Numerical Pruning for Efficient Autoregressive Models

Xuan Shen[1], Zhao Song[2], Yufa Zhou[3], Bo Chen[4], Jing Liu[5], Ruiyi Zhang[2], Ryan A. Rossi[2], Hao Tan[2], Tong Yu[2], Xiang Chen[2], Yufan Zhou[2], Tong Sun[2], Pu Zhao[1], Yanzhi Wang[1], Jiuxiang Gu[2]

[1]Northeastern University, [2]Adobe Research, [3]University of Pennsylvania,
[4]Middle Tennessee State University, [5]Monash University

Abstract

Transformers have emerged as the leading architecture in deep learning, proving to be versatile and highly effective across diverse domains beyond language and image processing. However, their impressive performance often incurs high computational costs due to their substantial model size. This paper focuses on compressing decoder-only transformer-based autoregressive models through structural weight pruning to improve the model efficiency while preserving performance for both language and image generation tasks. Specifically, we propose a training-free pruning method that calculates a numerical score with Newton's method for the Attention and MLP modules, respectively. Besides, we further propose another compensation algorithm to recover the pruned model for better performance. To verify the effectiveness of our method, we provide both theoretical support and extensive experiments. Our experiments show that our method achieves state-of-the-art performance with reduced memory usage and faster generation speeds on GPUs.

1 Introduction

Transformers have been extensively utilized across various domains and have become particularly dominant in the design of generative models. This includes Large Language Models (LLMs) (Vaswani et al. 2017; Touvron et al. 2023b) for language generation, as well as recent autoregressive image generation models (Van Den Oord, Vinyals et al. 2017; Esser, Rombach, and Ommer 2021; Ramesh et al. 2021; Yu et al. 2022). Notably, models such as LlamaGen (Sun et al. 2024), which use image tokenizers to convert continuous images into discrete tokens, have demonstrated the ability to surpass diffusion models (Ho, Jain, and Abbeel 2020; Rombach et al. 2022) in image generation tasks. The "*next-token prediction*" paradigm demonstrates significant capabilities in addressing both language and image generation tasks, enabling solutions that mimic human-like conversational interactions (Achiam, Adler et al. 2023; Li et al. 2024a).

Recognizing the capabilities of large autoregressive models with extensive parameters in generative tasks, pioneering works (Frantar and Alistarh 2023; Sun et al. 2023; Ma, Fang, and Wang 2023; Ashkboos et al. 2024; Zhan et al. 2021; Zhao et al. 2024; Zhan et al. 2024b) have sought to compress these models to enhance their execution efficiency. Compared to irregular pruning methods, structural pruning offers a more efficient reduction in both computational and memory overhead (Jian et al. 2021; Gong et al. 2022a,b, 2023). By maintaining a consistent and regular structure, it simplifies implementation, accelerates processing, and leads to more predictable resource savings (Kong et al. 2022, 2023), making it particularly advantageous for large-scale models and deployment on hardware devices. However, most of these efforts focus solely on language models and language-related research areas. Consequently, their methods are not readily applicable to image generation tasks because of the fundamental differences in data structure and computational requirements between language and image processing (Reed et al. 2016; Parmar et al. 2018; Lee et al. 2022; Shen et al. 2024a,c,b, 2023b,a). Therefore, it is crucial to explore the transformer architecture itself, rather than focusing on specific application models. This motivates us to develop a general method for compressing autoregressive models applicable to multiple kinds of generative tasks.

In addition to pruning the models, the recovery and optimization of pruned models are also crucial considerations. Full-parameter retraining of large autoregressive models after pruning is often computationally prohibitive, making calibrations with a few samples a preferred approach. Previous work (Frantar and Alistarh 2023) employs the Optimal Brain Surgeon (OBS) technique (Hassibi, Stork, and Wolff 1993; LeCun, Denker, and Solla 1989) for weight updates during pruning. However, its heavy reliance on the approximation information increases sensitivity to noise and reduces robustness across different datasets. SliceGPT (Ashkboos et al. 2024) relies on a large number of samples for pruning and calibration, leading to overfitting on calibration data and limiting the generalization to other different datasets.

In this work, we present a novel structural pruning approach that leverages our proposed numerical score, combined with compensation techniques for performance recovery. We first calculate the numerical score for each layer through solving the optimal pruning mask for the minimization of pruning errors using the Newton's method. By ranking these numerical scores of all layers, we generate the globally pruning mask with the specified pruning ratio. Additionally, we introduce a compensation algorithm to recover pruned models by updating the remaining weights to account for the loss caused by the pruned weights. We empirically evaluate our method using the LLaMA model family

including LLaMA, LLaMA-2, and LLaMA-3 as representative LLMs and LlamaGen for image generation tasks. Experimental results show that our method outperforms other state-of-the-art approaches in both language and image generation tasks, validating the effectiveness of our proposed numerical score and compensation algorithm. Moreover, our method reduces GPU memory usage and accelerates generation without requiring any additional GPU-specific modifications. Our main contributions are summarized as follows,

- We propose a numerical score, derived from the numerical solution of the optimal mask for minimizing pruning errors with Newton's method.
- We propose a compensation algorithm for the reconstruction of the pruned model, further enhancing the task performance of the pruned model.
- Experimental results show that our method not only achieves state-of-the-art performance but also reduces memory usage and accelerates generation on GPUs.

2 Related Work

2.1 Compression for LLMs

The large number of parameters in LLMs motivates the need for pruning (Gong et al. 2020; Wu et al. 2022; Zhan et al. 2024a; Li et al. 2022; Zhang et al. 2022; Zhan et al. 2024c; Shen et al. 2024d) to improve efficiency. The work (Frantar and Alistarh 2023) introduces the Optimal Brain Surgeon (OBS) method (Hassibi, Stork, and Wolff 1993; LeCun, Denker, and Solla 1989) to compress the LLMs, which removes weights with minimal impact on the loss function. It then updates the remaining weights by utilizing the inverse of the Hessian matrix to mitigate errors caused by the pruning process. Unfortunately, this kind of pruning method is still irregular, meaning it does not lead to significant reductions in memory and computational requirements. Subsequent works, such as LLM-Pruner (Ma, Fang, and Wang 2023), SliceGPT (Ashkboos et al. 2024), and FLAP (An et al. 2023), propose structural pruning methods that effectively reduce memory usage and accelerate inference on GPUs. These methods offer significant advantages over irregular pruning by directly enhancing the utility and efficiency of the models. While autoregressive models excel in sequential data processing, such as text, the distinct nature of image data, where spatial relationships and pixel-level details are critical, demands different approaches. As a result, adapting these models to image generation introduces complexities that limit their scalability and effectiveness.

2.2 Autoregressive Models in Image Generation

Autoregressive models, initially renowned for their success with LLMs, have recently gained popularity in the image generation research area. Pioneering works (Van Den Oord, Vinyals et al. 2017; Esser, Rombach, and Ommer 2021) introduced image tokenizers that convert continuous images into discrete tokens. These tokenizers, which have been demonstrated to be effective by the following works (Ramesh et al. 2021; Yu et al. 2021, 2022), enable autoregressive models to generate image tokens using the next-token prediction approach. Recent work (Sun et al. 2024) delivers a series of image generation models with a new constructed image tokenizer. This research demonstrates the effectiveness of LLM frameworks in image generation tasks, validating their potential beyond traditional language applications. Additionally, the work (Li et al. 2024a) delves deeper into the continuous-valued domains of autoregressive models and removes the image tokenizers for image generation tasks. This work achieves stronger results while leveraging the speed advantage of sequence modeling, which further enhances the utilization and demonstrates the potential of autoregressive models in image generation tasks.

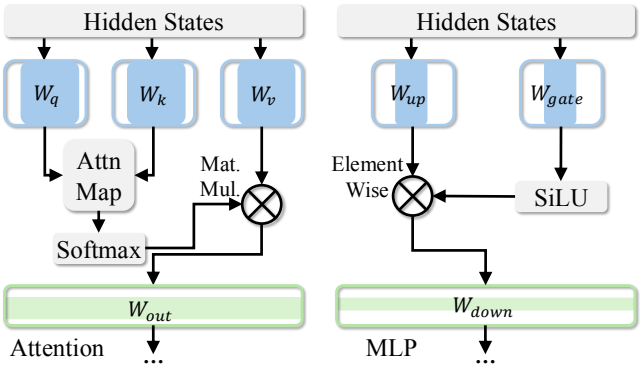

Figure 1: Pruning overview. Blue modules denote column pruning and green modules denote row pruning.

3 Methodology

3.1 Preliminary

Notations. We use $\mathbb{E}[\cdot]$ to denote the expectation. For two vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$, we use $\langle x, y \rangle$ to denote the inner product between x, y, *i.e.*, $\langle x, y \rangle = \sum_{i=1}^n x_i y_i$. We use $\mathbf{1}_n$ to denote a length-n vector where all the entries are ones. We use $x_{i,j}$ to denote the j-th coordinate of $x_i \in \mathbb{R}^n$. We use $\|x\|_p$ to denote the ℓ_p norm of a vector $x \in \mathbb{R}^n$. For each $a, b \in \mathbb{R}^n$, we use $a \circ b \in \mathbb{R}^n$ to denote the vector where i-th entry is $(a \circ b)_i = a_i b_i$ for all $i \in [n]$. We use $\|A\|$ to denote the spectral norm for matrix A. For a square matrix A, we use $\mathrm{tr}[A]$ to denote the trace of A, *i.e.*, $\mathrm{tr}[A] = \sum_{i=1}^n A_{i,i}$.

