⚡ Bypass Back-propagation: Optimization-based Structural Pruning for Large Language Models via Policy Gradient

Yuan Gao[2]*, **Zujing Liu**[1]*, **Weizhong Zhang**[3]*, **Bo Du**[1], **Gui-Song Xia**[2]†
[1]School of Computer Science, Wuhan University
[2]School of Artificial Intelligence, Wuhan University
[3]School of Data Science, Fudan University
ethan.y.gao@gmail.com, weizhongzhang@fudan.edu.cn
{zujing.liu, dubo, guisong.xia}@whu.edu.cn

Abstract

Recent Large-Language Models (LLMs) pruning methods typically operate at the post-training phase without the expensive weight finetuning, however, their pruning criteria often rely on **heuristically hand-crafted metrics**, potentially leading to suboptimal performance. We instead propose a novel **optimization-based structural pruning** that learns the pruning masks in a probabilistic space directly by optimizing the loss of the pruned model. To preserve efficiency, our method **eliminates the back-propagation** through the LLM *per se* during optimization, requiring only **the forward pass of the LLM**. We achieve this by learning an underlying `Bernoulli` distribution to sample binary pruning masks, where we decouple the `Bernoulli` parameters from LLM loss, facilitating efficient optimization via *policy gradient estimator* without back-propagation. Thus, our method can 1) *support global and heterogeneous pruning* (*i.e.*, automatically determine different redundancy for different layers), and 2) *optionally initialize with a metric-based method* (for our `Bernoulli` distributions). Extensive experiments conducted on LLaMA, LLaMA-2, LLaMA-3, Vicuna, and Mistral models using the C4 and WikiText2 datasets demonstrate the promising performance of our method in efficiency and effectiveness. Code is available at https://github.com/ethanygao/backprop-free_LLM_pruning.

1 Introduction

With the rapid development of Large Language Models (Brown et al., 2020; Achiam et al., 2023) (LLMs) and their expanding across various applications, the efficiency of LLMs with vast parameters and complex architectures becomes crucial for practical deployment. In this paper, we aim to compress the LLM through structural pruning, which removes certain structural components such as channels and attention heads, *i.e.*, *Width Pruning* (Ma et al., 2023; Muralidharan et al., 2024), which is also our main concern, to reduce the model size with hardware-friendly acceleration.

Structural pruning methods in the pre-LLM era prune channels or layers via *optimization*, using task loss back-propagation to determine pruning structures (Liu et al., 2018b; Blalock et al., 2020). These methods operate during training (Huang and Wang, 2018; Evci et al., 2020) or post-training (Molchanov et al., 2019; Wang et al., 2021), where the latter is more efficient without weight updates. We focus on post-training pruning for efficiency.

However, the heavy computational and memory demands of LLMs make existing *optimization-based pruning* methods less appropriate for efficiency. *Metric-based pruning* is introduced to alleviate this issue, which directly prunes specific network components based on carefully designed criteria (Sun et al., 2023; Das et al., 2023). Nonetheless, those criteria are often hand-crafted heuristically. As a result, metric-based pruning methods face challenges in achieving promising performance and generalizability, particularly at high pruning rates.

Moreover, most *metric-based pruning* methods typically prune the networks by manually-designed thresholds (Li et al., 2023; Zhang et al., 2023). Although different layers of LLMs may have varying levels of redundancy (Yin et al., 2023; Xu et al., 2024), *achieving a global and heterogeneous pruning strategy is challenging with metric-based approaches*. This is due to the significantly varying magnitudes of the manually designed metrics across layers, making it laborious or even impossible to set proper pruning threshold for each layer[1].

The above analysis leads to a natural question:

[1]As a practical compromise, most metric-based methods conduct a homogeneous/uniform pruning rate for all the layers, which violates the fact that different layers could possess the different amount of redundancy.

* Equal contribution.
† Corresponding author.

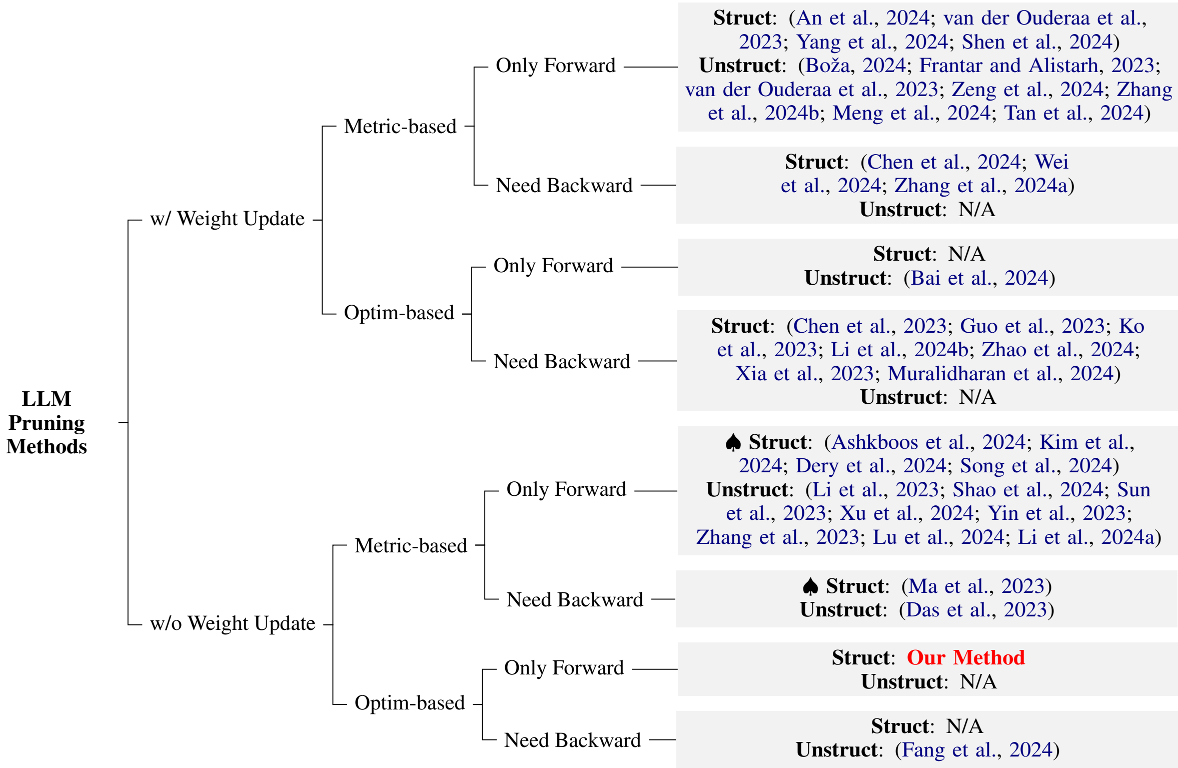

Figure 1: The taxonomy of our method among the LLM Pruning. Methods without weight update are used for comparison in our experiments (highlighted with ♠), due to the constraints on time and memory efficiency, as well as the accessibility of large-scale finetuning datasets.

*Can we attain the performance of **optimization-based methods** that facilitate global and heterogeneous pruning without relying on hand-crafted heuristics, while preserving a similar cost with the **metrics-based methods** that is affordable on a single commercial GPU?*

In view of the above analysis, our proposed method is essentially a novel lightweight *optimization-based method*, where it 1) efficiently avoids the back-propagation through the heavy LLM, 2) optionally can be initialized by an arbitrary metric-based approach. Particularly, our pruning efficiency is ensured via a *policy gradient estimator* (Williams, 1992), requiring only the LLM **forward pass** without back-propagation, which is analogous to many efficient metric-based methods and requires the same memory overhead, such as (Sun et al., 2023; An et al., 2024). Moreover, our method unifies the pruning of the entire LLM into a probabilistic space (optionally initialized by an arbitrary metric-based approach), eliminating the magnitude difference issue of most metric-based methods and therefore directly facilitating global and heterogeneous pruning across the entire LLM.

Specifically, we formulate our pruning as a binary mask optimization problem (Srinivas et al., 2017), where the binary masks determine whether to prune the corresponding structures via element-wise product. To efficiently learn those binary masks, we construct an underlying probabilistic space of `Bernoulli` distributions to sample them. By decoupling the `Bernoulli` parameters from sampled masks, our method disentangles these parameters from the LLM loss, enabling efficient optimization via *policy gradient estimator*, bypassing back-propagation[2]. Moreover, the probabilistic modeling of `Bernoulli` distribution facilitates global and heterogeneous pruning across the LLM.

The taxonomy of our methods is illustrated in Fig. 1. In the experiments, our method is compared with SOTA structural pruning methods that *do not update the model weight simultaneously*, due to the constraints on **time and memory efficiency**[3]. We extensively validate our methods using the C4 (Raffel et al., 2020) and WikiText2 (Merity et al., 2016) datasets on popular LLaMA (Touvron et al., 2023a), LLaMA-2 (Touvron et al., 2023b), LLaMA-3 (Dubey et al., 2024), Vicuna (Chiang et al., 2023), and Mistral (Jiang et al., 2023) models with various parameter sizes, pruning rates, and initializations, showing the promis-

[2]Note that our formulation can also be interpreted from a reinforcement learning (with dense rewards) perspective in terms of Markov Decision Process, detailed in Appendix A.3

[3]After pruning, it is affordable to finetune the pruned smaller model on a single commercial GPU. The performance with pruning then finetuning is included in our experiments.

ing performance and efficiency. For example, our method outperforms the SOTA methods regarding both perplexity and zero-shot performance and operates only 2.7 hours with about 35GB memory on a single A100 GPU to prune the LLaMA-2-13B model. Our method exhibits the following features:

- **Accuracy**, ensured by 1) our *optimization-based pruning* without heuristically hand-crafted metrics, which optionally takes metric-based pruning as initialization for a better convergence, and 2) the *global and heterogeneous pruning*, as supported by our probabilistic modeling of the pruning masks.
- **Efficiency** (regarding both computations and memory), achieved by the *policy gradient estimator* for back-propagation-free and forward-only optimization *w.r.t.* the heavy LLMs.

2 Related Work

Pruning has proven effective in traditional deep neural networks (Han et al., 2015; Frankle and Carbin, 2018; Kurtic et al., 2022; Liu et al., 2019; He et al., 2018), and extensive research has been conducted on this topic. Typically, post-pruning performance is restored or even enhanced through full-parameter fine-tuning (Liu et al., 2018b; Blalock et al., 2020). However, for large language models (LLMs) with vast parameters, full-parameter fine-tuning is computationally expensive and often impractical. To overcome this challenge, various pruning strategies (Ma et al., 2023; Zhang et al., 2024a; Sun et al., 2023; Ashkboos et al., 2024; Frantar and Alistarh, 2023) have been developed for LLMs in recent years. These strategies can be categorized into metric-based pruning and optimization-based pruning.

Metric-based Pruning. Metric-based pruning methods focus on designing importance metrics for model weights or modules. (Sun et al., 2023) introduces a pruning metric by considering both the magnitude of weights and activations. LLM-Pruner (Ma et al., 2023) eliminates coupled structures with low weight importance via loss change. These methods use pre-defined pruning metrics and often face challenges with high pruning rates. (Dery et al., 2024) proposed a structured pruning method using only forward passes with promising performance. It regresses the heuristically hand-crafted criteria, *e.g.*, the utility of the pruned sub-networks, and makes assumptions that may not hold universally, *e.g.*, the network's utility as a linear sum of building elements' utilities, and their utility being consistent/average-able across sub-networks.

Metric-based pruning methods use predefined criteria, potentially leading to suboptimal performance. Our optimization-based pruning framework, inspired by Neural Architecture Search (NAS) (Liu et al., 2018a), directly optimizes the loss function to identify the optimal pruned architectures while achieving higher efficiency through policy gradient optimization compared to conventional NAS that rely on back-propagation.

Optimization-based Pruning. Optimization-based pruning methods focus on determining the model mask in an optimized manner and also involve model weight updating. Sheared LLaMA (Xia et al., 2023) learns pruning masks to find a subnetwork that fits a target architecture with full-parameters updating. (Guo et al., 2023; Chen et al., 2023; Zhao et al., 2024) utilize LoRA (Hu et al., 2022) in the pruning process with weight updating.

However, these methods rely on costly back-propagation for optimization and weight updating. Instead, we propose using policy gradient estimation in the optimization process as an alternative, significantly reducing the computational demands.

3 Methodology

0

We introduce our optimization-based pruning for LLMs, which is efficient without back-propagation through the LLM, illustrated in Fig. 2.

3.1 Pruning via Probabilistic Mask Modeling

We formulate the network pruning as seeking binary masks (Srinivas et al., 2017) to determine whether the corresponding structure should be pruned or not. Those binary masks are further modeled by/sampled from the `Bernoulli` distributions stochastically. Such formulation possesses several merits: 1) the probabilistic `Bernoulli` modeling facilitates global and heterogeneous pruning across the entire LLM; 2) our stochastical sampling decouples `Bernoulli` parameters and the sampled masks from LLM loss empowering an efficient *policy gradient* optimization without back-propagate through the LLM (see Sect. 3.2); and 3) the mask formulation enables flexible pruning at channels, heads (of Multi-Head Attention, MHA), and layers.

