shown in Figure A1. The results, evaluated using the LLaMA-7B model with an 80% inheritance ratio on the WikiText2 dataset with a sequence length of 2048, indicate that model performance improves with an increasing number of iterations, peaking around 30 steps. Beyond this point, there is minimal difference between 30 and 50 iterations. Additionally, we examined the impact of different ρ values on the reformation, as depicted in Figure A2. Our findings show that for $\rho \in [0.01, 0.1]$, the reformed model's performance remains increasing, but deteriorates when ρ reaches 10 or bigger. Hence, $\rho = 1$ becomes the optimal value.

https://doi.org/10.52202/079017-4421

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We explain method and summarize the contribution in introduction.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The limitation is included in conclusion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

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

   Answer: [Yes]

https://doi.org/10.52202/079017-4421

Justification: We provide the theorem in methodology section and the proof in appendix.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: We provide the detailed experiment setup in experiment section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

Answer: [NA]

Justification: The dataset and model we used is open-source.

Guidelines:

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

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: We provide detailed experiment setting in experiment section.

   Guidelines:

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

7. **Experiment Statistical Significance**

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

   Answer: [NA]

   Justification: We did not report error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

https://doi.org/10.52202/079017-4421

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: We explain the computation resources in experiment section.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: Research is conducted in the paper conform with NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [Yes]

    Justification: We discuss them in conclusion.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [Yes]

    Justification: We discussed in introduction.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We originally implemented our method.

    Guidelines:

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

 https://doi.org/10.52202/079017-4421

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

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

    Answer: [Yes]

    Justification: We introduced in the introduction.

    Guidelines:

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

14. **Crowdsourcing and Research with Human Subjects**

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

    Answer: [NA]

    Justification: There is nothing related to human subjects in our work.

    Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

    Answer: [NA]

    Justification: There is nothing related to human subjects in our work.

    Guidelines:

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