Internal Computation Alignment. To maintain model interpretability and mitigate the risk of model drift, we ensure consistency in the internal computations of the Attention module. This approach is aligned with established methodologies in the literature (Vaswani et al. 2017; Yang et al. 2020). Considering the definition in this paper mainly focuses on $X \cdot W$ where $X \in \mathbb{R}^{N \times D}$ and $W \in \mathbb{R}^{D \times D'}$, we visualize the pruning strategy in Figure 1. In detail, we utilize the identical pruning mask (*i.e.*, pruning strategy) for the columns of weights associated with the query, key, and value, as well as for the rows of weights in the output projection. Meanwhile, we apply the same strategy to the MLP module, using column pruning for the up and gate weights, and row pruning for the down projection weights. In this paper, we construct structural pruning metrics focusing on the output projection layers of Attention module and the down projection layers of MLP module.

3.2 Numerical Score

We define the weight as $W \in \mathbb{R}^{D \times D'}$, input as $X \in \mathbb{R}^{N \times D}$, and we denote the mask as $M \in \{0,1\}^D$. Additionally, we define the pruning ratio $\rho \in [0,1]$ as the ratio of the number of zeros to the total number of entries in pruning mask M.

Note that, when we apply the mask column by column, the mask M is a D-dimensional vector. Specifically, if $M_j = 0$ for $j \in [D]$, we prune the entire row for W, *i.e.*, $W_j = 0$, and if $M_j = 1$ we keep the original W_j.

To compute the numerical score, we explore the bound of the error (*i.e.*, difference) between the original weights and pruned weights. For the bound of the error, we first formulate the error for $i \in [D']$ as

$$\|XW_{*,i} - X(M \circ W_{*,i})\|_2. \tag{1}$$

Simplify further, for $i \in [D']$, Eq. (1) can be transformed into the following,

$$\|X((\mathbf{1}_D - M) \circ W_{*,i})\|_2.$$

In the above equation, the $\|\mathbf{1}_D - M\|_2$ denotes the number of zero entries in M, which is corresponding to the simply $\rho \cdot D$. Furthermore, assuming $\|X\| \leq R$, we demonstrate that the following Lemma 1 holds,

Lemma 1 (informal version of Lemma 9 at Appendix D.2)**.** *We show that for $i \in [D']$ we have*

$$\|XW_{*,i} - X(M \circ W_{*,i})\|_2 \leq \rho R \|W_{*,i}\|_2.$$

It is intuitive for us to minimize the error after establishing the error bound in Lemma 1. Thus, we examine each term. In the error bound of Lemma 1, the pruning ratio ρ is manually specified. We adopt the normalization for the input X, then the norm of normalized X is upper bounded by 1. Meanwhile, for $\|W_{*,i}\|_2$ term, it is the ℓ_2 norm for i-th column of weight W.

In order to minimize the error, we regulate both ρ and $\|W_{*,i}\|$. Then, we generalize the mask M from binary value to real value for the calculation of the numerical score. Meanwhile, we set one threshold which converts the real-valued mask back into a binary mask. For mask $M \in [0,1]^D$ and pruning ratio $\rho \in [0,1]$, the calculation of the numerical score is formulated as follows,

$$\begin{aligned} \arg\min_{M} \quad & \sum_{i \in [D']} \|XW_{*,i} - X(M \circ W_{*,i})\|_2, \\ \text{s.t.} \quad & \langle \mathbf{1}_D, M \rangle = (1-\rho)D. \end{aligned} \tag{2}$$

To better solve Eq. (2), we define the numerical score $z \in [0,1]^D$ and $r := (1-\rho)D \in [0, D]$. The equality constraint in Eq. (2) is then equivalent to $\langle \mathbf{1}_D, z \rangle - r = 0$.

Then, Eq. (2) becomes the minimization problem with the equality constraint. To efficiently solve such problem, we adopt the Newton' method (Bubeck et al. 2015). By turning the equality constraint into a penalty term for regularization, we further generate the following equivalent problem,

$$\begin{aligned} \arg\min_{z \in [0,1]^D} \quad & \frac{1}{2} \sum_{i \in [D']} \|XW_{*,i} - X(z \circ W_{*,i})\|_2^2 \\ & + \frac{1}{2} \lambda \cdot (\langle \mathbf{1}_D, z \rangle - r)^2, \end{aligned} \tag{3}$$

Algorithm 1: Numerical Score with Newton's Method

procedure NUMERICALSCORE($X \in \mathbb{R}^{N \times D}, W \in \mathbb{R}^{D \times D'}, r \in [0, D], \lambda \in \mathbb{R}_+, T \in \mathbb{N}_+$) ▷ Theorem 2
  We choose the initial point z_0 such that $z_0 \in [0,1]^D$
  for $t = 0 \to T$ **do**
    $g_l \leftarrow ((WW^\top) \circ (X^\top X))(z - \mathbf{1}_D)$
    $g_r \leftarrow \lambda(\langle \mathbf{1}_D, z \rangle - r) \cdot \mathbf{1}_D$
    $g \leftarrow g_l + g_r \in \mathbb{R}^D$
    $H_l \leftarrow (WW^\top) \circ (X^\top X)$
    $H_r \leftarrow \lambda \cdot \mathbf{1}_{D \times D}$
    $H \leftarrow H_l + H_r \in \mathbb{R}^{D \times D}$
    $z_{t+1} \leftarrow z_t - H^{-1} g$
  end for
  $z \leftarrow z_{T+1}$
  return z
end procedure

where $\lambda \in \mathbb{R}_+$ is the regularization parameter.

To explain how we solve this, we define the loss function for $i \in [D']$ as follows,

$$L(z)_i = \frac{1}{2} \|XW_{*,i} - X(z \circ W_{*,i})\|_2^2. \tag{4}$$

Meanwhile, for regularization term, we define as follows,

$$L_{\mathrm{reg}}(z) = \frac{1}{2} \lambda \cdot (\langle \mathbf{1}_D, z \rangle - r)^2. \tag{5}$$

Combining Lemma 12 and Lemma 13 at Appendix D.3, we compute the gradient of Eq. (4) and Eq. (5) as follows,

$$\begin{aligned} g = & \underbrace{((WW^\top) \circ (X^\top X))(z - \mathbf{1}_D)}_{\text{Gradient of } L(z)} \\ & + \underbrace{\lambda(\langle \mathbf{1}_D, z \rangle - r) \cdot \mathbf{1}_D}_{\text{Gradient of } L_{\mathrm{reg}}(z)}. \end{aligned} \tag{6}$$

Combining Lemma 14 and Lemma 15 at Appendix D.3, we compute the Hessian of Eq. (4) and Eq. (5) as follows,

$$H = \underbrace{(WW^\top) \circ (X^\top X)}_{\text{Hessian of } L(z)} + \underbrace{\lambda \cdot \mathbf{1}_{D \times D}}_{\text{Hessian of } L_{\mathrm{reg}}(z)}. \tag{7}$$

Subsequently, using Algorithm 1, we efficiently compute the optimal numerical z in $O(TD^3)$, where T represents the number of iterations for Newton's Method, typically around 50 in practice. Besides, we derive the following Theorem 2.

Theorem 2 (Mask optimization, informal version of Theorem 10 at Appendix D.3)**.** *If the following conditions hold:*

- *Let $W \in \mathbb{R}^{D \times D'}$, $X \in \mathbb{R}^{N \times D}$.*
- *Let $z \in [0,1]^D$.*
- *Let $r \in [0, D]$ denote the number of ones (it can be a fractional number).*
- *Let $\lambda > 0$ denote a regularization co-efficients.*
- *Assume $\|X\| \leq R$.*

There exists an algorithm (Algorithm 1) that can get the optimal z in $O(TD^3)$ for Eq. (3)*.*

3.3 Global Pruning

To ensure consistent head-level computation in the Attention module, given numerical scores $z^{\text{attn}} \in \mathbb{R}^D$, we group the scores of channels for h-th head to determine the importance score z_h^{head} of individual heads as follows,

$$z_h^{\text{head}} = \frac{1}{D_h} \cdot \sum_{i=h\cdot D_h}^{(h+1)\cdot D_h} z_i^{\text{attn}},$$

where D_h denotes the dimension of each head, $h \in [H]$ is the head index.

Unlike the Attention module, where heads work in parallel to capture various aspects of the input and their outputs are interdependent, the MLP module has a simpler structure with minimal interdependencies between its components. Thus, we retain the channel scores $z^{\text{mlp}} \in \mathbb{R}^D$ to guide the pruning process for MLP module.