We denote the calibration dataset with N *i.i.d.* samples as $\mathcal{D}=\{(\mathbf{x}_i,\mathbf{y}_i)\}_{i=1}^N$, $\mathbf{w}=\{\mathbf{w}_i\}_{i=1}^n$ as the complete and non-overlapped modules of a LLM with model size n, and $\mathbf{m}=\{\mathbf{m}_i\}_{i=1}^n\in\{0,1\}^n$ as the corresponding binary masks, where $\mathbf{m}_i=0$ implies $\mathbf{w}_i$ is pruned and otherwise retained. Note that $\mathbf{w}_i$ and $\mathbf{m}_i$ can be defined at various granulari-

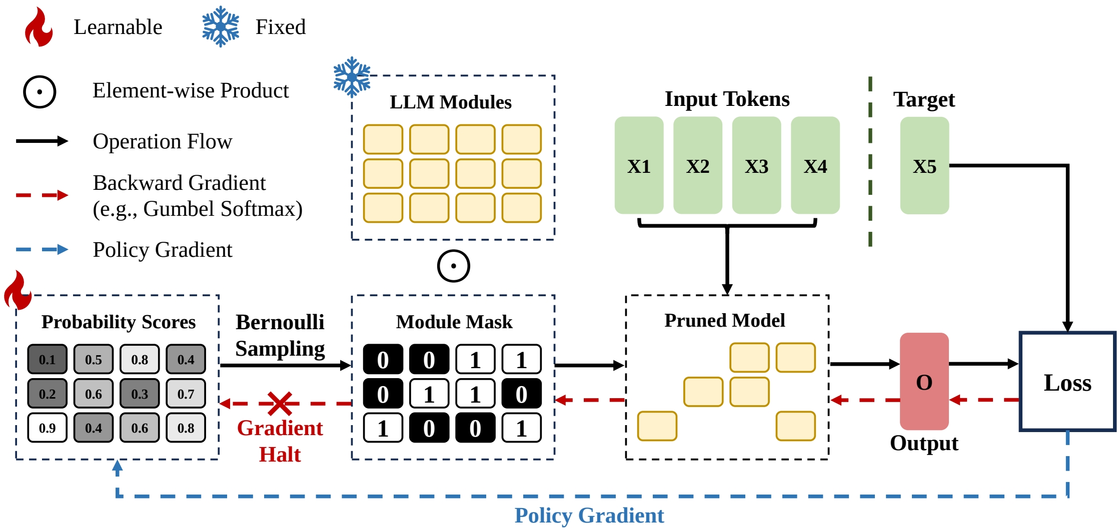

Figure 2: The overview of our method. We formulate LLM pruning as optimizing underlying `Bernoulli` distributions that sample binary masks. Being different from the conventional back-propagation method (*e.g.*, through *Gumbel Softmax* as shown by the red-dashed-arrows), our formulation decouples the masks and the `Bernoulli` parameters from the LLM loss (see Eq. (4) and Remark 3), facilitating efficient and unbiased *policy gradient* (the blue-dashed-arrow) without back-propagation through the LLM (see Eq. (5) and Remark 4).

ties such as channels, heads, and layers[4]. Then, our structural pruning of LLMs can be formulated as a binary optimization with constraints:

$$\min_{\mathbf{m}} \mathcal{L}(\mathcal{D};\mathbf{w}\odot\mathbf{m}) := \frac{1}{N}\sum_{i=1}^{N} \ell(f(\mathbf{x}_i;\mathbf{w}\odot\mathbf{m}),\mathbf{y}_i),$$
$$\text{s.t.} \ \ \|\mathbf{m}\|_1 \leq rn \ \ \text{and} \ \ \mathbf{m} \in \{0,1\}^n. \tag{1}$$

where $f(\cdot;\mathbf{w}\odot\mathbf{m})$ is the pruned network, $\ell(\cdot,\cdot)$ is the loss function, *e.g.*, the cross-entropy loss, and r is the target pruning rate. We note that the binary optimization problem Eq. (1), *i.e.*, finding optimal masks $\mathbf{m}$ from the discrete and exponentially growing solution space, is typically NP-hard.

Therefore, we relax the discrete optimization using a probabilistic approach, by treating n masks as binary *random variables* sampled from n underlying `Bernoulli` distributions with parameters $\mathbf{s} = \{\mathbf{s}_i\}_{i=1}^n \in [0,1]^n$. This yields the conditional distribution of $\mathbf{m}$ over $\mathbf{s}$:

$$p(\mathbf{m}|\mathbf{s}) = \prod_{i=1}^{n}(s_i)^{m_i}(1-s_i)^{1-m_i}. \tag{2}$$

By relaxing the ℓ_1 norm in Eq. (1) by its expectation, *i.e.*, $\|\mathbf{m}\|_1 \approx \mathbb{E}_{\mathbf{m}\sim p(\mathbf{m}|\mathbf{s})}\|\mathbf{m}\|_1 = \sum_{i=1}^n s_i = \mathbf{1}^\top\mathbf{s}$, we have the following expected loss minimization problem:

$$\min_{\mathbf{s}} \mathbb{E}_{p(\mathbf{m}|\mathbf{s})}\mathcal{L}(\mathcal{D};\mathbf{w}\odot\mathbf{m}),$$
$$\text{s.t.} \ \mathbf{1}^\top\mathbf{s} \leq rn \ \ \text{and} \ \ \mathbf{s} \in [0,1]^n. \tag{3}$$

Remark 1 *Problem* (3) *is a continuous relaxation of the discrete Problem* (1)*. The feasible region of* (3) *is the intersection of the cube* $[0,1]^n$ *and the half-space* $\mathbf{1}^\top\mathbf{s} \leq rn$*. Moreover, the parameterization of* (3) *in the probabilistic space facilitates automatically learning the redundancy across different layers for global and heterogeneous pruning.*

3.2 Policy Gradient Optimization

Conventional neural network training paradigm usually adopts back-propagation to estimate the gradient of Eq. (3), *e.g.*, through Gumbel Softmax (Maddison et al., 2016; Dupont et al., 2022) which reparameterizes the mask $\mathbf{m}$ as a function of $\mathbf{s}$, *i.e.*, $m_i = \phi(s_i)$ or $m_i = \phi(s_i,\epsilon)$ with $\epsilon \sim \mathcal{N}(0,1)$. However, the back-propagation has the following intrinsic issues in LLM pruning.

Remark 2 ***Intrinsic issues of back-propagation in LLM pruning****: 1) the back-propagation is computationally expensive and memory-intensive; 2) the computation of gradients can not be satisfied by using the sparsity in* $\mathbf{m}$, i.e., $\frac{\partial m_i}{\partial s_i} \neq 0$ *even if* $m_i = 0$*. In other words, one has to go through the full model for back-propagation even when lots of the LLM modules have been masked.*

Now we present our efficient (back-propagation-free) and unbiased optimization for Problem (3). We propose using Policy Gradient Estimator (PGE) for the gradient estimation with only forward pass, avoiding the pathology of the chain-rule estimator. Specifically, in order to update the `Bernoulli` parameters $\mathbf{s}$, we have the objective $\Phi(\mathbf{s})$:

$$\Phi(\mathbf{s}) = \mathbb{E}_{p(\mathbf{m}|\mathbf{s})}\mathcal{L}(\mathbf{m}) = \int p(\mathbf{m}|\mathbf{s})\mathcal{L}(\mathbf{m})\mathrm{d}\mathbf{m},$$
$$\text{s.t.} \ \mathbf{1}^\top\mathbf{s} \leq rn \ \text{and} \ \ \mathbf{s} \in [0,1]^n. \tag{4}$$

[4]For the channel and head granularity, we prune the dimensions of the hidden states following (Ma et al., 2023) while preserving output channels of each block to maintain residue connections(see Appendix A.2).

Our key idea is that in Eq. (4), the score vector $\mathbf{s}$ only appears in the conditional probability $p(\mathbf{m}|\mathbf{s})$ for sampling $\mathbf{m}$, which is decoupled from the network loss term $\mathcal{L}(\mathbf{m})$, short for $\mathcal{L}(\mathcal{D}; \mathbf{w} \odot \mathbf{m})$.

Remark 3 ***Differences with Gumbel Softmax****: 1) As shown in Eq.* (4)*, our PGE formulates the mask* $\mathbf{m}$ *as a random variable which is only related to the distribution* $\mathbf{s}$ *through the conditional probability* $p(\mathbf{m}|\mathbf{s})$ *of probabilistic sampling. Thus, the expensive back-propagation through the LLM can be omitted in gradient estimation using the PGE. In contrast, for the Gumbel Softmax estimator,* $\mathbf{m}$ *is a function of* $\mathbf{s}$*, requiring the back-propagation through the whole network (see the blue and red gradient flows in Fig.* 2*). 2) As a result, Gumbel Softmax is challenged by the back-propagation issues discussed in Remark* 2*. 3) Gumbel Softmax is known to be* biased *especially when the temperature is high (Huijben et al., 2022). 4) The vanilla PGE might suffer from* large variance *(Liu et al., 2020), so we exploit a variance-reduced PGE discussed later in Eq.* (7) *with theoretical analysis and empirical ablations in Appendices A.4 and A.15.*

Specifically, the optimization of Eq. (4) via the policy gradient estimator holds that:

$$
\begin{aligned}
\nabla_{\mathbf{s}}\Phi(\mathbf{s}) &= \int \mathcal{L}(\mathbf{m})\nabla_{\mathbf{s}} p(\mathbf{m}|\mathbf{s}) + \underbrace{p(\mathbf{m}|\mathbf{s})\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{m})}_{=0} \,\mathrm{d}\mathbf{m} \\
&= \int \mathcal{L}(\mathbf{m}) p(\mathbf{m}|\mathbf{s}) \nabla_{\mathbf{s}} \log(p(\mathbf{m}|\mathbf{s}))\mathrm{d}\mathbf{m} \\
&= \mathbb{E}_{p(\mathbf{m}|\mathbf{s})} \mathcal{L}(\mathbf{m}) \nabla_{\mathbf{s}} \log(p(\mathbf{m}|\mathbf{s})). \qquad (5)
\end{aligned}
$$

The final equality provides conclusive proof that $\mathcal{L}(\mathbf{m})\nabla_{\mathbf{s}} \log(p(\mathbf{m}|\mathbf{s}))$ is an unbiased stochastic gradient for $\Phi(\mathbf{s})$.

Remark 4 ***The efficiency of Eq.* (5)***: 1) Equation* (5) *can be computed purely with forward propagation. 2) The computation cost for the gradients,* i.e., $\nabla_{\mathbf{s}} \log(p(\mathbf{m}|\mathbf{s})) = \frac{\mathbf{m}-\mathbf{s}}{\mathbf{s}(1-\mathbf{s})}$*, is negligible. Therefore, our PGE is much efficient compared to the backward-propagation-based estimators.*

The stochastic gradient descent algorithm is:

$$
\begin{aligned}
\mathbf{s} &\leftarrow \mathbf{proj}_{\mathcal{C}}(\mathbf{z}), \\
\mathbf{z} &:= \mathbf{s} - \eta \mathcal{L}(\mathcal{D}_B; \mathbf{w} \odot \mathbf{m}) \nabla_{\mathbf{s}} \log(p(\mathbf{m}|\mathbf{s})).
\end{aligned} \qquad (6)
$$

where $\mathcal{D}_B = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{B}$ is batch samples from $\mathcal{D}$ with batch size B, and $\mathcal{L}(\mathcal{D}_B; \mathbf{w} \odot \mathbf{m})$ is the loss on $\mathcal{D}_B$ with the pruned model by masks $\mathbf{m}$. The projection operator $\mathbf{proj}_{\mathcal{C}}(\cdot)$ is to ensure the updated scores $\mathbf{s}$ to be constrained in the feasible domain $\mathcal{C} = \{\mathbf{1}^\top \mathbf{s} \leq K\} \bigcap \{\mathbf{s} \in [0,1]^n\}$. We implement the projection operator from (Wang and Carreira-Perpinán, 2013), the details of which can be found in Appendix A.1.

Policy gradient might suffer from large variance (Liu et al., 2020). To reduce the variance for fast and stable training, we minus a moving average baseline (Zhao et al., 2011) which is calculated by 1) obtaining the averaged loss of multiple sampling trials, then 2) taking the moving average of the current and the previous losses given a window size. Denote the baseline as δ, given window size T (set to 5), and mask sampling times N_s (set to 2), we update $\mathbf{s}$ in each training step via Eqs. (7) and (8). The theoretical analysis and empirical ablations can be found in Appendices A.4 and A.15.

$$
\mathbf{s} \leftarrow \mathbf{proj}_{\mathcal{C}}(\mathbf{z}) \text{ with } \mathbf{z} := \mathbf{s} - \eta \Bigg[\frac{1}{N_s} \sum_{i=1}^{N_s} \big(\mathcal{L}(\mathcal{D}_B; \mathbf{w} \odot \mathbf{m}^{(i)}) - \delta\big) \nabla_{\mathbf{s}} \log(p(\mathbf{m}^{(i)}|\mathbf{s})) \Bigg]. \qquad (7)
$$

$$
\delta \leftarrow \frac{T-1}{T}\delta + \frac{1}{N_s T}\sum_{i=1}^{N_s} \mathcal{L}(\mathcal{D}_B; \mathbf{w} \odot \mathbf{m}^{(i)}). \qquad (8)
$$

Our efficient pruning algorithm is summarized in Appendix A.1. Note that our formulation can be interpreted as a dense rewards reinforcement learning problem, detailed in Appendix A.3.

Initialization. Algorithms based on policy gradient usually require an effective initialization to get enhanced results. In this context, previous handcrafted pruning metric can be applied to initialize the probability of each module: $\mathbf{s}_0 \leftarrow \sigma(\mathbf{x})$, in which $\mathbf{x}$ can be any pruning metric derived from existing method, $\mathbf{s}_0$ represents the initial probability assigned to each module, and σ symbolizes a non-linear transformation. **We note that initializing from a prior metric-based method is only an option, while a random initialization strategy can already produce good performance.** Please refer to different initializations $\mathbf{x}$ and transformations σ discussed in Appendices A.17 and A.16.

Applicability of PGE in Learning Pruning Masks. We note that the precision of PGE may not match that of conventional back-propagation. Given that we are learning the **binary** masks $\mathbf{m}$ (distinct from the **float** weights), it is expected that the precision requirement of $\mathbf{s}$ can be modest. Moreover, our PGE is unbiased (compared to the biased Gumbel Softmax). These factors make the PGE suitable for learning the masks, which is empirically validated with extensive experiments. We also compared the results of using PGE and Gumbel Softmax respectively in Sect. 5.2.

Practical applicability of our efficient pruning method. Our method is particularly effective in memory-constrained settings where GPU memory may only accommodate inference for the unpruned model. This addresses a growing practical challenge, as state-of-the-art models continue to expand while many researchers and institutions face hardware limitations. Moreover, the pruned smaller model remains affordable to fine-tune under these limitations. We discuss these practical implications in detail in Appendix A.5.

4 Experiments

We conduct extensive experiments to validate the promising performance of the proposed method, across **different LLM models with various sizes**, **pruning rates**, and **initializations** (in the ablation analysis). First, we detail our experimental setups in Sect. 4.1. After that, our main results against the state-of-the-art methods for channels and heads pruning are shown in Sect. 4.2. We illustrate the zero-shot performance in Sect. 4.3, Appendices A.7 and A.9. Our method runs 2.7 hours for LLaMA-2-13B with a similar GPU memory (*i.e.*, ∼35GB) as Wanda-sp (An et al., 2024) as shown in Appendix A.6. Considering the constraints on computations and memory, we compare with the state-of-the-art methods without in-pruning weight update, and report the *pruning then finetuning* performance in Appendix A.7, as it becomes affordable to finetune a smaller pruned model. We also show generated samples of the pruned models in Table A19 of Appendix A.13 and multiple-run statistics of our method in Appendix A.14.

4.1 Experimental Setups

Structural Granularities for Pruning. We validate our method on *Head and Channel Granularity* for pruning, *i.e.*, Width Pruning. For the effects of different initializations, we extensively investigated them in Sect. 5 and Appendices A.17 and A.16.

Head and Channel Granularity: We follow (Ma et al., 2023; An et al., 2024) to prune the *heads* of the multi-head attention (MHA) modules and the *channels* of the MLP modules in Sect. 4.2. We initialize our methods with an efficient metric-based structural pruning method, *i.e.*, Wanda-sp (An et al., 2024). Our method is compared to the state-of-the-art Wanda-sp (An et al., 2024), LLM-Pruner (Ma et al., 2023), SliceGPT (Frantar and Alistarh, 2023), and Bosai (Dery et al., 2024).

Additionally, we also validate our method on *Layer Granularity*, *i.e.*, Depth Pruning (Kim et al., 2024; Song et al., 2024), by pruning the entire transformer layer, shown in Appendix A.8.

LLM Models and Sizes. LLaMA-{7B, 13B} (Touvron et al., 2023a), LLaMA-2-{7B, 13B} (Touvron et al., 2023b), LLaMA-3-8B (Dubey et al., 2024), Vicuna-{7B, 13B} (Chiang et al., 2023), and Mistral-7B-Instruct-v0.3 (Jiang et al., 2023) are used as the source models in our experiments.

Pruning Rate. Promising performance with a high pruning rate could be challenging to obtain when employing metric-based pruning, owing to the heuristically designed metrics. To validate the superior performance of our optimization-based pruning under this situation, we select high pruning rates ranging from 30% to 50%, *i.e.*, **structurally** removing 30% to 50% model parameters.

Datasets. We perform the experiments following the cross-dataset settings in (Sun et al., 2023), where the C4 dataset (Raffel et al., 2020) is used for training and the WikiText2 dataset (Merity et al., 2016) is used for evaluation. This challenging cross-dataset setup potentially better reflects the generalization of the pruned model.

Training and Evaluation Details. We update the underlying `Bernoulli` distributions (for mask sampling) simply using SGD with a learning rate of 6e-3 for LLaMA-3 experiments and 2e-3 for the remaining. The batch size is fixed to 8 and we train our lightweight policy gradient estimator for 1 epoch on the C4 dataset with 120K segments, where each segment has a sequence length of 128. Ablations on calibration dataset size are also conducted, detailed in Appendix A.19.

To reduce the evaluation variance, we deterministically generate the pruned evaluation architecture, *i.e.*, given a pruning rate r, we first rank all the $\mathbf{s}$, then deterministically set $\mathbf{m}$ corresponding to the minimal r of $\mathbf{s}$ as 0 (otherwise 1). We report the perplexity on the WikiText2 dataset using a sequence length of 128. Given a tokenized sequence $\mathbf{X} = (x_0, x_1, \dots, x_t)$, the perplexity of $\mathbf{X}$ is:

$$\text{Perplexity}(\mathbf{X}) = \exp\left\{-\frac{1}{t}\sum_{i}^{t}\log p_\theta(x_i|x_{<i})\right\},$$

where $\log p_\theta(x_i|x_{<i})$ is the log-likelihood of token x_i conditioned on the preceding tokens $x_{<i}$.

4.2 Results on Channels and Heads Pruning

The results of channels and heads pruning are shown in Table 1. Our method achieves the lowest perplexity scores. It verifies the superiority of optimization-based global and heterogeneous pruning. Especially, such outperformance is more

Method	PruneRate	LLaMA		LLaMA-2		LLaMA-3	Vicuna	
		7B	13B	7B	13B	8B	7B	13B
Dense	0%	12.62	10.81	12.19	10.98	14.14	16.24	13.50
LLM-Pruner	30%	38.41	24.56	38.94	25.54	40.18	48.46	31.29
SliceGPT		-	-	40.40	30.38	183.94	52.23	57.75
Bonsai		30.49	26.24	39.01	24.23	80.89	44.28	54.16
Wanda-sp		98.24	25.62	49.13	41.57	92.14	57.60	80.74
Ours		**25.61**	**19.70**	**28.18**	**21.99**	**38.99**	**34.51**	**26.42**
LLM-Pruner	40%	72.61	36.22	68.48	37.89	70.60	88.96	46.88
SliceGPT		-	-	73.76	52.31	353.09	89.79	130.86
Bonsai		60.65	58.17	69.18	50.97	204.61	95.32	272.10
Wanda-sp		110.10	165.43	78.45	162.50	213.47	85.51	264.22
Ours		**42.96**	**28.12**	**39.81**	**31.52**	**63.85**	**51.86**	**43.59**
LLM-Pruner	50%	147.83	67.94	190.56	72.89	145.66	195.85	91.07
SliceGPT		-	-	136.33	87.27	841.20	160.04	279.33
Bonsai		275.63	148.92	216.85	146.38	440.86	180.75	424.33
Wanda-sp		446.91	406.60	206.94	183.75	413.86	242.41	373.95
Ours		**72.02**	**49.08**	**65.21**	**52.23**	**119.75**	**71.18**	**68.13**

Table 1: Results (perplexity) on *channels and heads* pruning. Our method is initialized by Wanda-sp (please also refer to Sect. 5.1 and Appendix A.17 for a detailed discussion about initializations). All the methods are calibrated using the C4 dataset and validated on the WikiText2 dataset *w.r.t.* perplexity.

Method	PruneRate	PPL ↓	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	Average
Dense	0%	14.14	79.71	60.19	72.61	80.09	50.34	68.59
LLM-Pruner	30%	40.18	71.38	37.84	55.64	57.78	27.21	49.97
SliceGPT		183.94	68.34	**53.92**	57.22	49.41	28.07	51.39
Bonsai		80.89	64.53	36.10	55.09	47.64	22.52	45.18
Wanda-sp		92.14	59.74	31.46	52.64	44.02	19.88	41.55
Ours		**38.99**	**72.25**	43.56	**59.04**	**59.85**	**29.44**	**52.83**
LLM-Pruner	40%	70.60	66.26	31.90	54.06	49.74	22.52	44.90
SliceGPT		353.09	61.53	**39.98**	52.80	36.66	**25.17**	43.23
Bonsai		204.61	58.81	29.43	48.93	33.21	18.15	37.71
Wanda-sp		213.47	56.58	27.46	50.35	32.07	17.06	36.70
Ours		**63.85**	**67.63**	37.36	**56.91**	**50.67**	24.91	**47.50**
LLM-Pruner	50%	145.65	61.15	29.10	**51.93**	39.98	19.36	40.30
SliceGPT		841.20	56.37	**32.66**	48.38	32.45	**22.10**	38.39
Bonsai		440.86	55.66	26.94	50.51	30.64	17.83	36.32
Wanda-sp		413.86	55.39	27.07	49.72	29.59	18.26	36.01
Ours		**119.75**	**62.51**	30.89	51.85	**41.12**	20.65	**41.40**

Table 2: Perplexity (PPL) and zero-shot accuracies (%) of LLaMA-3-8B for 5 zero-shot tasks.

significant at larger pruning rates over 40%. The results on Mistral-7B-Instruct-v0.3 (Jiang et al., 2023) are shown in Table A11 of Appendix A.12. We further validate the method for pruning rates from 10% to 50% with more evaluation in Appendix A.10, and provide a comparison between our method and existing approaches that incorporate weight update, detailed in Appendix A.11.

4.3 Performance on Zero-shot Tasks

We follow SliceGPT (Ashkboos et al., 2024) to assess our pruned LLM by EleutherAI LM Harness (Gao et al., 2023) on five zero-shot tasks: PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2021), HellaSwag (Zellers et al., 2019), ARC-e and ARC-c (Clark et al., 2018) with the average scores across the five tasks. Our results on LLaMA-3-8B and LLaMA-2-7B in Tables 2 and A8 of Appendix A.9, demonstrate overall superior performance to the baselines, though C4-only pruning may negatively impact on particular cross-dataset zero-shot tasks such as Hellaswag (Zellers et al., 2019).

5 Ablation Analysis

We investigate 1) the effect of various initialization of our method in Sect. 5.1, Appendices A.17 and A.16, 2) comparison with Gumbel Softmax, 3) performance of global and heterogeneous pruning versus that of local and homogenous pruning in Sect. 5.3, 4) the remaining modules after pruning in Appendix A.18, and 5) the effect of the variance-reduced policy gradient in Appendix A.15.

5.1 Different Initializations

Our `Bernoulli` policy requires initialization to perform policy gradient optimization and to sample pruning masks. In this section, we investigate the **effect** and the **necessity** of using different metric-based methods as initializations. Moreover, the initialization of the `Bernoulli` policy should be probabilistic values between 0 and 1, but the metrics calculated by the metric-based methods (Sun et al., 2023; Ma et al., 2023) may not hold this range. We thus discuss **different projection strategies** that transform those metrics to [0, 1] in Appendix A.16.

Method	PruneRate	Perplexity	PruneRate	Perplexity	PruneRate	Perplexity
LLM-Pruner	30%	38.94	40%	68.48	50%	190.56
SliceGPT		40.40		73.76		136.33
Bonsai		39.01		69.18		216.85
Wanda-sp		49.13		78.45		206.94
Ours (Random Init)	30%	37.24	40%	60.16	50%	160.75
Ours (Random-Prog. Init)		<u>31.43</u>		<u>49.86</u>		<u>86.55</u>
Ours (LLM-Pruner Init)	30%	35.75	40%	65.32	50%	116.80
Ours (Wanda-sp Init)		**28.18**		**39.81**		**65.21**

Table 3: Channels and heads pruning results with *different initializations* on LLaMA-2-7B. **Bold** and <u>Underscored</u> denote the first and second best results, respectively.

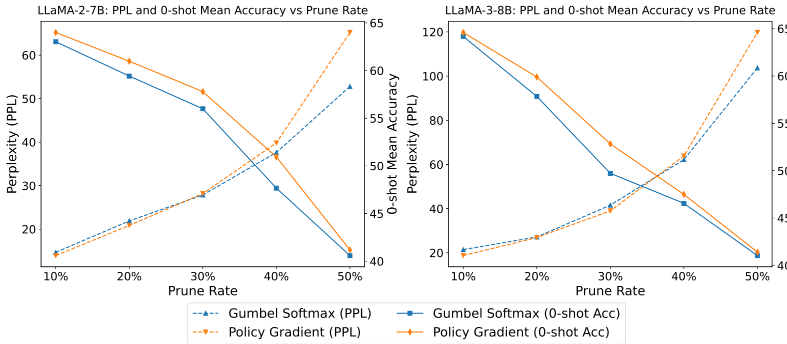

Figure 3: Comparison of Policy Gradient and Gumbel Softmax.