To ensure a balanced pruning process that reflects the relative importance of each layer, we simultaneously sort the numerical scores across all layers to derive the globally pruning mask. Since a single head in the Attention module is evaluated with one score but contains significantly more weights than a single channel in the MLP module, we apply scaling factors based on the model design to balance number of pruned parameters between the Attention heads and MLP channels. For a specified global pruning ratio ρ, hidden state dimension D, head dimension D_h in the Attention module, and intermediate size D_{inter} in the MLP module, we define Ψ as the set that stores the scores for the whole model, then apply the scaling factor α when generating the threshold η for all the scores as follows,

$$\Psi := \alpha \cdot (\{\{z_{h,l}^{\text{head}}\}_{h=1}^{H}\}_{l=1}^{L} \bigcup \{\{z_{i,l}^{\text{mlp}}\}_{i=1}^{D_{\text{inter}}}\}_{l=1}^{L}, \quad (8)$$

$$\eta = \text{sort}(\Psi)[(\rho(L \cdot H + L \cdot D_{\text{inter}}))],\ \alpha = \frac{4D_h}{3},$$

where $H = \text{index}(D/D_h)$ denotes the number of head in Attention module, L denotes the number of layers for the whole model. Since the Attention module involves pruning 4 linear projections (query, key, value, and output) in head level, while the MLP module prunes only 3 (up, gate, and down) in channel level, the scaling factor α is given by $\frac{4D_h}{3}$.

When the threshold η is determined, we prune the heads in the Attention module and the channels in the MLP module across all layers based on the strategy that removes heads or channels with numerical scores below the threshold.

3.4 Compensation for Pruning

With the above discussion, we obtain the pruning mask with Newton's method. To further improve the model performance, we modify the remaining weights in the model to compensate the loss of the pruned weights.

Problem Formulation. Note that to align the internal computations in the attention and MLP modules, we prune the rows of the output layers in the modules and the columns in other layers of the modules. If the columns of a layer with W is pruned in XW, the corresponding columns of the output also become zero and we are not able to compensate its loss, since modifying other unpruned columns can not change the zero output for the pruned columns. Thus, we only update the weights of the output layers with row pruning in the Attention and MLP modules. We modify the remaining rows based on pruned rows in W. For layers with column pruning, we do not modify their unpruned weights.

For the original weights W, after pruning, there are k pruned rows which are all zeros and their row indexes are denoted by $p_i, \forall i \in [k]$. We modify the weights with the weight perturbations δW, so that the layer output difference (before and after pruning) measured with ℓ_2 norm is minimized. The weight optimization problem can formulated as the following,

$$\begin{aligned} \min_{\delta W} \quad & \mathcal{L}(\delta W) = \|X(W+\delta W) - XW\|_2^2 = \|X\delta W\|_2^2, \\ \text{s.t.} \quad & e_{p_i}^\top \delta W + (W)_{p_i,*} = 0, \quad \text{for } i = 1, 2, \ldots, k. \end{aligned} \quad (9)$$

where $e_{p_i} \in \mathbb{R}^{D\times 1}$ is the one-hot vector with the p_i^{th} element as 1 and all others as 0. Thus, $e_{p_i}^\top \delta W$ denotes selecting the p_i^{th} row of δW. $(W)_{i,j}$ represents the element in the i^{th} row and j^{th} column of the matrix. Then, $(W)_{p_i,*}$ represents the p_i^{th} row of W. We can see that the constraint in Eq. (9) ensures that the corresponding pruned rows in the modified weights are all zeros, and the remaining weights are optimized to minimize the loss incurred by pruned rows.

It can be further transformed to the following,

$$\begin{aligned} \min_{\delta W} \quad & \mathcal{L}(\delta W) = \|X\delta W\|_2^2, \\ \text{s.t.} \quad & M_p^\top \delta W + W_p = 0, \end{aligned} \quad (10)$$

where $M_p \in \mathbb{R}^{D\times k}$ is the collection of all e_{p_i}, *i.e.*, $(M_p)_{*,i} = e_{p_i}$, or $(M_p^\top)_{i,*} = e_{p_i}^\top, \forall i \in [k]$. Similarly, W_p is a collection of all pruned rows in W with $(W_p)_{i,*} = (W)_{p_i,*}, \forall i \in [k]$. We have $W_p = M_p^\top W$.

Optimal Solution. Eq. (10) can be solved analytically with the following Theorem 3. The detailed proof is shown in Appendix B.

Theorem 3. *The optimal solution for Eq.* (10) *can be derived as the following,*

$$\delta W^* = -(2X^\top X)^{-1} M_p (M_p^\top (2X^\top X)^{-1} M_p)^{-1} M_p^\top W. \quad (11)$$

Remark 4. *The optimal loss of Problem* (10) *corresponding to the optimal weight perturbation can be expressed as*

$$L^* = \frac{1}{2} \sum_i (W^\top M_p (M_p^\top (2X^\top X)^{-1} M_p)^{-1} M_p^\top W)_{i,i}. \quad (12)$$

The sum in Eq. (12) *is computed over D' (the number of columns in W),* i.e.*, $i \in [D']$.*

Remark 5. *If the rank of $2X^\top X$ is not full so that the inversion $(2X^\top X)^{-1}$ is unavailable, we apply the dampening method to compute $(2X^\top X + \gamma \cdot I)^{-1}$ instead of $(2X^\top X)^{-1}$, with γ as the dampening ratio.*

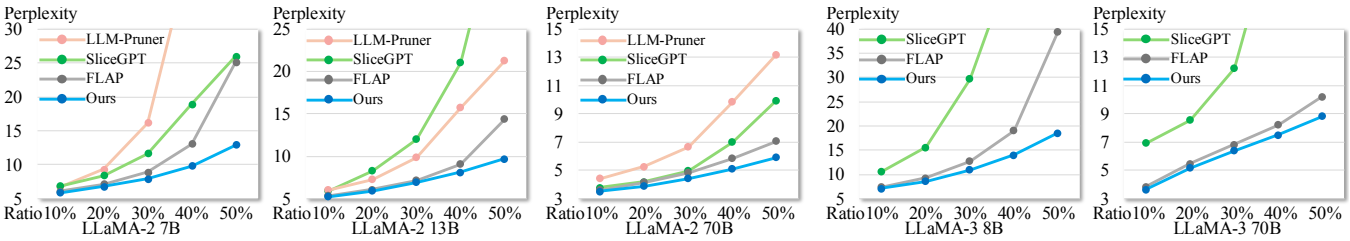

Figure 2: Perplexity (↓) results for LLaMA-2 and LLaMA-3 models on WikiText2 dataset with 2048 sequence length. Comprehensive detailed results are included in Table 6 and Table 7 at Appendix A.2 and A.3.

Method	Prune Ratio	LLaMA-7B			LLaMA-13B			LLaMA-30B			LLaMA-65B		
		Wiki	PTB	C4	Wiki	PTB	C4	Wiki	PTB	C4	Wiki	PTB	C4
Baseline	\	5.68	27.34	7.08	5.09	19.23	6.61	4.10	16.29	5.98	3.53	17.61	5.62
LLM-Pruner	10%	7.41	36.73	9.25	6.38	31.85	8.16	4.92	18.17	6.63	3.98	19.44	6.08
SliceGPT	10%	6.97	88.48	23.54	6.11	60.15	20.18	5.24	39.72	17.83	4.57	36.20	14.14
FLAP	10%	6.34	32.39	8.058	5.45	20.99	7.33	4.52	17.29	6.49	3.91	19.35	6.04
Ours	10%	**6.01**	**31.65**	**7.94**	**5.38**	**20.52**	**7.27**	**4.43**	**17.26**	**6.47**	**3.82**	**19.28**	**6.02**
LLM-Pruner	20%	10.73	59.73	12.15	6.38	31.85	9.42	5.83	20.18	7.55	4.65	21.85	6.75
SliceGPT	20%	8.42	120.89	35.93	7.17	86.26	29.70	6.18	50.95	26.85	5.34	61.09	21.86
FLAP	20%	7.40	36.77	9.99	6.03	23.33	8.42	5.18	19.30	7.42	4.45	21.45	6.75
Ours	20%	**6.60**	**35.75**	**9.49**	**5.89**	**23.11**	**8.39**	**4.92**	**18.58**	**7.36**	**4.26**	**20.94**	**6.73**
LLM-Pruner	30%	18.58	93.24	17.78	11.81	45.42	12.65	7.59	24.97	9.08	5.52	26.38	7.53
SliceGPT	30%	12.75	258.90	67.33	9.18	125.40	46.46	7.74	75.89	42.71	6.56	74.43	35.68
FLAP	30%	9.18	47.35	13.08	6.97	27.36	10.01	6.28	21.88	8.53	5.10	23.91	7.59
Ours	30%	**7.56**	**41.05**	**11.53**	**6.57**	**26.27**	**9.98**	**5.46**	**20.48**	**8.46**	**4.75**	**22.13**	**7.51**
LLM-Pruner	50%	126.0	460.7	73.88	45.69	152.99	36.94	19.68	78.29	18.64	9.34	43.79	12.16
SliceGPT	50%	1540	6364	4847	18.75	277.34	122.5	15.60	195.4	118.5	12.01	160.3	92.66
FLAP	50%	21.89	135.8	30.86	12.88	53.54	18.37	13.41	47.30	13.17	6.98	28.52	10.36
Ours	50%	**11.66**	**82.55**	**20.72**	**8.91**	**37.56**	**16.12**	**7.25**	**26.68**	**11.91**	**6.02**	**25.17**	**9.73**
LLM-Pruner	70%	9010	4111	2655	5900	6039	1334	895.7	3274	456.9	Nan	Nan	Nan
SliceGPT	70%	3605	7304	8096	67.65	874.9	537.4	71.25	633.1	406.6	102.4	863.9	662.8
FLAP	70%	577.9	1835	833.7	647.8	1588	975.1	2786	2735	2416	Nan	2333	Nan
Ours	70%	**162.9**	**721.3**	**361.6**	**41.66**	**275.7**	**115.3**	**39.88**	**124.2**	**50.43**	**9.65**	**69.49**	**20.84**

Table 1: Perplexity (↓) results for LLaMA-1 family models with different pruning ratios on WikiText2, PTB, and C4 with 2048 sequence length. Full results with larger sparsity ratios are included in Table 5 at Appendix A.1.