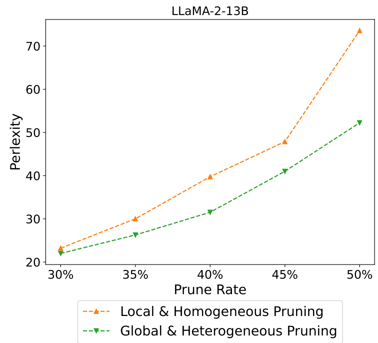

Figure 4: Global vs. local pruning.

To address the practical case when a metric-based pruning is not *apriori*, we propose progressive pruning with random initialization (*Random-Progressive*), trained progressively with increasing pruning rates (each for only $1/3$ epoch). Details can be found in Appendix A.17.

Different initializations are tested on LLaMA-2-7B. The baselines include simple random initialization with the target pruning rate (*Random*) and progressive pruning with random initialization (*Random-Progressive*). For channels and heads pruning, we investigate the initializations from Wanda-sp (An et al., 2024) and LLM-Pruner[5] (Ma et al., 2023), as shown in Table 3.

Our results in Tables 3 demonstrate that 1) different initializations lead to different results, 2) compared to the state-of-the-art methods, our method with most initializations except the random one exhibit new state-of-the-art results, and 3) The proposed *Random-Progressive* initialization ranks the second place in most cases, surpassing previous state-of-the-art methods, **which suggests less necessity for employing a prior metric-based method to initiate our algorithm**.

5.2 Comparision with Gumbel Softmax

As highlighted in Remark 3, our proposed PGE bypasses the costly back-propagation process through the LLM required by Gumbel Softmax, while maintaining comparable gradient estimation accuracy.

[5]We follow LLM-Pruner to fix the first four and the last two layers from pruning.

To substantiate this advantage, we conduct empirical ablation studies comparing different gradient estimators. The performance on LLaMA-2-7B and LLaMA-3-8B, measured by both perplexity and mean accuracy of 5 zero-shot tasks, of our PGE approach and back-propagation/Gumbel Softmax approach in Figure 3. The performance of PGE is generally comparable to that of the Gumbel Softmax, except that PGE exhibits slightly higher perplexity at a 50% pruning rate. This discrepancy may be attributed to the increased variance observed at this pruning level, which consequently amplifies the gradient estimation error.

We also illustrate the training time and memory usage of LLaMA-2-7B in Table 4, which demonstrates that our method achieves comparable performance with significantly reduced resources.

Method	Memory (GiB)		Time (h)
	Min	Max	
Gumbel Softmax	19.93	23.97	3.47
Policy Gradient	**17.23**	**17.39**	**1.56**

Table 4: Memory and Time Consumption Comparison between Gumbel Softmax and Policy Gradient.

5.3 Merits of Global Pruning

Our method is able to perform global and heterogeneous pruning throughout the entire network, which is difficult for metric-based pruning methods (Sun et al., 2023; Ma et al., 2023), as the metrics across different layers often exhibit different magnitudes. As a compromise, those metric-based methods prune each layer locally and homogeneously.

We validate the merits of global and heterogeneous pruning over local and homogeneous pruning, where we compare our method with a variant in which we prune each layer homogeneously. The channels and heads pruning results on LLaMA-2-13B are shown in Fig. 4, demonstrating that the global and heterogeneous pruning significantly outperforms its local and homogeneous counterpart.

6 Conclusion

We propose an efficient optimization-based structural pruning method for LLMs, which 1) does not need back-propagation through the LLM *per se*, 2) enables global and heterogeneous pruning throughout the LLM. Our method can take a metric-based pruning as initialization to achieve further improved performance. We implement our method by learning an underlying `Bernoulli` distribution of binary pruning mask. As we decouple the `Bernoulli` parameter and the sampled masks from the LLM loss, the `Bernoulli` distribution can thus be optimized by a policy gradient estimator without back-propagation through the LLM. Our method operates for 2.7 hours with approximately 35GB of memory on a single A100 GPU. Extensive experiments on various LLM models and sizes with detailed ablation analysis validate the promising performance of the proposed method.

7 Limitations

Firstly, as an optimization-based pruning, though our method exhibits *improved performance* over the (heuristic) metric-based methods, and a *similar memory complexity* (approximately 35GB, as only LLM forward is required), it simultaneously *requires more training time* for optimization (*e.g.*, 2.7 hours for LLaMA-2-13B) than the metric-based pruning methods.

Secondly, there exist advanced policy gradient algorithms with potentially lower variance from the reinforcement learning community. As 1) the primary focus of this paper is on the back-propagation-free formulation of the LLM pruning problem, and 2) our formulation ensures dense rewards at each step, we thus use a basic policy gradient algorithm similar to REINFORCE with simple variance reduction using a moving average baseline. We leave exploiting more powerful policy gradient algorithms as our future work.

Lastly, the performance of the proposed method on specific domains/tasks can rely heavily on the availability of domain-specific datasets. Though the cross-dataset evaluation is verified *w.r.t.* perplexity, pruning with only C4 dataset might have a negative influence on certain cross-dataset zero-shot tasks such as WinoGrande and Hellaswag.

8 Ethical Considerations

We have developed an efficient pruning method for Large Language Models (LLMs) that significantly accelerates inference speed. This approach optimizes computational efficiency and reduces energy consumption for online-deployed LLMs like ChatGPT, improving user experience while promoting sustainable AI. However, it also inherits the inherent ethical challenges of LLM technologies, requiring careful consideration in deployment.

9 Acknowledgements

This work was supported by the National Natural Science Foundation of China (62306214, 62325111, 62472097), the Natural Science Foundation of Hubei Province of China (2023AFB196), the Knowledge Innovation Program of Wuhan-Shuguang Project (2023010201020258), and Shanghai Municipal Science and Technology Commission (24511106102).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2024. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 10865–10873.

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

Guangji Bai, Yijiang Li, Chen Ling, Kibaek Kim, and Liang Zhao. 2024. Sparsellm: Towards global pruning of pre-trained language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. 2020. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146.

Vladimír Boža. 2024. Fast and optimal weight update for pruned large language models. *arXiv preprint arXiv:2401.02938*.

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

Tianyi Chen, Tianyu Ding, Badal Yadav, Ilya Zharkov, and Luming Liang. 2023. Lorashear: Efficient large language model structured pruning and knowledge recovery. *arXiv preprint arXiv:2310.18356*.

Xiaodong Chen, Yuxuan Hu, and Jing Zhang. 2024. Compressing large language models by streamlining the unimportant layer. *arXiv preprint arXiv:2403.19135*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

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

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

Rocktim Jyoti Das, Liqun Ma, and Zhiqiang Shen. 2023. Beyond size: How gradients shape pruning decisions in large language models. *arXiv preprint arXiv:2311.04902*.

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

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

Robin Dupont, Mohammed Amine Alaoui, Hichem Sahbi, and Alice Lebois. 2022. Extracting effective subnetworks with gumbel-softmax. In *ICIP*, pages 931–935.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. 2020. Rigging the lottery: Making all tickets winners. In *International conference on machine learning*, pages 2943–2952. PMLR.

Gongfan Fang, Hongxu Yin, Saurav Muralidharan, Greg Heinrich, Jeff Pool, Jan Kautz, Pavlo Molchanov, and Xinchao Wang. 2024. Maskllm: Learnable semi-structured sparsity for large language models. *arXiv preprint arXiv:2409.17481*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Song Guo, Jiahang Xu, Li Lyna Zhang, and Mao Yang. 2023. Compresso: Structured pruning with collaborative prompting learns compact large language models. *arXiv preprint arXiv:2310.05015*.

Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. 2018. Amc: Automl for model compression and acceleration on mobile devices. In *ECCV*, pages 784–800.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *ICLR*.

Zehao Huang and Naiyan Wang. 2018. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 304–320.

Iris AM Huijben, Wouter Kool, Max B Paulus, and Ruud JG Van Sloun. 2022. A review of the gumbel-max trick and its extensions for discrete stochasticity in machine learning. *IEEE TPAMI*, 45(2):1353–1371.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

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

Jongwoo Ko, Seungjoon Park, Yujin Kim, Sumyeong Ahn, Du-Seong Chang, Euijai Ahn, and Se-Young Yun. 2023. Nash: A simple unified framework of structured pruning for accelerating encoder-decoder language models. *arXiv preprint arXiv:2310.10054*.

Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. 2022. The optimal bert surgeon: Scalable and accurate second-order pruning for large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4163–4181.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Lujun Li, Peijie Dong, Zhenheng Tang, Xiang Liu, Qiang Wang, Wenhan Luo, Wei Xue, Qifeng Liu, Xiaowen Chu, and Yike Guo. 2024a. Discovering sparsity allocation for layer-wise pruning of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Shengrui Li, Xueting Han, and Jing Bai. 2024b. Nuteprune: Efficient progressive pruning with numerous teachers for large language models. *arXiv preprint arXiv:2402.09773*.

Yun Li, Lin Niu, Xipeng Zhang, Kai Liu, Jianchen Zhu, and Zhanhui Kang. 2023. E-sparse: Boosting the large language model inference through entropy-based N: M sparsity. *arXiv preprint arXiv:2310.15929*.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018a. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Yanli Liu, Kaiqing Zhang, Tamer Basar, and Wotao Yin. 2020. An improved analysis of (variance-reduced) policy gradient and natural policy gradient methods. *NeurIPS*, 33:7624–7636.

Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. 2019. Metapruning: Meta learning for automatic neural network channel pruning. In *ICCV*.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2018b. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*.

Haiquan Lu, Yefan Zhou, Shiwei Liu, Zhangyang Wang, Michael W Mahoney, and Yaoqing Yang. 2024. Alphapruning: Using heavy-tailed self regularization theory for improved layer-wise pruning of large language models. *arXiv preprint arXiv:2410.10912*.

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

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and B Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods.

Xiang Meng, Kayhan Behdin, Haoyue Wang, and Rahul Mazumder. 2024. Alps: Improved optimization for highly sparse one-shot pruning for large language models. *arXiv preprint arXiv:2406.07831*.

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

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11264–11272.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Bhuminand Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. 2024. Compact language models via pruning and knowledge distillation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21(140):1–67.

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

Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. 2010. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559.

Hang Shao, Bei Liu, and Yanmin Qian. 2024. One-shot sensitivity-aware mixed sparsity pruning for large language models. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11296–11300. IEEE.

Xuan Shen, Pu Zhao, Yifan Gong, Zhenglun Kong, Zheng Zhan, Yushu Wu, Ming Lin, Chao Wu, Xue Lin, and Yanzhi Wang. 2024. Search for efficient large language models. *arXiv preprint arXiv:2409.17372*.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. 2024. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. *arXiv preprint arXiv:2402.09025*.

Suraj Srinivas, Akshayvarun Subramanya, and R Venkatesh Babu. 2017. Training sparse neural networks. In *CVPR Workshops*, pages 138–145.

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

Zhendong Tan, Xingjun Zhang, and Zheng Wei. 2024. Wrp: Weight recover prune for structured sparsity. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6433–6443.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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

Tycho FA van der Ouderaa, Markus Nagel, Mart van Baalen, Yuki M Asano, and Tijmen Blankevoort. 2023. The llm surgeon. *arXiv preprint arXiv:2312.17244*.

Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. 2021. Neural pruning via growing regularization. In *ICLR*.

Weiran Wang and Miguel A Carreira-Perpinán. 2013. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv preprint arXiv:1309.1541*.

Boyi Wei, Kaixuan Huang, Yangsibo Huang, Tinghao Xie, Xiangyu Qi, Mengzhou Xia, Prateek Mittal, Mengdi Wang, and Peter Henderson. 2024. Assessing the brittleness of safety alignment via pruning and low-rank modifications. *arXiv preprint arXiv:2402.05162*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*.

Peng Xu, Wenqi Shao, Mengzhao Chen, Shitao Tang, Kaipeng Zhang, Peng Gao, Fengwei An, Yu Qiao, and Ping Luo. 2024. Besa: Pruning large language models with blockwise parameter-efficient sparsity allocation. *arXiv preprint arXiv:2402.16880*.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024. Laco: Large language model pruning via layer collapse. *arXiv preprint arXiv:2402.11187*.

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

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

Hongchuan Zeng, Hongshen Xu, Lu Chen, and Kai Yu. 2024. Multilingual brain surgeon: Large language models can be compressed leaving no language behind. *arXiv preprint arXiv:2404.04748*.

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. 2024a. Loraprune: Structured pruning meets low-rank parameter-efficient fine-tuning. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 3013–3026.

Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. 2024b. Plug-and-play: An efficient post-training pruning method for large language models. In *ICLR*.

Yuxin Zhang, Lirui Zhao, Mingbao Lin, Yunyun Sun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. 2023. Dynamic sparse no training: Training-free fine-tuning for sparse llms. *arXiv preprint arXiv:2310.08915*.

Bowen Zhao, Hannaneh Hajishirzi, and Qingqing Cao. 2024. Apt: Adaptive pruning and tuning pretrained language models for efficient training and inference. *arXiv preprint arXiv:2401.12200*.

Tingting Zhao, Hirotaka Hachiya, Gang Niu, and Masashi Sugiyama. 2011. Analysis and improvement of policy gradient estimation. In *NIPS*.

Analysis	Projection operator for sparsity constraint and the overall algorithm in Appendix A.1.	(Page 13)
	Details on hidden states pruning for channel and head granularities in Appendix A.2.	(Page 13)
	A reinforcement learning perspective of the proposed method in Appendix A.3.	(Page 13)
	Theoretical analysis of moving average baseline for policy gradient in Appendix A.4.	(Page 14)
	An in-depth discussion on the practical applicability of the proposed method in Appendix A.5.	(Page 15)
Results	Statistics of the training time & memory, and the inference latency in Appendix A.6.	(Page 15)
	Performance after pruning and (then) finetuning in Appendix A.7.	(Page 16)
	Performance on layer pruning in Appendix A.8.	(Page 16)
	Zero-shot performance on LLaMA-2-7B in Appendix A.9.	(Page 17)
	Further evaluation with wider pruning rate range A.10.	(Page 17)
	Comparision with approaches with weight update in Appendix A.11.	(Page 17)
	Performance on Mstral-7B-Instruct-V0.3 in Appendix A.12.	(Page 18)
	Generated samples of the pruned model in Appendix A.13.	(Page 18)
	Random error-bar statistics in Appendix A.14.	(Page 18)
Ablations	Ablations on the moving average baseline for policy gradient in Appendix A.15	(Page 19)
	Ablations on projection strategy for initialization: from metric to probability in Appendix A.16	(Page 19)
	More ablations with different initializations in Appendix A.17	(Page 20)
	More ablations of the post-pruning modules on LLaMA-2-7B in Appendix A.18	(Page 20)
	Ablations of the calibration data size in Appendix A.19	(Page 21)

Table A1: Summary of the Appendix materials.

A Appendix

We discuss the following additional analyses, results, and ablations in the appendices. The catalogs are in Table A1.

A.1 Projection Operator for Sparsity Constraint and the Overall Algorithm

Details of the Projection Operator. In our proposed probabilistic framework, the sparsity constraint manifests itself in a feasible domain on the probability space defined in Problem (3). We denote the feasible domain as $\mathcal{C} = \{\mathbf{1}^\top \mathbf{s} \leq K\} \bigcap \{\mathbf{s} \in [0,1]^n\}$. The theorem (Wang and Carreira-Perpinán, 2013) below shows that the projection of a vector onto C can be calculated efficiently.

Theorem 1. *For each vector* $\mathbf{z}$ *, its projection* $\mathbf{proj}_{\mathcal{C}}(\mathbf{z})$ *in the set* C *can be calculated as follows:*

$$\mathbf{proj}_{\mathcal{C}}(\mathbf{z}) = \min(1, \max(0, \mathbf{z} - v_2^* \mathbf{1})) \tag{A1}$$

where $v_2^* = max(0, v_1^*)$ *with* v_1^* *being the solution of the following equation*

$$\mathbf{1}^T \left[\min(1, \max(0, \mathbf{z} - v_1^* \mathbf{1}))\right] - K = 0 \tag{A2}$$

Equation (A2) can be solved by the bisection method efficiently.

The theorem above as well as its proof is standard and it is a special case of the problem stated in (Wang and Carreira-Perpinán, 2013). This component, though not the highlight of our work, is included for the reader's convenience and completeness.

Algorithm. The pseudo-code of our overall algorithm is detailed below.

Algorithm 1 Pseudo-code of PG pruning

Input: target remaining ratio $r > 0$, a dense pretrained network $\mathbf{w}$, the step size $\eta > 0$, mini-batch size $B > 0$, moving average window size T, and calibration dataset $\mathcal{D}$

Initialize: Init probability $\mathbf{s}$ from any pruning metric $\mathbf{x}$, ans set moving average $\delta = 0$

while until convergence **do**

  Sample a mini-batch from the entire calibration dataset: $\mathcal{D}_B = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^B \sim \mathcal{D}$

  Sample $\mathbf{m}^{(i)}$ from $p(\mathbf{m}|s)$, $i = 1, 2, \ldots, N_s$

  Update the moving average baseline δ via Eq. (8)

  Uptate $\mathbf{s}$ via Eqs. (7), (A1), and (A2).

end while

A.2 Details on Hidden States Pruning for Channel and Head Granularities

We note that for pruning on the channel and head granularities, it must be guaranteed that the final output dimension for each block (*e.g.*, multi-head attention, MLP) should remain, so as to facilitate the residue connections (*e.g.*, additions) across blocks. We thus follow (Ma et al., 2023; An et al., 2024) to prune the dimensions of the hidden states, while keeping the final output channels unchanged, ensuring that they can be added to the input through the residual connections. A conceptual figure illustrating this procedure is shown in Fig. A1.

A.3 A Reinforcement Learning Perspective

Our formulation can also be interpreted from the dense-reward model-free reinforcement learning perspective. Particularly, the heavy LLM can be viewed as the agnostic and fixed <u>environment</u>.

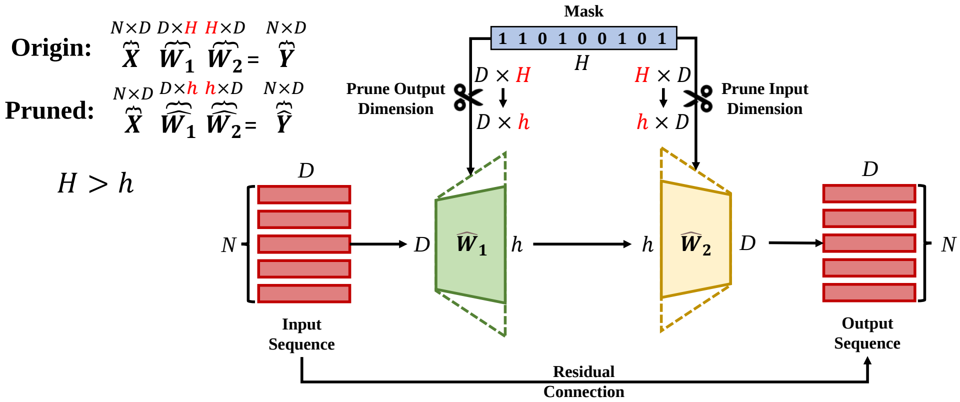

Figure A1: Output dimension is invariant for each block that might be used for residual connections, but instead prune the dimension of the intermediate hidden state.

In terms of the Markov Decision Process (MDP) (**action** a, **states** s, **state transition probability** p, **reward** r, **discount factor** γ), the environment takes the **action** a sampled from the current `Bernoulli` *policy* π to insert the binary masks for pruning, produces the **states** s as the masked/pruned network deterministically (*i.e.*, the **state transition probability** p is constantly 1), and generate the stepwise dense **reward** r as the performance (*e.g.*, the cross-entropy loss) of the pruned LLM. Since our problem exhibits dense rewards, the **discount factor** γ is 1.

As a result, our policy to take actions, *i.e.*, the `Bernoulli` distribution to sample the binary masks, can be learned efficiently exploiting the *policy gradient estimator* (similar to REINFORCE), without back-propagating through the agnostic and fixed environment of the heavy LLM.

A.4 Theoretical Analysis of Moving Average Baseline for Policy Gradient

We give the theoretical analysis on the variance reduction technique by considering a general-purpose technique for reducing the variance of Monte Carlo method with the general problem $\mathbb{E}_{p(\mathbf{x};\theta)}[f(\mathbf{x})]$. We take a strategy that replacing the function $f(\mathbf{x})$ in the expectation by a substitute function $\tilde{f}(\mathbf{x})$ whose expectation $\mathbb{E}_{p(\mathbf{x};\theta)}[\tilde{f}(\mathbf{x})]$ is the same, but whose variance is lower. Given a function $h(\mathbf{x})$ with a known expectation $\mathbb{E}_{p(\mathbf{x};\theta)}[h(\mathbf{x})]$, we can easily construct such a substitute function along with the corresponding estimator as follows:

$$\tilde{f}(\mathbf{x}) = f(\mathbf{x}) - \beta(h(\mathbf{x}) - \mathbb{E}_{p(\mathbf{x};\theta)}[h(\mathbf{x})]), \tag{A3}$$

$$\bar{\eta}_N = \frac{1}{N}\sum_{n=1}^{N}\tilde{f}(\hat{\mathbf{x}}^n) = \bar{f} - \beta(\bar{h} - \mathbb{E}_{p(\mathbf{x};\theta)}[h(\mathbf{x})]).$$

where $\hat{\mathbf{x}}^n \sim p(\mathbf{x};\theta)$ and $\bar{f}$ and $\bar{h}$ are the sample averages. β is a control coefficient and $h(\mathbf{x})$ is considered as control variate. We can show that if the variance of $h(\mathbf{x})$ is finite, the unbiasedness the estimator Eq. A3 is maintained, *e.g.*,

$$\begin{aligned}\mathbb{E}_{p(\mathbf{x};\theta)}[(\mathbf{x};\beta)] =& \mathbb{E}[\bar{f} - \beta(\bar{h} - \mathbb{E}_{h(\mathbf{x})})] \\ =& \mathbb{E}[\bar{f}] = \mathbb{E}_{p(\mathbf{x};\theta)}[f(\mathbf{x})].\end{aligned} \tag{A4}$$

For the variance of the estimator (for N = 1), we have

$$\begin{aligned}\mathbb{V}[\tilde{f}] =& \mathbb{V}[f(\mathbf{x}) - \beta(h(\mathbf{x}) - \mathbb{E}_{p(\mathbf{x};\theta)}[h(\mathbf{x})])] \\ =& \mathbb{V}[f] - 2\beta Cov[f,h] + \beta^2\mathbb{V}[h].\end{aligned} \tag{A5}$$

By minimizing Eq. A5 we can find that the optimal value of the coefficient is

$$\beta^* = \frac{\mathrm{Cov}[f,h]}{\mathbb{V}[h]} = \sqrt{\frac{\mathbb{V}[f]}{\mathbb{V}[h]}}\mathrm{Corr}(f,h), \tag{A6}$$

where we expressed the optimal coefficient in terms of the variance of f and h and the correlation coefficient $\mathrm{Corr}(f,h)$. The effectiveness of a control variate can be measured by the variance ratio between its estimator and the original estimator: it is effective if the ratio is substantially less than 1. Using the optimal control coefficient in Eq. A6, the potential variance reduction is

$$\begin{aligned}\frac{\mathbb{V}[\tilde{f}(\mathbf{x})]}{\mathbb{V}[f(\mathbf{x})]} =& \frac{\mathbb{V}[f(\mathbf{x} - \beta(h(\mathbf{x}) - \mathbb{E}_{p(\mathbf{x};\theta)}[h(\mathbf{x})])]}{\mathbb{V}[f(\mathbf{x})]} \\ =& 1 - \mathrm{Corr}(f(\mathbf{x}), h(\mathbf{x}))^2.\end{aligned} \tag{A7}$$

Therefore, as long as $f(\mathbf{x})$ and $h(\mathbf{x})$ are not uncorrelated, we can always obtain a reduction in variance using control variables. In practice, the optimal β^* will not be known and so we will usually need to estimate it empirically.

In our problem formulation of structured pruning for LLMs, $\mathbb{E}_{p(\mathbf{m}|\mathbf{s})}\mathcal{L}(\mathcal{D};\mathbf{w}\odot\mathbf{m})\nabla_{\mathbf{s}}\log(p(\mathbf{m}|\mathbf{s}))$, is a score-function gradient estimator [1], in which $p(\mathbf{m}|\mathbf{s})$ is the `Bernoulli` distribution of each module of LLMs with $\mathbf{s}$ corresponds the θ, $\mathbf{m}$ corresponds the θ and $\mathcal{L}(\mathcal{D};\mathbf{w}\odot\mathbf{m})\nabla_{\mathbf{s}}\log(p(\mathbf{m}|\mathbf{s}))$ corresponds $f(\mathbf{x})$ in the preliminary. To reduce the variance of a score-function gradient estimator, one simple and general way is to use the score function itself as a control variate, that is $h(\mathbf{m}) = \delta\nabla_\theta \log p(\mathbf{m}|\mathbf{s})$ and δ is an independent estimation of $\mathcal{L}(\mathcal{D}; \mathbf{w} \odot \mathbf{m})$, since its expectation under the measure is zero, as

$$\begin{aligned} &\mathbb{E}_{p(\mathbf{m}|\mathbf{s})}[\delta\nabla_\mathbf{s} \log p(\mathbf{m}|\mathbf{s})] \\ =&\delta \int p(\mathbf{m}|\mathbf{s}) \frac{\nabla_\mathbf{s} p(\mathbf{m}|\mathbf{s})}{p(\mathbf{m}|\mathbf{s})} d\mathbf{m} \\ =&\delta\nabla_\mathbf{s} \int p(\mathbf{m}|\mathbf{s}) d\mathbf{m} = \delta\nabla_\mathbf{s} 1 = \mathbf{0}. \end{aligned} \tag{A8}$$

Therefore, the estimator in Eq. A3 format is:

$$\begin{aligned} \bar{\eta}_N = &\frac{1}{N}\sum_{n=1}^{N}(\mathcal{L}(\mathcal{D}; \mathbf{w} \odot \mathbf{m}^{(n)}) \\ &- \beta\delta)\nabla_\mathbf{s} \log(p(\mathbf{m}^{(n)}|\mathbf{s}));\ \mathbf{m}^{(n)} \sim p(\mathbf{m}|\mathbf{s}), \end{aligned} \tag{A9}$$

where $\mathbf{m}^{(n)}$ is the sampled mask of modules. In reinforcement learning, the term $\beta\delta$ is called a baseline (Williams, 1992) and has historically been estimated with a running average of the cost. Note that δ needs to be estimated, we choose moving average baseline in our method, which is a commonly used baseline in policy gradient estimation (Zhao et al., 2011; Sehnke et al., 2010).