3.5 Complexity Analysis

For the computation of numerical score, according to the Lemma 1 and Theorem 2, the complexity is $O(TD^3)$ where T represents the number of iterations for Newton's Method, typically around 50 in practice. Additionally, for the compensation method, as demonstrated in Eq. (11), the complexity is $O(D^3)$ as we need to compute the inverse of a matrix. The matrix multiplication with M_p or M_p^T just selects the columns or rows of a matrix, without the need of actual multiplication. The complexity for numerical score calculation and compensation is the same with state-of-the-art methods, such as SparseGPT (Frantar and Alistarh 2023). In practice, the compensation is finished with just a few data samples on only the output projection layers of the Attention module and the down projection layers of the MLP module, which is more efficient compared with other recovery methods such as LLM-Pruner (Ma, Fang, and Wang 2023) to finetune the whole model on whole dataset, or SliceGPT (Ashkboos et al. 2024) to adopt a large amount of samples for calibration.

4 Experimental Results

4.1 Experiment Setup

We conduct the experiments on LLaMA model families including LLaMA-1 (Touvron et al. 2023a), LLaMA-2 (Touvron et al. 2023b), and LLaMA-3 (Meta 2024) for the language generation tasks. For evaluations, we compare the perplexity of the models on the WikiText2 (Merity et al. 2016), PTB (Marcus, Santorini, and Marcinkiewicz 1993), and C4 (Raffel et al. 2020) datasets with the 2048 sequence length. We also follow LLM-Pruner to evaluate the zero-shot accuracy on common sense reasoning zero-shot classification datasets including BoolQ (Clark et al. 2019a), PIQA (Bisk et al. 2020), HellaSwag (Zellers et al. 2019), WinoGrande (Sakaguchi et al. 2021), ARC-easy (Clark et al. 2018), ARC-challenge (Clark et al. 2018), and OpenbookQA (Mihaylov et al. 2018). For experiments, we adopt 128 samples from training dataset of WikiText2 to compute the numerical score and compensate the pruned models. For fairness, we also adopt 128 samples for other methods.

Method	Prune Ratio	BoolQ	PIQA	Hella Swag	Wino Grande	ARC-e	ARC-c	OBQA	Average Acc.
LLaMA-7B	/	73.18	78.35	72.99	67.01	67.45	41.38	42.40	63.25
LLM-Pruner(v)	20%	61.44	71.71	57.27	54.22	55.77	33.96	38.40	53.25
LLM-Pruner(e2)		59.39	75.57	65.34	61.33	59.18	37.12	39.80	56.82
LLM-Pruner(e1)		57.06	75.68	66.80	59.83	60.94	36.52	40.00	56.69
SliceGPT	20%	37.89	64.09	45.67	62.75	53.62	31.74	33.20	46.99
FLAP	20%	68.59	74.21	64.98	64.40	59.89	37.80	40.20	58.58
Ours	20%	67.92	74.76	67.31	66.54	58.80	36.77	39.4	**58.79**

Table 2: Pruning results for LLaMA-7B on common sense reasoning datasets. LLM-Pruner (v) and (ei) denote vector-wise and element-wise with i-th order ($i = 1, 2$). Full results with more sparsity ratios and LLaMA-13B are in Table 9 at Appendix B.

Figure 3: Visualization of generated images through LlamaGen-3B in 384×384 resolution (cfg=1.65) with 10% sparsity.

As for the image generation tasks, we adopt the LlamaGen (Sun et al. 2024) model family with LlamaGen-XXL and LlamaGen-3B to verify the effectiveness of our method on image generation tasks. We adopt the Fréchet inception distance (FID) (Heusel et al. 2017), Inception Score (IS) (Salimans et al. 2016), sFID (Nash et al. 2021), and Precision/Recall (Kynkäänniemi et al. 2019) as the evaluation metrics on ImageNet dataset (Deng et al. 2009). For all evaluations, we utilized ADM's TensorFlow scripts (Dhariwal and Nichol 2021) to ensure fair and consistent comparisons. Given that LLM-Pruner requires a backward process and SliceGPT has slow pruning, we further implement FLAP for comparative analysis in image generation tasks. In pratical, we generate 128 images for each class of ImageNet with LlamaGen models for the computation of numerical score and compensation. Same strategy for FLAP for fairness.

4.2 Results of LLMs

For the LLaMA models, we present the results with different pruning ratios varying from 10% to 70% in Table 1. Based on the perplexity results evaluated with 2048 sequence length on three datasets, our method consistently outperforms other methods across all pruning ratios, demonstrating the effectiveness of our proposed approach. Full results with more sparse ratios are included in Table 5 of Appendix A.1. Results show that for the larger model LLaMA-65B with pruning ratio of 70%, both LLM-Pruner and FLAP fail to produce an effective pruned model with their respective methods. In contrast, our method successfully maintains the most of the model's capabilities.

We further evaluate the zero-shot capabilities of the pruned model across seven downstream tasks. The results of LLaMA-7B model are shown in Table 2. Full results, including additional pruning ratios and the LLaMA-13B model, are detailed in Table 9 in Appendix A.5. Our method demonstrates superior performance compared to the other three methods on those common sense reasoning zero-shot classification datasets. Besides, we show the results with LLaMA and LLaMA-2 models of our method on MMLU (Hendrycks et al. 2021) and GSM8K (Cobbe et al. 2021) datasets in Table 8 of Appendix A.4, which demonstrates that our method retains both generative and mathematical capabilities.

We show the results for LLaMA-2 and LLaMA-3 models with 2048 sequence length on WikiText2 dataset in Figure 2. The detailed perplexity results for both model families on three datasets are shown in Table 6 and Table 7 of Appendix A.2 and A.3. The blue line representing our method's results consistently appears at the lowest position on the graphs, indicating its superior performance compared to the other methods with all model families.

Method	Ratio	FID ↓	sFID ↓	IS ↑	Prec ↑	Rec ↑
LlamaGen-XXL (cfg=1.75)						
/	/	2.39	6.02	253.16	80.73%	59.60%
FLAP	10%	7.87	9.92	145.25	61.96%	63.34%
Ours	10%	**6.09**	**7.70**	**168.96**	**70.98%**	**65.01%**
FLAP	15%	15.93	11.81	100.48	52.05%	62.02%
Ours	15%	**11.29**	**9.85**	**124.31**	**62.95%**	**65.84%**
FLAP	20%	53.86	20.63	32.41	28.31%	67.14%
Ours	20%	**22.45**	**14.16**	**78.64**	**53.68%**	**67.66%**
LlamaGen-3B (cfg=1.65)						
/	/	2.26	6.19	260.46	82.07%	58.35%
FLAP	10%	7.57	8.40	158.74	67.19%	64.90%
Ours	10%	**3.97**	**7.45**	**202.93**	**73.93%**	62.86%
FLAP	15%	38.45	23.61	57.29	43.45%	63.27%
Ours	15%	**8.92**	**10.32**	**152.36**	**66.83%**	**63.97%**
FLAP	20%	162.15	93.98	5.97	13.36%	28.03%
Ours	20%	**20.16**	**16.29**	**95.05**	**54.87%**	**63.72%**

Table 3: Sparse results for image generation task with LlamaGen model family in 384×384 resolution.

4.3 Results of Image Generation

We implement the FLAP pruning method on LlamaGen model and compare this method on image generation task. We show the sparse results with LlamaGen-XXL (1.4B) and LlamaGen-3B models on ImageNet with 384×384 resolution in Table 3. We observe that for the smaller model LlamaGen-XXL (1.4B), our method shows a distinct advantage at higher pruning ratios. For the larger model LlamaGen-3B, our method consistently outperforms across all pruning ratios, effectively preserving most of the original model's capabilities. We further visualize the images generated by 10% sparsity models in Figure 3 with additional visualizations provided in Figure 6 of Appendix A.6. We observe that our method generates better image results compared to FLAP method in most cases.

4.4 Ablation Study

Results with 128 Sequence Length. To demonstrate the effectiveness of our method for short sequence lengths, we present the results generated with a sequence length of 128 in Table 4 using the LLaMA-7B model and the WikiText2 dataset. Comprehensive results, including additional pruning ratios and datasets, are provided in Table 10 of Appendix A.7. As observed, our method consistently performs the best across all pruning ratios.

Number of Samples for Compensation. To verify the efficiency of the compensation process for our method, we conducted experiments using different numbers of samples. The results of these experiments are shown in Figure 4. The results demonstrate that the performance difference between compensation with 128 samples versus 512 or even 1024 samples is minimal across all pruning ratios. This indicates that 128 samples are sufficient for our compensation method, highlighting its efficiency.