A.5 Practical Applicability Discussion

While our optimization-based pruning method inherently requires more training time for optimization than the metric-based methods (Table A2), it delivers superior performance with similar memory efficiency, *i.e.*our method is particularly well suited for scenarios where *GPU memory is strictly limited*, especially when *available memory only allows for inference on the unpruned model.* This case is common and practical, *as SOTA models continue to grow in size, such constraints are prevalent for individual practitioners or academic institutions.*

Thus, in our main experiments, we focus on comparisons with methods that *do not require weight updates and use only forward propagation.*

Compared to Gumbel-Softmax. Gumbel-Softmax involves backpropagation and is prone to OUT-OF-MEMORY (OOM) errors under constrained GPU memory limits. In contrast, our method avoids this issue, achieving comparable performance (Fig. 3) while using 38% less memory and 122% less training time (Table 4).

Method	Metric/Optim	Pruning Time
LLM-Pruner	Metric	38.11s
SliceGPT	Metric	17.15min
Wanda-sp	Metric	36.35s
Ours (Policy Gradient)	Optim	1.56h
Ours (Gumbel Softmax)	Optim	3.47h

Table A2: Pruning time comparison for LLaMA-2-7B.

Compared to metric-based pruning. Our method requires similar memory, since it also relies only on forward propagation, but delivers significantly better performance. This is because metric-based methods depend on heuristics, whereas our approach directly optimizes the loss. Notably, increasing compute for chasing a better metric does not easily close such heuristic vs. optimization performance gap; for instance, our method outperforms and is also faster than a recent metric-based SOTA, Bonsai (Dery et al., 2024), across Tables 1, 2, A5, A6, A8, A9.

Compared to weight-update methods with similar training time. These approaches also require back-propagation and thus suffer from OOM issues under memory constraints. In contrast, once our efficient pruning is applied, the resulted smaller model enables feasible finetuning. We show that such pruning-then-finetuning further enhances performance in memory-constrained settings (Tables A5, A6, and A10). Especially, Table A10 demonstrates that our method, when followed by finetuning, outperforms SOTA weight-update methods such as FLAP (An et al., 2024) and Search-LLM (Shen et al., 2024).

A.6 Statistics of the Training Time & Memory, and the Inference Latency

Method	7B		13B	
	Min	Max	Min	Max
Wanda-sp	17.5	20.3	29.5	36.9
Ours	17.2	17.4	34.1	35.8

Table A3: Memory requirements (GB) for channel and head pruning on LLaMA-2-7B/13B.

Method	P.R	#Params	Memory	Latency	PPL
LLM-Pruner	30%	4.837	**9290.54**	53.53	27.13
SliceGPT		5.293	10181.81	50.24	22.29
Ours		**4.796**	9338.24	**46.94**	**12.68**
LLM-Pruner	40%	4.197	**8069.55**	**36.75**	53.21
SliceGPT		4.501	8826.01	46.84	39.21
Ours		**4.149**	8096.25	42.85	**15.95**
LLM-Pruner	50%	3.539	**6815.05**	**31.49**	171.57
SliceGPT		3.730	7274.01	41.73	65.92
Ours		**3.500**	6880.92	34.62	**27.63**

Table A4: #Params (B), memory requirements (MiB), latency (s) and WikiText2 perplexity (*i.e.*PPL) of LLaMA-2-7B. Experiments are conducted on NVIDIA A100 40G, with 2048 sequence length and 4 batch size for full GPU utilization. P.R. is short for pruning rate.

Our training times for channel and head pruning on LLaMA-2-7B and LLaMA-2-13B are 1.76 and 2.72 hours, respectively. The statistics of the

Method	PruneRate	PPL ↓	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	Average
Dense	0%	12.19	78.02	57.17	68.43	76.30	43.51	64.69
LLM-Pruner	30%	33.45	74.10	46.61	58.17	64.31	33.62	55.36
SliceGPT		78.59	74.70	**64.29**	61.96	57.49	36.69	59.03
Bonsai		33.23	75.03	49.69	62.19	67.34	32.25	57.30
Wanda-sp		32.01	73.88	50.08	62.19	67.09	34.47	57.54
Ours		**25.34**	**76.01**	51.80	**64.33**	**67.93**	**36.86**	**59.39**
LLM-Pruner	40%	40.21	70.29	40.45	53.04	53.03	27.30	48.82
SliceGPT		175.67	65.29	**56.77**	**60.06**	42.68	**31.74**	51.31
Bonsai		44.71	72.36	45.10	58.80	59.64	30.03	53.19
Wanda-sp		43.71	70.40	42.73	52.72	57.24	29.95	50.61
Ours		**29.43**	**72.74**	45.75	55.72	**61.36**	31.06	**53.33**
LLM-Pruner	50%	44.83	**67.30**	35.47	51.93	48.23	21.84	44.95
SliceGPT		296.97	58.65	**46.83**	**55.09**	36.99	**28.33**	45.18
Bonsai		62.95	66.70	40.16	54.30	49.83	26.53	**47.50**
Wanda-sp		110.12	63.27	32.71	52.72	43.48	20.73	42.58
Ours		**39.46**	67.03	36.42	52.41	**50.17**	24.15	46.04

Table A5: Perplexity (PPL) and Accuracies (%) of LLaMA-2-7B for 5 zero-shot tasks with pruning rates from 30% to 50% after weight fine-tuning on Alapca dataset.

time consumption of different pruning methods on LLaMA-2-7B are shown in Table A2. Although our method is slower than metric-based methods such as Wanda-sp (An et al., 2024), the trade-off is justified by the substantial performance gains delivered by our optimization-based approach.

The GPU memory requirements for channel and head pruning on LLaMA-2-7B and LLaMA-2-13B for our method, as well as the representative metric-based method, *e.g.*, Wanda-sp, are illustrated in Table A3. We do not compare it to LLM-Pruner and SliceGPT because 1) the LLM-Pruner requires much more memory for back-propagation (therefore the authors also used the CPU memory), 2) the original implementation of SliceGPT also used both CPU and GPU memory for computations. Table A3 shows that our method exhibits a similar GPU memory requirement to the efficient Wanda-sp, as we only need the forward pass of the LLM. The slight additional memory required by our method comes from the need to store the `Bernoulli` parameters $\mathbf{s}$ and sampled masks $\mathbf{m}$.

We note that for the same pruning rate (*i.e.*, similar remaining #Params), the inference latencies of pruned models from different structural pruning methods are expected to be comparable, as the inference latency is mainly affected by the #Params given the same architecture. We validate this in Table A4. Table A4 demonstrates that, given the same pruning rates, our pruned model has very much close #Params, memory, and inference latencies to that pruned by LLM-Pruner, while our perplexity significantly outperformed all the counterparts. We note that under the same pruning rates, SliceGPT often possesses different (higher) #Params, memory, and inference latencies than our method and LLM-Pruner, potentially because SliceGPT alters the network structure during the pruning.

A.7 Performance after Pruning and (Then) Finetuning

We note that after pruning, it becomes affordable to finetune a smaller pruned model. Therefore, following the idea from (Ma et al., 2023), we finetune the post-pruning model *w.r.t.* the perplexity with LoRA (Hu et al., 2022). Specifically, we utilize 4k samples from the Alpaca (Taori et al., 2023) dataset, which has a sequence length of 1024. For all weight fine-tuning experiments, we use *lora_r* $=16$, *lora_alpha* $=10$, and use default values for all other hyperparameters in the HuggingFace PEFT package (Mangrulkar et al., 2022).

The cross-dataset performance on WikiText of the post-pruning fine-tuned model for LLaMA-2-7B and LLaMA-3-8B is illustrated in Tables A5 and A6, which demonstrate that our method achieves consistently superior performance before and after fine-tuning. Compared with the pre-finetuned model, the performance of most post-finetuned models shows significant improvements, and our models remain the best for most cases after finetuning, which validates our potential for narrowing the performance gap after pruning and for being applicable in practical use.

A.8 Results on Layer Pruning

We also validate the layer granularity by pruning the entire transformer layer, consisting of an MHA module and a MLP module. Note that pruning on the layer granularity is less exploited for LLMs, thus in this experiment, we use the lightweight Layerwise-PPL (Kim et al., 2024) for initialization, and compare our method with Layerwise-PPL (Kim et al., 2024) and SLEB (Song et al., 2024).

Method	PruneRate	PPL ↓	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	Average
Dense	0%	14.14	79.71	60.19	72.61	80.09	50.34	68.59
LLM-Pruner	30%	35.11	**74.64**	46.93	60.22	**66.16**	34.13	56.42
SliceGPT		226.39	70.29	**56.47**	60.06	53.20	34.81	54.97
Bonsai		42.59	71.87	45.17	59.51	66.50	**36.52**	55.91
Wanda-sp		38.04	70.84	44.11	59.43	62.96	34.04	54.28
Ours		**33.91**	74.48	46.62	**63.69**	65.70	34.30	**56.96**
LLM-Pruner	40%	47.83	**71.54**	40.71	55.40	**62.16**	28.92	51.75
SliceGPT		523.05	63.66	**42.75**	53.12	41.88	27.65	45.81
Bonsai		57.31	69.58	39.47	53.98	57.24	28.67	49.79
Wanda-sp		56.32	65.18	36.33	54.77	51.56	24.32	46.43
Ours		**47.28**	70.56	41.09	**59.98**	59.97	**29.01**	**52.12**
LLM-Pruner	50%	68.14	**67.95**	35.81	53.12	**53.91**	26.36	47.43
SliceGPT		963.42	60.83	**37.04**	52.25	37.21	25.26	42.52
Bonsai		88.72	62.89	34.84	52.80	47.73	24.15	44.48
Wanda-sp		84.53	61.42	32.12	52.72	41.83	21.07	41.83
Ours		**67.48**	67.08	35.84	**54.38**	53.54	**26.45**	**47.46**

Table A6: Perplexity (PPL) and Accuracies (%) of LLaMA-3-8B for 5 zero-shot tasks with pruning rates from 30% to 50% after weight fine-tuning on Alapca dataset.

Method	PruneRate	LLaMA		LLaMA-2		LLaMA-3	Vicuna	
		7B	13B	7B	13B	8B	7B	13B
Dense	0%	12.62	10.81	12.19	10.98	14.14	16.24	13.50
Layerwise-PPL	30%	31.65	24.23	24.83	20.52	45.47	37.99	29.85
SLEB		27.36	**20.45**	23.43	**19.97**	37.92	29.40	26.37
Ours		**24.45**	24.44	**23.20**	21.93	**36.42**	**29.16**	**24.68**
Layerwise-PPL	40%	54.97	50.57	41.45	32.48	75.12	64.96	54.12
SLEB		44.65	**32.79**	40.26	**30.16**	73.61	**48.99**	43.12
Ours		**42.73**	39.07	**38.26**	30.99	**63.70**	54.37	**35.73**
Layerwise-PPL	50%	107.12	183.93	126.08	78.04	393.18	517.46	153.53
SLEB		108.87	77.38	131.49	**55.23**	303.03	146.12	92.32
Ours		**94.97**	**66.38**	**104.37**	69.92	**295.39**	**126.24**	**84.90**

Table A7: Results on *layers* pruning. Our method is initialized by Layerwise-PPL (please also refer to Appendix A.17 for detailed discussion about initializations). All the methods are calibrated using the C4 dataset and validated on the WikiText2 dataset *w.r.t.* perplexity.

We illustrate the results on layer pruning in Table A7, which show that our method generally achieves better performance than the baseline methods, especially at pruning rates above 40%. For LLaMA-13B and LLaMA-2-13B with a moderate pruning rate of 30%, our method performs comparably to Layerwise-PPL. This suggests that with coarser layer granularity, the search space may be limited, and larger 13B models with more redundancy benefit from metric-based pruning at lower rates.

A.9 Zero-shot Performance on LLaMA-2-7B

We validate the zero-shot performance of LLaMA-2-7B with pruning rates from 30% to 50%, shown in Table A8. We note that the overall performance is in general superior to the baselines, though using only the C4 dataset for pruning might introduce a negative influence on some particular cross-dataset zero-shot tasks such as WinoGrande (Sakaguchi et al., 2021) and Hellaswag (Zellers et al., 2019).