Prune Ratio	10%	20%	30%	40%	50%
LLM-Pruner	15.37	19.09	30.64	52.28	122.8
SliceGPT	14.52	19.27	44.96	535.5	2241
FLAP	13.84	14.62	17.62	22.32	31.80
Ours	**13.31**	**14.47**	**16.40**	**19.04**	**23.32**

Table 4: Results for LLaMA-7B model on WikiText2 with 128 sequence length. Full results with more datasets are in Table 10 at Appendix B.

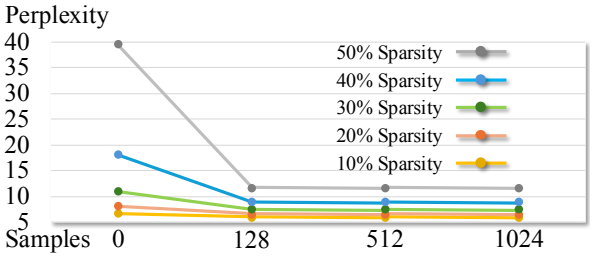

Figure 4: Ablation for number of samples for compensation.

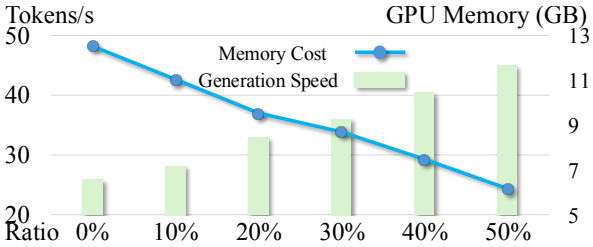

Figure 5: GPU memory v.s. generation speed.

Memory & Generation Speed. We show the memory reduction and generation acceleration in Figure 5. The results are obtained using an NVIDIA A100 GPU with a sentence consisting of 64 tokens as the model input. The results show that as the pruning ratio increases, there is a corresponding decrease in GPU memory usage and an increase in generation speed, which validates the effectiveness of our method.

5 Conclusion and Limitation

In this paper, we propose the numerical score which is calculated through Newton's Method for the minimization of pruning errors. We further sort numerical scores across all model layers for global pruning. Additionally, we introduce a compensation algorithm to reconstruct weights in pruned models. Experimental results show that our method achieves the state-of-the-art performance, which demonstrates the effectiveness of our method. Meanwhile, our method reduces memory usage and accelerates generation on GPUs without requiring additional implementations. One limitation of our method is its reduced effectiveness with smaller LlamaGen models for image generation tasks, primarily due to the usage of the discrete image tokenizer, which tends to lose important details as the sparsity increases.

References

Achiam, J.; Adler, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Alman, J.; Liang, J.; Song, Z.; Zhang, R.; and Zhuo, D. 2022. Bypass exponential time preprocessing: Fast neural network training via weight-data correlation preprocessing. *arXiv preprint arXiv:2211.14227*.

Alman, J.; and Song, Z. 2024. How to Capture Higher-order Correlations? Generalizing Matrix Softmax Attention to Kronecker Computation. In *The Twelfth International Conference on Learning Representations*.

Amini, A. A.; and Wainwright, M. J. 2009. High-dimensional analysis of semidefinite relaxations for sparse principal components. *The Annals of Statistics*, 37(5B).

An, Y.; Zhao, X.; Yu, T.; Tang, M.; and Wang, J. 2023. Fluctuation-based Adaptive Structured Pruning for Large Language Models. arXiv:2312.11983.

Anstreicher, K. M. 2000. The Volumetric Barrier for Semidefinite Programming. *Math. Oper. Res.*, 25(3): 365–380.

Arora, S.; Du, S.; Hu, W.; Li, Z.; and Wang, R. 2019a. Fine-grained analysis of optimization and generalization for over-parameterized two-layer neural networks. In *International Conference on Machine Learning*, 322–332. PMLR.

Arora, S.; Du, S. S.; Hu, W.; Li, Z.; Salakhutdinov, R. R.; and Wang, R. 2019b. On exact computation with an infinitely wide neural net. *Advances in neural information processing systems*, 32.

Ashkboos, S.; Croci, M. L.; Nascimento, M. G. d.; Hoefler, T.; and Hensman, J. 2024. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*.

Bartlett, P. L.; Bousquet, O.; and Mendelson, S. 2005. LOCAL RADEMACHER COMPLEXITIES. *The Annals of Statistics*, 33(4): 1497–1537.

Belinkov, Y. 2022. Probing Classifiers: Promises, Shortcomings, and Advances. *Computational Linguistics*, 48(1): 207–219.

Bian, S.; Song, Z.; and Yin, J. 2023. Federated Empirical Risk Minimization via Second-Order Method. *arXiv preprint arXiv:2305.17482*.

Bisk, Y.; Zellers, R.; Gao, J.; Choi, Y.; et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, 05, 7432–7439.

Bottou, L.; and Bousquet, O. 2007. The tradeoffs of large scale learning. *Advances in neural information processing systems*, 20.

Brand, J. v. d. 2020. A deterministic linear program solver in current matrix multiplication time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 259–278. SIAM.

Brand, J. v. d.; Lee, Y. T.; Sidford, A.; and Song, Z. 2020a. Solving tall dense linear programs in nearly linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, 775–788.

Brand, J. v. d.; Peng, B.; Song, Z.; and Weinstein, O. 2020b. Training (overparametrized) neural networks in near-linear time. *arXiv preprint arXiv:2006.11648*.

Brand, J. v. d.; Song, Z.; and Zhou, T. 2023. Algorithm and Hardness for Dynamic Attention Maintenance in Large Language Models. *arXiv preprint arXiv:2304.02207*.

Bubeck, S.; et al. 2015. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4): 231–357.

Cai, T.; Gao, R.; Hou, J.; Chen, S.; Wang, D.; He, D.; Zhang, Z.; and Wang, L. 2019. Gram-gauss-newton method: Learning overparameterized neural networks for regression problems. *arXiv preprint arXiv:1905.11675*.

Cao, Y.; and Gu, Q. 2019. Generalization bounds of stochastic gradient descent for wide and deep neural networks. *Advances in neural information processing systems*, 32.

Chang, C.-C.; and Lin, C.-J. 2001. Training v-support vector classifiers: theory and algorithms. *Neural computation*, 13(9): 2119–2147.

Chen, B.; Li, X.; Liang, Y.; Long, J.; Shi, Z.; and Song, Z. 2024a. Circuit Complexity Bounds for RoPE-based Transformer Architecture. *arXiv preprint arXiv:2411.07602*.

Chen, B.; Li, X.; Liang, Y.; Shi, Z.; and Song, Z. 2024b. Bypassing the Exponential Dependency: Looped Transformers Efficiently Learn In-context by Multi-step Gradient Descent. *arXiv preprint arXiv:2410.11268*.

Chen, B.; Liang, Y.; Sha, Z.; Shi, Z.; and Song, Z. 2024c. HSR-Enhanced Sparse Attention Acceleration. *arXiv preprint arXiv:2410.10165*.

Chu, T.; Song, Z.; and Yang, C. 2024. How to Protect Copyright Data in Optimization of Large Language Models? In *Proceedings of the AAAI Conference on Artificial Intelligence*, 16, 17871–17879.

Clark, C.; Lee, K.; Chang, M.-W.; Kwiatkowski, T.; Collins, M.; and Toutanova, K. 2019a. BoolQ: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Clark, K.; Khandelwal, U.; Levy, O.; and Manning, C. D. 2019b. What Does BERT Look At? An Analysis of BERT's Attention. arXiv:1906.04341.

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

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*.

Cohen, M. B.; Lee, Y. T.; and Song, Z. 2021. Solving linear programs in the current matrix multiplication time. *Journal of the ACM (JACM)*, 68(1): 1–39.

d'Aspremont, A.; Ghaoui, L. E.; Jordan, M. I.; and Lanckriet, G. R. G. 2006. A direct formulation for sparse PCA using semidefinite programming. arXiv:cs/0406021.

Défossez, A.; and Bach, F. 2014. Constant step size least-mean-square: Bias-variance trade-offs and optimal sampling distributions. *arXiv preprint arXiv:1412.0156*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

Deng, Y.; Li, Z.; and Song, Z. 2023. Attention scheme inspired softmax regression. *arXiv preprint arXiv:2304.10411*.

Deng, Y.; Mahadevan, S.; and Song, Z. 2023. Randomized and deterministic attention sparsification algorithms for over-parameterized feature dimension. *arXiv preprint arXiv:2304.04397*.

Deng, Y.; Song, Z.; Xie, S.; and Yang, C. 2023. Unmasking transformers: A theoretical approach to data recovery via attention weights. *arXiv preprint arXiv:2310.12462*.

Dhariwal, P.; and Nichol, A. Q. 2021. Diffusion Models Beat GANs on Image Synthesis. In Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*.

Diakonikolas, I.; Kamath, G.; Kane, D.; Li, J.; Moitra, A.; and Stewart, A. 2019. Robust Estimators in High Dimensions without the Computational Intractability. arXiv:1604.06443.

Dong, S.; Lee, Y. T.; and Ye, G. 2023. A Nearly-Linear Time Algorithm for Linear Programs with Small Treewidth: A Multiscale Representation of Robust Central Path. arXiv:2011.05365.

Dong, Y.; Hopkins, S. B.; and Li, J. 2019. Quantum Entropy Scoring for Fast Robust Mean Estimation and Improved Outlier Detection. arXiv:1906.11366.