A.10 Further Evaluation on Wider Pruning Rate Range

For a more comprehensive validation of the proposed method, we experiment on LLaMA-3-8B with a wider range of pruning rate, from 10% to 50%, following the same settings of the main results. Beyond WikiText2 perplexity and 5 zero-shot tasks, the comparison on MMLU benchmark (Hendrycks et al., 2020) for five-shot and additional zero-shot task, LAMBADA (Paperno et al., 2016), RACE (Lai et al., 2017) and BoolQ (Clark et al., 2019), are also included. The results shown in Table A9 demonstrate the consistent superiority of our method across a wide range of sparsity levels.

A.11 Comparision with Approaches with Weight Update

We additionally conduct experiments of performance comparison with approaches that involve weight update (Shen et al., 2024; An et al., 2024). We follow the pruning settings of (Shen et al., 2024), in which we calibrate on WikiText2 dataset and evaluate perplexity on it with a sequence length of 2048. For zero-shot tasks evaluation, we follow the procedure applied in (Shen et al., 2024). We compared our vanilla method (pruning only, without weight update) with (Shen et al., 2024; An et al., 2024), denoted as ours (prune-only) in Table A10. The experiments are performed on LLaMA-7B consistent with (Shen et al., 2024).

Method	PruneRate	PPL ↓	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	Average
Dense	0%	12.19	78.02	57.17	68.43	76.30	43.51	64.69
LLM-Pruner	30%	38.94	71.81	43.64	54.06	63.42	30.30	52.64
SliceGPT		40.40	72.31	**60.11**	**63.22**	53.10	32.00	56.15
Bonsai		39.01	73.94	47.05	60.93	59.93	30.37	54.44
Wanda-sp		49.13	71.60	46.62	60.30	63.01	34.04	55.11
Ours		**28.18**	**75.41**	50.34	61.60	**66.03**	**35.58**	**57.79**
LLM-Pruner	40%	68.48	67.52	35.76	51.70	48.31	24.65	45.59
SliceGPT		73.76	65.40	**48.91**	**60.38**	42.13	26.88	48.74
Bonsai		69.18	68.44	40.63	55.41	48.11	26.19	47.75
Wanda-sp		78.45	64.63	35.61	52.17	48.11	25.51	45.21
Ours		**39.81**	**71.11**	42.44	55.72	**56.94**	**28.50**	**50.94**
LLM-Pruner	50%	190.56	59.52	29.74	50.11	36.48	21.84	39.54
SliceGPT		136.33	59.47	**37.96**	**56.27**	33.63	**22.78**	**42.02**
Bonsai		216.85	59.52	32.63	53.12	33.54	22.61	40.28
Wanda-sp		206.94	54.30	26.81	52.80	29.12	19.20	36.45
Ours		**65.21**	**61.80**	30.94	52.64	**40.11**	20.47	41.19

Table A8: Perplexity (PPL) and accuracies (%) of LLaMA-2-7B for 5 zero-shot tasks with 30% - 50% pruning rates.

Method	PruneRate	PPL ↓	LAMBADA	RACE	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	MMLU
Dense	0%	14.14	69.14	40.29	81.41	79.71	60.19	72.61	80.09	50.34	66.58
LLM-Pruner	10%	19.25	53.85	37.32	73.24	77.04	52.93	68.11	73.44	39.50	48.37
SliceGPT		39.14	59.67	**40.29**	**80.58**	75.57	54.78	68.35	72.56	40.87	55.38
Bonsai		20.43	54.12	38.75	75.69	77.64	54.96	70.32	75.92	**43.00**	39.56
Wanda-sp		35.94	28.58	32.25	57.12	69.64	42.86	64.64	65.07	32.68	29.00
Ours		**18.73**	**62.63**	39.62	79.39	**78.94**	**56.88**	**69.45**	**76.18**	41.55	**56.38**
LLM-Pruner	20%	28.62	37.45	32.63	55.96	74.92	42.94	59.19	65.57	32.51	26.78
SliceGPT		84.99	46.52	**39.52**	**76.12**	73.23	48.24	63.69	64.77	34.13	33.55
Bonsai		29.05	48.36	35.41	64.16	75.46	47.00	67.01	65.61	35.41	29.60
Wanda-sp		47.43	22.76	31.10	53.21	67.90	39.27	60.38	58.50	29.01	27.96
Ours		**26.92**	**51.02**	37.03	74.22	**76.28**	**51.00**	**67.64**	**69.15**	**35.41**	**44.99**
LLM-Pruner	30%	40.18	28.74	30.63	58.56	71.38	37.84	55.64	57.78	27.21	25.36
SliceGPT		183.94	29.17	**36.75**	**68.20**	68.34	**53.92**	57.22	49.41	28.07	25.89
Bonsai		80.89	15.50	31.29	45.29	64.53	36.10	55.09	47.64	22.52	23.41
Wanda-sp		92.14	13.87	28.52	51.28	59.74	31.46	52.64	44.02	19.88	26.25
Ours		**38.99**	**44.81**	35.41	66.15	**72.25**	43.56	**59.04**	**59.85**	**29.44**	**27.38**
LLM-Pruner	40%	70.60	14.09	28.13	59.57	66.26	31.90	54.06	49.74	22.52	25.36
SliceGPT		354.24	16.28	**33.4**	**62.87**	61.53	**39.98**	52.80	36.66	**25.17**	26.10
Bonsai		204.61	2.04	23.35	46.27	58.81	29.43	48.93	33.21	18.15	25.09
Wanda-sp		213.47	8.73	28.23	52.78	56.58	27.46	50.35	32.07	17.06	25.57
Ours		**63.85**	**30.80**	32.63	61.96	**67.63**	37.36	**56.91**	**50.67**	24.91	**27.50**
LLM-Pruner	50%	145.66	4.37	24.50	45.53	61.15	29.10	**51.93**	39.98	19.36	24.36
SliceGPT		841.20	7.99	**30.72**	57.00	56.37	**32.66**	48.38	32.45	**22.10**	24.16
Bonsai		440.86	0.25	22.10	42.20	55.66	26.94	50.51	30.64	17.83	24.35
Wanda-sp		413.86	3.07	23.25	45.99	55.39	27.07	49.72	29.59	18.26	24.73
Ours		**119.75**	**17.43**	26.79	**61.80**	**62.51**	30.89	51.85	**41.12**	20.65	**25.33**

Table A9: Perplexity (PPL) and accuracies (%) of LLaMA-3-8B for 8 zero-shot tasks and MMLU benchmark in five-shot with pruning rates from 10% to 50%.

Additionally, since fine-tuning becomes feasible after pruning smaller models, we also included results for our prune-then-finetune approach for comparison. The results demonstrate that our pruning-only method achieves comparable performance to (Shen et al., 2024), while the prune-then-finetune approach, involving weight update, outperforms (Shen et al., 2024) in the majority of cases.

A.12 Results on Mistral-7B-Instruct-v0.3

To further validate the performance of the proposed method on more LLMs, we additionally perform experiments on Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), which calibrates on C4 dataset and evaluates on the WikiText2 dataset (*e.g.*, cross-dataset setting, as those in our Table 1). We note that the original implementations of SliceGPT (Ashkboos et al., 2024) and Bonsai (Dery et al., 2024) were based on LLaMA-2, which do not trivially adapt to the Mistral model directly, therefore, we exclude SliceGPT and Bonsai for comparison.

The results, including both perplexity and the zero-shot performance, on Mistral-7B-Instruct-v0.3 in Table A11 demonstrate the consistent superiority of our method across various LLMs.

A.13 Generated Samples of the Pruned Model

We provide some generated sentences of the pruned models. Table A19 illustrates the generated sentences of LLaMA-2-7B with the pruning rate of 30%, from different pruning methods, where the input prompts are adopted from (Ma et al., 2023). We observe that the generated content from our method not only maintains superior coherence and innovation but also is more factual and professional despite a high pruning rate (30%). It demonstrates that our method optimizes the balance between knowledge retention and performance in the compression process, ensuring the quality and diversity of the generated text.

A.14 Random Error-Bar Statistic

The standard deviation statistics of our method are shown in Table A12. Theoretically, the variance

Method	PruneRate	PPL ↓	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	Average
Dense	0%	5.68	78.35	72.99	67.01	67.45	41.38	65.44
FLAP	10%	6.34	75.41	68.68	67.01	65.78	38.48	63.07
search-llm		**6.10**	76.88	70.71	**67.56**	68.39	40.10	64.73
ours (prune-only)		6.17	77.53	**71.85**	66.14	69.23	40.87	65.12
ours (prune-then-finetune)		7.03	**77.64**	71.53	67.32	**69.49**	**41.98**	**65.59**
FLAP	20%	7.40	74.21	64.98	64.40	59.89	37.80	60.26
search-llm		6.89	74.92	67.29	**64.64**	64.23	36.52	61.52
ours (prune-only)		7.07	74.92	**68.32**	61.56	62.63	37.20	60.93
ours (prune-then-finetune)		**6.29**	**76.11**	68.19	63.38	**66.24**	**38.99**	**62.58**

Table A10: Perplexity (PPL) and accuracies (%) of LLaMA-7B for 5 zero-shot tasks with pruning rates from 10% to 20%, compared with approaches with weight update, FLAP (An et al., 2024) and search-llm (Shen et al., 2024).

Method	PruneRate	PPL ↓	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	Average
Dense	0%	12.70	81.77	64.84	74.51	84.22	57.34	72.54
LLM-Pruner	30%	**30.32**	69.58	41.52	57.77	53.99	28.58	50.29
Wanda-sp		47.30	75.68	49.94	62.35	64.90	36.60	57.89
Ours		31.87	**76.49**	**52.69**	**64.48**	**67.76**	**36.77**	**59.64**
LLM-Pruner	40%	49.30	65.18	34.79	52.80	46.42	23.89	44.62
Wanda-sp		76.45	68.01	38.75	52.64	52.36	26.28	47.61
Ours		**43.02**	**68.61**	**40.80**	**56.67**	**54.80**	**27.82**	**49.74**
LLM-Pruner	50%	86.24	61.31	30.64	49.64	37.67	22.52	40.36
Wanda-sp		407.33	56.69	29.08	49.25	32.36	21.59	37.79
Ours		**74.25**	**65.18**	**35.02**	**51.06**	**48.15**	**22.61**	**44.40**

Table A11: Perplexity (PPL) and accuracies (%) of Mistral-7B-Instruct-v0.3 for 5 zero-shot tasks with 30% - 50% pruning rates.

arises from stochastic sampling from `Bernoulli` distribution in the policy gradient optimization if the initialization is fixed. Thus, we fixed initialization as Wanda-sp to calculate the standard deviation of the proposed method. Experiments of head and channel pruning, along with layer pruning, are executed using LLaMA-2-7B for 10 run trials, demonstrating reasonable deviation.

Granularity	PruneRate		
	30%	40%	50%
Head & Channel	28.18±1.83	39.81±1.41	65.21±2.52
Layer	23.20±0.67	38.26±2.68	104.37±1.05

Table A12: Mean and standard deviation of our method for LLaMA-2-7B.

A.15 Ablations on the Moving Average Baseline for Policy Gradient

We conduct experiments on pruning channels and heads of LLaMA-2-7B/13B with/without the *Moving Average Baseline* in policy gradient. Table A13 illustrates the effectiveness of the moving average baseline in the policy gradient estimator for our proposed pruning method.

Moreover, we also tested all the hyper-parameters, *e.g.*, the window size and mask sampling times (T and N_s in Eq. (8)). The results in Table A14 demonstrate that being different from with vs. without moving average baseline, small T and N_s can already offer promising performance, further increasing them only produces marginal improvement. In other words, our method is robust to those hyper-parameter values. Considering computational overhead, we choose small $T = 5$ and $N_s = 2$ throughout our entire experiments.

Method	PruneRate	LLaMA-2-7B	LLaMA-2-13B
w/o MAB	30%	32.53	24.73
with MAB		**28.18**	**21.99**
w/o MAB	40%	60.99	64.34
with MAB		**39.81**	**31.52**
w/o MAB	50%	69.47	185.87
with MAB		**65.21**	**52.23**

Table A13: Ablations on the proposed Moving Average Baseline (MAB) in the policy gradient estimator for Channels and heads pruning on LLaMA-2-7B/13B.

Hyper-params	T			N_s		
	3	5⋆	7	2⋆	3	4
Perplexity	21.23	21.99	**20.08**	21.99	21.71	**21.37**

Table A14: Ablation on the hyperparameters of the moving average baseline, *i.e.*, different window sizes T and mask sampling times N_s. Perplexity is tested on the WikiText2 dataset of LLaMA-2-13B with 30% pruning rate. The hyper-parameter values used in the main results are denoted with ⋆.

A.16 Ablations on Projection Strategy for Initialization: From Metric to Probability

As the initialization of our `Bernoulli` policy should be probabilistic values between 0 and 1, but the metrics calculated by the metric-based methods (Sun et al., 2023; An et al., 2024; Ma et al., 2023) may not hold this range, we thus need to project those metric values to [0, 1] as our initialization.

(a) Channels and Heads Pruning.

Method	Sparsity	7B	13B
Sigmoid-Norm	30%	**28.18**	**21.99**
Score-Const		32.25	25.38
Sigmoid-Norm	35%	**32.52**	**26.27**
Score-Const		40.61	40.51
Sigmoid-Norm	40%	**39.81**	**31.52**
Score-Const		44.46	52.10
Sigmoid-Norm	45%	**52.07**	**40.99**
Score-Const		65.31	61.04
Sigmoid-Norm	50%	**65.21**	**52.23**
Score-Const		77.07	88.72

(b) Layer Pruning.

Method	Sparsity	7B	13B
Sigmoid-Norm	30%	**23.20**	21.93
Score-Const		25.32	**19.31**
Sigmoid-Norm	35%	33.27	26.46
Score-Const		**31.37**	**23.40**
Sigmoid-Norm	40%	**38.26**	30.99
Score-Const		42.30	**29.25**
Sigmoid-Norm	45%	69.23	**39.26**
Score-Const		**63.91**	39.50
Sigmoid-Norm	50%	**104.37**	69.92
Score-Const		135.51	**54.37**

Table A15: Results with *different projection strategies* for pruning heads, channels, and layers on LLaMA-2-7B/13B. Initialization metrics are from Wanda-sp for heads/channels and Layerwise-PPL for layers.

Method	PruneRate	Perplexity	PruneRate	Perplexity	PruneRate	Perplexity
Layerwise-PPL	30%	24.83	40%	41.45	50%	126.08
SLEB		<u>23.43</u>		40.26		131.49
Ours (Random Init)	30%	26.65	40%	42.76	50%	125.20
Ours (Random-Prog. Init)		30.05		<u>38.28</u>		<u>111.87</u>
Ours (Layerwise-PPL Init)	30%	**23.20**	40%	**38.26**	50%	**104.37**

Table A16: Layer pruning results with *different initializations* using LLaMA-2-7B. **Bold** and <u>Underscored</u> denote the first and second best results, respectively.

We introduce two projection strategies from metric values $\mathbf{m}$ to probabilities $\mathbf{s}$. The first is called *Sigmoid-Norm* strategy, which is applied in our main experiments:

$$\mathbf{s} = \text{sigmoid}(\text{Norm}(\mathbf{x})) \quad \text{(A10)}$$

where $\text{Norm}(\cdot)$ is used to linearly normalize the input to a Gaussian distribution with 0 mean and unit variance, then $sigmoid(\cdot)$ is used to transform the input to $[0, 1]$.

An alternative second strategy is named *Score-Const*. It straightforwardly sets mask 1 from metric-based methods as a constant c, and mask 0 as $1 - c$:

$$s_i = \begin{cases} c, & \text{if } m_i = 1, \\ 1-c, & \text{if } m_i = 0, \end{cases} \quad \text{(A11)}$$

The constant c is set to 0.8 in the following experiments, indicating that the initialized `Bernoulli` probability of the remaining modules is 0.8 and those to be pruned is 0.2.

The results of different projection strategies on LLaMA-2-7B/13B are detailed in Table A15, which shows that the *Sigmoid-Norm* projection outperforms its *Score-Const* counterpart for most cases. It may be because the order-preserving projection strategy of *Sigmoid-Norm* preserves more information about relative importance among modules, and therefore benefits the optimization.

A.17 More Ablations with Different Initializations

Progressive Pruning with Random (Random-Progressive) Initialization. Our progressive pruning with random initialization is inspired by the facts that 1) the *continous* `Bernoulli` probability learned by our method indicates the importance of the corresponding module, therefore the *continous* probability scores from a low pruning rate (*e.g.*, 10%) encodes fatal information and can be naturally used as the initialization for a higher pruning rate (*e.g.*, 15%); and 2) the LLMs is likely to exhibit large redundancy when the pruning rate is extremely low (*e.g.*, 5%), thus random initialization will not significantly degrade the pruning performance (compared to a carefully chosen metric-based pruning initialization) given an extremely low pruning rate such as 5%. Therefore, to validate our method without a prior metric-based initialization, we propose a progressive pruning strategy, by starting from 5% pruning rate with random initialization and progressively pruning rate to 50% by a step size of 5%. We train this strategy with each pruning rate for 1/3 epoch to maintain efficiency.

Moreover, Table A16 shows *layer* pruning results with different initializations on LLaMA-2-7B.

A.18 Analysis of the Post-Pruning Modules

As global and heterogeneous pruning is performed through our optimization, it is interesting to investigate the pruned modules in each layer. We show the channels, heads, and layers sparsity (*i.e.*, the pruned portion of the corresponding granularity) on LLaMA-2-{7B, 13B} with channels and heads pruning at 40% in Fig. A2.

Figures A2 demonstrate that the pruned LLM exhibits low sparsity in the first and last layers, which is consistent with the previous studies that

nsamples	PPL	
	mean	std
64	27.85	1.16
128	27.94	1.54
256	28.05	1.28
512	28.52	1.37
1024	27.92	1.46
40k	27.60	1.32
120k	27.19	1.18

(a) Effect of the number of calibration samples

seqlen	PPL	
	mean	std
128	27.94	1.54
256	27.90	0.86
512	28.51	0.51
1024	27.51	0.68
2048	26.68	0.64

(b) Effect of the calibration sequence lengths

Table A17: Ablations on the number of calibration samples and sequence lengths on PPL (evaluated with 128 sequence length).

Method	PruneRate	PPL ↓	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	Average
Dense	0%	12.19	78.02	57.17	68.43	76.30	43.51	64.69
SliceGPT	20%	24.87	74.92	49.91	66.22	69.11	35.32	59.10
Wanda-sp		23.08	77.09	**54.34**	65.90	71.21	**40.27**	61.76
Bonsai		23.03	76.82	53.10	64.25	71.17	39.85	61.04
Ours		**19.61**	**77.09**	53.45	**66.38**	**72.39**	40.02	**61.87**
SliceGPT	30%	40.96	71.71	44.58	**64.80**	60.73	30.20	54.40
Wanda-sp		42.96	74.59	48.43	59.12	63.47	34.30	55.98
Bonsai		48.30	72.85	48.25	57.77	63.8	33.87	55.31
Ours		**27.13**	**75.79**	**49.00**	62.27	**65.36**	**34.56**	**57.40**

Table A18: LLaMA-2-7B pruning results with *the same calibration data* in all methods, evaluated in perplexity (PPL) and accuracies (%) for 5 zero-shot tasks with pruning rates 20% and 30%.

these layers have a profound impact on the performance of LLMs (Ma et al., 2023). Moreover, it can be observed that the heads (of MHA) granularity exhibits lower sparsity in the shallow layers (especially in the first layer), while such observation does not hold for the channels (of MLP) granularity. In other words, the pruned sparsity of the channel granularity is more evenly distributed whereas the deeper layers have slightly less sparsity. This might imply that the shallow layers focus more on attention, while the deeper layer imposes slightly more responsibility for lifting the feature dimensions through MLP.

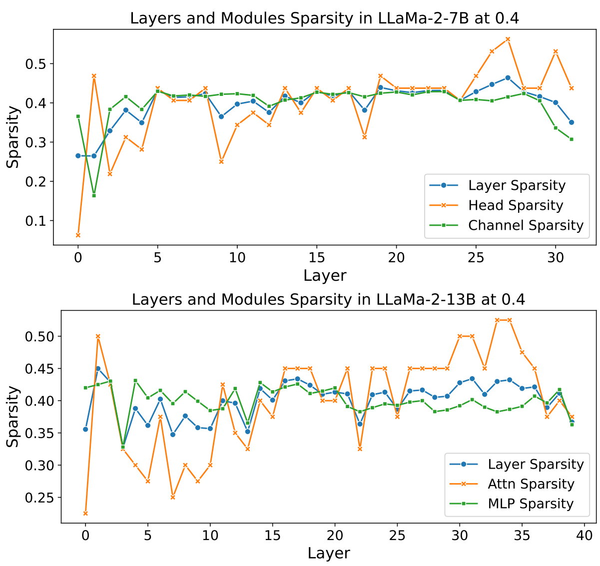

Figure A2: Channels, heads, and layers sparsities of LLaMA-2-{7B, 13B} with 40% pruning rate.

A.19 Analysis of the Calibration Data Size

Ablations on the number of calibration samples and sequence lengths. Extensive ablations on the number of calibration samples (`nsamples`) and sequence lengths (`seqlen`) are conducted to show the robustness of our method. Specifically, we calibrate our method using C4, and evaluate on WikiText2 on LLaMA2-7B with 30% sparsity, we report the mean and std over 5 replicates for each experiment.

- **Effect of the number of calibration samples** in Table A17a: we fix `seqlen` to 128 and vary `nsamples` from 64 to 120K.
- **Effect of the calibration seqence lengths** in Table A17b: we fix `nsamples` to 128 and vary `seqlen` from 128 to 2048.

The ablations demonstrate that our method is *robust to various calibration data sizes, in terms of both samples*[6] *and sequence lengths.*

Pruning performance using the same calibration data. We prune LLaMA-2-7B for 20% and 30% sparsity with *the same calibration data* across all methods, i.e., `nsamples` = 128 and `seqlen` = 2048 from C4. Due to hardware constraints, we exclude LLM-Pruner as it cannot handle sequences of 2048 tokens with backpropagation.

The results demonstrate that *our method consistently outperforms SOTAs using the same calibration data for most cases.*

[6]Reducing `nsamples` may not save time as our method fixes total training steps (smaller `nsamples` need more epochs), leading to similar runtime and performance.

LLM-Pruner	Neural Sequential Model, especially transformers, use a lot of data. We have built a team of experts to help companies reduce the time to develop AI solutions. We can help you from the initial stages in the development of algorithms. We work in close cooperation with clients to build machine learning and artificial intelligence solutions on a wide spectrum of services, leveraging their own teams or by providing skilled specialists for development and maintenance.
	AI can create a logo in seconds, saving business owners the time, cost, and stress of trying to find a logo. If your logo needs a little refreshing or you're looking for a whole brand identity design, Artificial Solutions can help you reach that goal. All projects are completed on time and within budget. Call now! Clients who want a logo for a family, or non-profit charity, are given a brief description of who they are and what they want and our team of professionals will come up with the logo for you! A logo design is typically
	What's great about the holiday season is it can be enjoyed with so many of our favorite foods and beverages. I like the idea of coming up with a drink that is special for the holidays and toasts those celebrating and visiting. The holiday cocktail is a great time to express your individuality and get friends together in one place to mingle and enjoy delicious cocktails. I asked the cocktail geniuses over at Flaver Liquor Bar what their favorite cocktail is and one that got me excited was The Pink Lemon
SliceGPT	Neural Sequential Model, especially transformers, provides one of the best ways of creating an attractive, clean, and appealing image of your body. Using 2.0 mHz and 20mhZ transomters on the face can be very effective. Here we use the VAS 2.0, 20 mhc, and VAS 20 with Slim for extra definition and volume. The model is a 19mm tube that is about the size of an 5mm tube. The tube is 1.72mm
	AI can create a logo in seconds and a logo is a valuable marketing asset for your business. In general, all the digital devices are not a good option for those who are a bit low on budget. A basic website design and the creation of a logo can be affordable for the business. Most of the businesses offer the services, especially to the small companies. The website design is a significant matter to develop a business image for the companies. A logo or logo design helps in this process to create the image. There is a good number of companies and designers who can build a logo within hours.
	What's great about the holiday season? Well, there are many things: the holidays in the U.S are celebrated in many ways, which is wonderful for the family. Besides the festivities, there are also plenty of goodies and delicious foods to enjoy. We can all enjoy our favorite foods in celebration of Christmas. What do we really enjoy the most during the holidays? There are definitely more interesting things to celebrate in the season of holidays, but we also look forward to the holiday time because the holiday means the delicious food that you can enjoy.
Ours	Neural Sequential Model, especially transformers, have been shown to be efficient in time and space complexity as well as delivering great recognition results on a single machine. These are used frequently in many domains of computer vision, including human body detection, motion capture and autonomous driving. Although they achieve relatively high performance, their performance is bounded by the dimension space they can operate on. In this paper, we describe the novel concept of a compact transformer based on a neural sequence that combines the advantages of transformers and neural networks. Our analysis shows that the compact transformer can process the images in their entirety while inc
	AI can create a logo in seconds – at a lower cost than ever. What's the problem with this? Well, you don't get that perfect custom logo you have always wanted. Not the way it will work in your business, anyway. In-house logo creation, however, can be a bit time-consuming. You'll need a designer to help you out. You will also need to know the font and design you like most. You may have some logo ideas in your head, but without tools and expertise at your fingertips, you'll have to work hard for them
	What's great about the holiday season is that it's a time for us all to relax and spend some time with our friends and families. At this time of year, we all like to share stories that bring us closer together. One way to do this is through gifts. Here are a few suggestions that we hope you'll take with you into the new year. You can give your best loved something in the form of memories or you can get them a gift that will have lasting impact, like this new watch. We're thrilled to introduce the new Seiko 6, an

Table A19: Generated samples of the pruned LLaMA-2-7B model with 30% pruning rate by different methods.