Du, S. S.; Zhai, X.; Poczos, B.; and Singh, A. 2018. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*.

Esser, P.; Rombach, R.; and Ommer, B. 2021. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 12873–12883.

Feldman, V.; Guruswami, V.; Raghavendra, P.; and Wu, Y. 2012. Agnostic learning of monomials by halfspaces is hard. *SIAM Journal on Computing*, 41(6): 1558–1590.

Frantar, E.; and Alistarh, D. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *ICML*, 10323–10337. PMLR.

Frostig, R.; Ge, R.; Kakade, S. M.; and Sidford, A. 2015. Competing with the empirical risk minimizer in a single pass. In *Conference on learning theory*, 728–763. PMLR.

Gao, Y.; Mahadevan, S.; and Song, Z. 2023. An over-parameterized exponential regression. *arXiv preprint arXiv:2303.16504*.

Gao, Y.; Song, Z.; Wang, W.; and Yin, J. 2023a. A fast optimization view: Reformulating single layer attention in llm based on tensor and svm trick, and solving it in matrix multiplication time. *arXiv preprint arXiv:2309.07418*.

Gao, Y.; Song, Z.; Yang, X.; Zhang, R.; and Zhou, Y. 2023b. Fast Quantum Algorithm for Attention Computation. *arXiv preprint arXiv:2307.08045*.

Gao, Y.; Song, Z.; Yang, X.; and Zhou, Y. 2023c. Differentially private attention computation. *arXiv preprint arXiv:2305.04701*.

Gao, Y.; Song, Z.; and Yin, J. 2023. An iterative algorithm for rescaled hyperbolic functions regression. *arXiv preprint arXiv:2305.00660*.

Gao, Y.; Song, Z.; Zhang, R.; and Zhou, Y. 2024. Quantum Speedup for Spectral Approximation of Kronecker Products. *arXiv preprint arXiv:2402.07027*.

Gong, Y.; Yuan, G.; Zhan, Z.; Niu, W.; Li, Z.; Zhao, P.; Cai, Y.; Liu, S.; Ren, B.; Lin, X.; et al. 2022a. Automatic mapping of the best-suited dnn pruning schemes for real-time mobile acceleration. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 27(5): 1–26.

Gong, Y.; Zhan, Z.; Li, Z.; Niu, W.; Ma, X.; Wang, W.; Ren, B.; Ding, C.; Lin, X.; Xu, X.; et al. 2020. A privacy-preserving-oriented dnn pruning and mobile acceleration framework. In *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, 119–124.

Gong, Y.; Zhan, Z.; Zhao, P.; et al. 2022b. All-in-one: A highly representative dnn pruning framework for edge devices with dynamic power management. In *ICCAD*, 1–9.

Gong, Y.; Zhao, P.; Zhan, Z.; et al. 2023. Condense: A Framework for Device and Frequency Adaptive Neural Network Models on the Edge. In *DAC*, 1–6. IEEE.

Gu, J.; Li, C.; Liang, Y.; Shi, Z.; and Song, Z. 2024a. Exploring the frontiers of softmax: Provable optimization, applications in diffusion model, and beyond. *arXiv preprint arXiv:2405.03251*.

Gu, J.; Li, C.; Liang, Y.; Shi, Z.; Song, Z.; and Zhou, T. 2024b. Fourier circuits in neural networks: Unlocking the potential of large language models in mathematical reasoning and modular arithmetic. *arXiv preprint arXiv:2402.09469*.

Gu, J.; Liang, Y.; Sha, Z.; Shi, Z.; and Song, Z. 2024c. Differential Privacy Mechanisms in Neural Tangent Kernel Regression. *arXiv preprint arXiv:2407.13621*.

Gu, Y.; and Song, Z. 2022. A Faster Small Treewidth SDP Solver. arXiv:2211.06033.

Gu, Y.; Song, Z.; and Zhang, L. 2023. A nearly-linear time algorithm for structured support vector machines. *arXiv preprint arXiv:2307.07735*.

Hassibi, B.; Stork, D.; and Wolff, G. 1993. Optimal Brain Surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, 293–299 vol.1.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021. Measuring Massive Multitask Language Understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Hewitt, J.; and Liang, P. 2019. Designing and Interpreting Probes with Control Tasks. arXiv:1909.03368.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33: 6840–6851.

Huang, B.; Jiang, S.; Song, Z.; Tao, R.; and Zhang, R. 2021a. Solving SDP Faster: A Robust IPM Framework and Efficient Implementation. arXiv:2101.08208.

Huang, B.; Li, X.; Song, Z.; and Yang, X. 2021b. Fl-ntk: A neural tangent kernel-based framework for federated learning analysis. In *International Conference on Machine Learning*, 4423–4434. PMLR.

Jambulapati, A.; Li, J.; and Tian, K. 2020. Robust Sub-Gaussian Principal Component Analysis and Width-Independent Schatten Packing. arXiv:2006.06980.

Ji, Z.; and Telgarsky, M. 2019. Polylogarithmic width suffices for gradient descent to achieve arbitrarily small test error with shallow relu networks. *arXiv preprint arXiv:1909.12292*.

Jian, T.; Gong, Y.; Zhan, Z.; Shi, R.; Soltani, N.; Wang, Z.; Dy, J.; Chowdhury, K.; Wang, Y.; and Ioannidis, S. 2021. Radio frequency fingerprinting on the edge. *IEEE Transactions on Mobile Computing*, 21(11): 4078–4093.

Jiang, H.; Kathuria, T.; Lee, Y. T.; Padmanabhan, S.; and Song, Z. 2020a. A faster interior point method for semidefinite programming. In *2020 IEEE 61st annual symposium on foundations of computer science (FOCS)*, 910–918. IEEE.

Jiang, H.; Lee, Y. T.; Song, Z.; and wai Wong, S. C. 2020b. An Improved Cutting Plane Method for Convex Optimization, Convex-Concave Games and its Applications. arXiv:2004.04250.

Jiang, S.; Song, Z.; Weinstein, O.; and Zhang, H. 2020c. Faster dynamic matrix inverse for faster lps. *arXiv preprint arXiv:2004.07470*.

Jin, C.; Liu, L. T.; Ge, R.; and Jordan, M. I. 2018. On the local minima of the empirical risk. *Advances in neural information processing systems*, 31.

Joachims, T. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 217–226.

Johnson, R.; and Zhang, T. 2013. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26.

Kacham, P.; Mirrokni, V.; and Zhong, P. 2023. Polysketchformer: Fast transformers via sketches for polynomial kernels. *arXiv preprint arXiv:2310.01655*.

Kong, Z.; Dong, P.; Ma, X.; Meng, X.; Niu, W.; Sun, M.; Shen, X.; Yuan, G.; Ren, B.; Tang, H.; et al. 2022. Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In *ECCV*.

Kong, Z.; Ma, H.; Yuan, G.; Sun, M.; Xie, Y.; Dong, P.; Meng, X.; Shen, X.; Tang, H.; Qin, M.; et al. 2023. Peeling the onion: Hierarchical reduction of data redundancy for efficient vision transformer training. In *AAAI*.

Kynkäänniemi, T.; Karras, T.; Laine, S.; Lehtinen, J.; and Aila, T. 2019. Improved Precision and Recall Metric for Assessing Generative Models. *CoRR*, abs/1904.06991.

LeCun, Y.; Denker, J.; and Solla, S. 1989. Optimal Brain Damage. In Touretzky, D., ed., *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Lee, D.; Kim, C.; Kim, S.; Cho, M.; and Han, W.-S. 2022. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11523–11532.

Lee, J. D.; Shen, R.; Song, Z.; Wang, M.; et al. 2020. Generalized leverage score sampling for neural networks. *Advances in Neural Information Processing Systems*, 33: 10775–10787.

Lee, Y. T.; Song, Z.; and Zhang, Q. 2019. Solving empirical risk minimization in the current matrix multiplication time. In *Conference on Learning Theory*, 2140–2157. PMLR.

Li, S.; Song, Z.; Xia, Y.; Yu, T.; and Zhou, T. 2023a. The closeness of in-context learning and weight shifting for softmax regression. *arXiv preprint arXiv:2304.13276*.

Li, T.; Tian, Y.; Li, H.; Deng, M.; and He, K. 2024a. Autoregressive Image Generation without Vector Quantization. *arXiv preprint arXiv:2406.11838*.

Li, X.; Liang, Y.; Shi, Z.; Song, Z.; and Zhou, Y. 2024b. Fine-grained Attention I/O Complexity: Comprehensive Analysis for Backward Passes. *arXiv preprint arXiv:2410.09397*.

Li, X.; Long, J.; Song, Z.; and Zhou, T. 2024c. Fast Second-order Method for Neural Network under Small Treewidth Setting. In *2024 IEEE International Conference on Big Data (BigData)*. IEEE.

Li, Y.; and Liang, Y. 2018. Learning overparameterized neural networks via stochastic gradient descent on structured data. *Advances in neural information processing systems*, 31.

Li, Y.; Zhao, P.; Yuan, G.; Lin, X.; Wang, Y.; and Chen, X. 2022. Pruning-as-search: Efficient neural architecture search via channel pruning and structural reparameterization. *arXiv preprint arXiv:2206.01198*.

Li, Z.; Song, Z.; Wang, Z.; and Yin, J. 2023b. Local Convergence of Approximate Newton Method for Two Layer Nonlinear Regression. *arXiv preprint arXiv:2311.15390*.

Li, Z.; Song, Z.; and Zhou, T. 2023. Solving regularized exp, cosh and sinh regression problems. *arXiv preprint arXiv:2303.15725*.

Liang, Y.; Long, J.; Shi, Z.; Song, Z.; and Zhou, Y. 2024a. Beyond Linear Approximations: A Novel Pruning Approach for Attention Matrix. *arXiv preprint arXiv:2410.11261*.

Liang, Y.; Sha, Z.; Shi, Z.; Song, Z.; and Zhou, Y. 2024b. Looped ReLU MLPs May Be All You Need as Practical Programmable Computers. *arXiv preprint arXiv:2410.09375*.

Liang, Y.; Sha, Z.; Shi, Z.; Song, Z.; and Zhou, Y. 2024c. Multi-Layer Transformers Gradient Can be Approximated in Almost Linear Time. *arXiv preprint arXiv:2408.13233*.

Liang, Y.; Shi, Z.; Song, Z.; and Zhou, Y. 2024d. Differential Privacy of Cross-Attention with Provable Guarantee. *arXiv preprint arXiv:2407.14717*.

Liang, Y.; Shi, Z.; Song, Z.; and Zhou, Y. 2024e. Tensor Attention Training: Provably Efficient Learning of Higher-order Transformers. *arXiv preprint arXiv:2405.16411*.

Liang, Y.; Shi, Z.; Song, Z.; and Zhou, Y. 2024f. Unraveling the Smoothness Properties of Diffusion Models: A Gaussian Mixture Perspective. *arXiv preprint arXiv:2405.16418*.

Lianke, Q.; Song, Z.; Zhang, L.; and Zhuo, D. 2023. An online and unified algorithm for projection matrix vector multiplication with application to empirical risk minimization. In *International Conference on Artificial Intelligence and Statistics*, 101–156. PMLR.

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

Marcus, M. P.; Santorini, B.; and Marcinkiewicz, M. A. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 313–330.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2016. Pointer sentinel mixture models. *arXiv*.

Meta. 2024. LLaMA 3: The most capable openly available LLM to date. *https://ai.meta.com/blog/meta-llama-3/*.

Mihaylov, T.; Clark, P.; Khot, T.; and Sabharwal, A. 2018. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. In *EMNLP*, 2381–2391. Brussels, Belgium: ACL.

Moulines, E.; and Bach, F. 2011. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. *Advances in neural information processing systems*, 24.

Nash, C.; Menick, J.; Dieleman, S.; and Battaglia, P. W. 2021. Generating images with sparse representations. *arXiv preprint arXiv:2103.03841*.

Nemirovski, A.; Juditsky, A.; Lan, G.; and Shapiro, A. 2009. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4): 1574–1609.

Nesterov, Y. 1983. A method for solving the convex programming problem with convergence rate O (1/k2). In *Dokl akad nauk Sssr*, volume 269, 543.

Nesterov, Y. 2013. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media.

Oymak, S.; and Soltanolkotabi, M. 2020. Toward moderate overparameterization: Global convergence guarantees for training shallow neural networks. *IEEE Journal on Selected Areas in Information Theory*, 1(1): 84–105.

Parmar, N.; Vaswani, A.; Uszkoreit, J.; Kaiser, L.; Shazeer, N.; Ku, A.; and Tran, D. 2018. Image transformer. In *International conference on machine learning*, 4055–4064. PMLR.

Polyak, B. T.; and Juditsky, A. B. 1992. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4): 838–855.

Qin, L.; Song, Z.; and Sun, B. 2023. Is Solving Graph Neural Tangent Kernel Equivalent to Training Graph Neural Network? *arXiv preprint arXiv:2309.07452*.

Qin, L.; Song, Z.; and Yang, Y. 2023. Efficient SGD Neural Network Training via Sublinear Activated Neuron Identification. *arXiv preprint arXiv:2307.06565*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*.

Ramesh, A.; Pavlov, M.; Goh, G.; Gray, S.; Voss, C.; Radford, A.; Chen, M.; and Sutskever, I. 2021. Zero-shot text-to-image generation. In *International conference on machine learning*, 8821–8831. Pmlr.

Reed, S.; Akata, Z.; Yan, X.; Logeswaran, L.; Schiele, B.; and Lee, H. 2016. Generative adversarial text to image synthesis. In *ICML*, 1060–1069. PMLR.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *CVPR*, 10684–10695.

Sakaguchi, K.; Bras, R. L.; Bhagavatula, C.; and Choi, Y. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9): 99–106.

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; and Chen, X. 2016. Improved techniques for training gans. *NeurIPS*, 29.

Shalev-Shwartz, S.; and Zhang, T. 2013. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(1).

Shen, X.; Dong, P.; Lu, L.; et al. 2024a. Agile-Quant: Activation-Guided Quantization for Faster Inference of LLMs on the Edge. In *AAAI*.

Shen, X.; Han, Z.; Lu, L.; et al. 2024b. HotaQ: Hardware Oriented Token Adaptive Quantization for Large Language Models. *TCAD*.

Shen, X.; Kong, Z.; Qin, M.; et al. 2023a. Data level lottery ticket hypothesis for vision transformers. *IJCAI*.

Shen, X.; Kong, Z.; Yang, C.; et al. 2024c. EdgeQAT: Entropy and Distribution Guided Quantization-Aware Training for the Acceleration of Lightweight LLMs on the Edge. *arXiv preprint arXiv:2402.10787*.

Shen, X.; Wang, Y.; Lin, M.; et al. 2023b. DeepMAD: Mathematical Architecture Design for Deep Convolutional Neural Network. In *CVPR*.

Shen, X.; Zhao, P.; Gong, Y.; Kong, Z.; Zhan, Z.; Wu, Y.; Lin, M.; Wu, C.; Lin, X.; and Wang, Y. 2024d. Search for Efficient Large Language Models. In *NeurIPS*.

Shrivastava, A.; Song, Z.; and Xu, Z. 2023. A Theoretical Analysis Of Nearest Neighbor Search On Approximate Near Neighbor Graph. *arXiv preprint arXiv:2303.06210*.

Song, Z.; Wang, W.; and Yin, J. 2023. A unified scheme of resnet and softmax. *arXiv preprint arXiv:2309.13482*.

Song, Z.; Xu, G.; and Yin, J. 2023. The Expressibility of Polynomial based Attention Scheme. *arXiv preprint arXiv:2310.20051*.

Song, Z.; and Yang, X. 2019. Quadratic suffices for over-parametrization via matrix chernoff bound. *arXiv preprint arXiv:1906.03593*.

Song, Z.; Ye, M.; and Zhang, L. 2023. Streaming Semidefinite Programs: $O(\sqrt{n})$ Passes, Small Space and Fast Runtime. *arXiv preprint arXiv:2309.05135*.

Song, Z.; and Yu, Z. 2021. Oblivious sketching-based central path method for solving linear programming problems.

Song, Z.; Zhang, L.; and Zhang, R. 2021. Training multi-layer over-parametrized neural network in subquadratic time. *arXiv preprint arXiv:2112.07628*.

Sun, M.; Liu, Z.; Bair, A.; and Kolter, J. Z. 2023. A Simple and Effective Pruning Approach for Large Language Models. *arXiv preprint arXiv:2306.11695*.

Sun, P.; Jiang, Y.; Chen, S.; Zhang, S.; Peng, B.; Luo, P.; and Yuan, Z. 2024. Autoregressive Model Beats Diffusion: Llama for Scalable Image Generation. *arXiv preprint arXiv:2406.06525*.

Tarzanagh, D. A.; Li, Y.; Thrampoulidis, C.; and Oymak, S. 2023. Transformers as support vector machines. *arXiv preprint arXiv:2308.16898*.

Tenney, I.; Das, D.; and Pavlick, E. 2019. BERT Rediscovers the Classical NLP Pipeline. arXiv:1905.05950.

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

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

van den Brand, J. 2020. Unifying Matrix Data Structures: Simplifying and Speeding up Iterative Algorithms. arXiv:2010.13888.

Van Den Oord, A.; Vinyals, O.; et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Vapnik, V. 1991. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4.

Vapnik, V. 2013. *The nature of statistical learning theory*. Springer science & business media.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Vig, J.; and Belinkov, Y. 2019. Analyzing the Structure of Attention in a Transformer Language Model. arXiv:1906.04284.

Wu, Y.; Gong, Y.; Zhao, P.; et al. 2022. Compiler-aware neural architecture search for on-mobile real-time super-resolution. In *ECCV*, 92–111. Springer.

Yang, Y.-Y.; Rashtchian, C.; Zhang, H.; Salakhutdinov, R. R.; and Chaudhuri, K. 2020. A Closer Look at Accuracy vs. Robustness. In *NeurIPS*, volume 33, 8588–8601. Curran Associates, Inc.

Yu, J.; Li, X.; Koh, J. Y.; Zhang, H.; Pang, R.; Qin, J.; Ku, A.; Xu, Y.; Baldridge, J.; and Wu, Y. 2021. Vector-quantized image modeling with improved vqgan. *arXiv preprint arXiv:2110.04627*.

Yu, J.; Xu, Y.; Koh, J. Y.; Luong, T.; Baid, G.; Wang, Z.; Vasudevan, V.; Ku, A.; Yang, Y.; Ayan, B. K.; et al. 2022. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2(3): 5.

Zellers, R.; Holtzman, A.; Bisk, Y.; Farhadi, A.; and Choi, Y. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Zhan, Z.; Gong, Y.; Zhao, P.; Yuan, G.; et al. 2021. Achieving on-mobile real-time super-resolution with neural architecture and pruning search. In *ICCV*, 4821–4831.

Zhan, Z.; Kong, Z.; Gong, Y.; et al. 2024a. Exploring Token Pruning in Vision State Space Models. In *NeurIPS*.

Zhan, Z.; Wu, Y.; Gong, Y.; et al. 2024b. Fast and Memory-Efficient Video Diffusion Using Streamlined Inference. In *NeurIPS*.

Zhan, Z.; Wu, Y.; Kong, Z.; et al. 2024c. Rethinking Token Reduction for State Space Models. In *EMNLP*, 1686–1697. Miami, Florida, USA: ACL.

Zhang, G.; Martens, J.; and Grosse, R. B. 2019. Fast convergence of natural gradient descent for over-parameterized neural networks. *Advances in Neural Information Processing Systems*, 32.

Zhang, J.; Karimireddy, S. P.; Veit, A.; Kim, S.; Reddi, S.; Kumar, S.; and Sra, S. 2020a. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33: 15383–15393.

Zhang, L. 2022. *Speeding up optimizations via data structures: Faster search, sample and maintenance*. Ph.D. thesis, Master's thesis, Carnegie Mellon University.

Zhang, Y.; Plevrakis, O.; Du, S. S.; Li, X.; Song, Z.; and Arora, S. 2020b. Over-parameterized adversarial training: An analysis overcoming the curse of dimensionality. *Advances in Neural Information Processing Systems*, 33: 679–688.

Zhang, Y.; and Xiao, L. 2017. Stochastic primal-dual coordinate method for regularized empirical risk minimization. *Journal of Machine Learning Research*, 18(84): 1–42.

Zhang, Y.; Yao, Y.; Ram, P.; Zhao, P.; Chen, T.; Hong, M.; Wang, Y.; and Liu, S. 2022. Advancing model pruning via bi-level optimization. *Advances in Neural Information Processing Systems*, 35: 18309–18326.

Zhao, P.; Sun, F.; Shen, X.; et al. 2024. Pruning Foundation Models for High Accuracy without Retraining. In *Findings of EMNLP 2024*, 9681–9694. ACL.

Zou, D.; and Gu, Q. 2019. An improved analysis of training over-parameterized deep neural networks. *Advances in neural information processing systems*, 32.

Appendix

A More Experimental Results

A.1 Full LLaMA Results

We show the full results of LLaMA family models on three different datasets in Table 5 varying sparsity ratio from 10% to 70% with all kinds of model size. Our method consistently achieves better performance than all other three state-of-the-art methods.

Method	Prune Ratio	LLaMA-7B			LLaMA-13B			LLaMA-30B			LLaMA-65B		
		Wiki	PTB	C4	Wiki	PTB	C4	Wiki	PTB	C4	Wiki	PTB	C4
Baseline	\	5.68	27.34	7.08	5.09	19.23	6.61	4.10	16.29	5.98	3.53	17.61	5.62
LLM-Pruner	10%	7.41	36.73	9.25	6.38	31.85	8.16	4.92	18.17	6.63	3.98	19.44	6.08
SliceGPT	10%	6.97	88.48	23.54	6.11	60.15	20.18	5.24	39.72	17.83	4.57	36.20	14.14
FLAP	10%	6.34	32.39	8.058	5.45	20.99	7.33	4.52	17.29	6.49	3.91	19.35	6.04
Ours	10%	**6.01**	**31.65**	**7.94**	**5.38**	**20.52**	**7.27**	**4.43**	**17.26**	**6.47**	**3.82**	**19.28**	**6.02**
LLM-Pruner	20%	10.73	59.73	12.15	6.38	31.85	9.42	5.83	20.18	7.55	4.65	21.85	6.75
SliceGPT	20%	8.42	120.89	35.93	7.17	86.26	29.70	6.18	50.95	26.85	5.34	61.09	21.86
FLAP	20%	7.40	36.77	9.99	6.03	23.33	8.42	5.18	19.30	7.42	4.45	21.45	6.75
Ours	20%	**6.60**	**35.75**	**9.49**	**5.89**	**23.11**	**8.39**	**4.92**	**18.58**	**7.36**	**4.26**	**20.94**	**6.73**
LLM-Pruner	30%	18.58	93.24	17.78	11.81	45.42	12.65	7.59	24.97	9.08	5.52	26.38	7.53
SliceGPT	30%	12.75	258.90	67.33	9.18	125.40	46.46	7.74	75.89	42.71	6.56	74.43	35.68
FLAP	30%	9.18	47.35	13.08	6.97	27.36	10.01	6.28	21.88	8.53	5.10	23.91	7.59
Ours	30%	**7.56**	**41.05**	**11.53**	**6.57**	**26.27**	**9.98**	**5.46**	**20.48**	**8.46**	**4.75**	**22.13**	**7.51**
LLM-Pruner	40%	38.27	238.09	29.97	20.24	75.17	18.81	10.59	40.47	11.76	6.92	31.46	9.03
SliceGPT	40%	250.2	1657	834.8	13.80	196.9	83.46	11.53	136.3	78.77	9.11	115.2	64.33
FLAP	40%	12.34	65.54	16.95	8.67	35.91	12.31	8.73	29.11	10.12	5.83	23.61	8.55
Ours	40%	**8.95**	**52.84**	**14.56**	**7.41**	**29.54**	**12.01**	**6.21**	**22.90**	**9.93**	**5.33**	**22.79**	**8.42**
LLM-Pruner	50%	126.0	460.7	73.88	45.69	152.99	36.94	19.68	78.29	18.64	9.34	43.79	12.16
SliceGPT	50%	1540	6364	4847	18.75	277.34	122.5	15.60	195.4	118.5	12.01	160.3	92.66
FLAP	50%	21.89	135.8	30.86	12.88	53.54	18.37	13.41	47.30	13.17	6.98	28.52	10.36
Ours	50%	**11.66**	**82.55**	**20.72**	**8.91**	**37.56**	**16.12**	**7.25**	**26.68**	**11.91**	**6.02**	**25.17**	**9.73**
LLM-Pruner	60%	2194	1164	1327	357.5	588.7	189.1	71.71	288.1	55.66	24.91	168.1	35.05
SliceGPT	60%	2559	6401	5816	31.92	459.4	223.2	28.40	338.3	208.9	20.84	278.3	168.9
FLAP	60%	57.51	626.2	120.4	31.36	206.7	42.99	12.84	54.01	18.96	9.79	58.73	17.66
Ours	60%	**18.91**	**195.15**	**40.68**	**12.15**	**68.04**	**26.18**	**8.78**	**34.30**	**16.02**	**7.10**	**32.53**	**12.43**
LLM-Pruner	70%	9010	4111	2655	5900	6039	1334	895.7	3274	456.9	Nan	Nan	Nan
SliceGPT	70%	3605	7304	8096	67.65	874.9	537.4	71.25	633.1	406.6	102.4	863.9	662.8
FLAP	70%	577.9	1835	833.7	647.8	1588	975.1	2786	2735	2416	Nan	2333	Nan
Ours	70%	**162.9**	**721.3**	**361.6**	**41.66**	**275.7**	**115.3**	**39.88**	**124.2**	**50.43**	**9.65**	**69.49**	**20.84**

Table 5: Perplexity (↓) results for LLaMA-1 family models with different pruning ratios on WikiText2, PTB, and C4 with 2048 sequence length.

A.2 LLaMA-2 Results

We show the detailed perplexity results of LLaMA-2 family models on three different datasets in Table 6. Our methods achieves better performance and shows better generalization on different datasets than all the other methods.

Method	Prune Ratio	LLaMA-2-7B			LLaMA-2-13B			LLaMA-2-70B		
		Wiki	PTB	C4	Wiki	PTB	C4	Wiki	PTB	C4
Baseline	\	5.47	22.51	6.97	4.88	28.87	6.47	3.32	15.65	5.52
LLM-Pruner	10%	6.70	30.60	8.65	5.94	34.46	7.61	3.79	16.78	5.91
SliceGPT	10%	6.86	63.25	24.66	6.01	69.44	21.81	4.44	37.37	15.09
FLAP	10%	6.04	25.52	8.04	5.33	35.17	7.40	3.62	16.67	5.90
Ours	10%	**5.88**	**25.39**	**7.98**	**5.23**	**34.14**	**7.33**	**3.52**	**16.26**	**5.88**
LLM-Pruner	20%	9.35	42.27	11.53	8.27	54.67	9.77	4.18	18.97	6.82
SliceGPT	20%	8.38	103.5	37.50	7.26	101.1	33.08	5.28	51.92	23.19
FLAP	20%	7.12	31.70	10.18	6.09	43.63	9.27	4.15	18.75	6.72
Ours	20%	**6.76**	**31.60**	**10.10**	**5.93**	**43.59**	**9.14**	**3.90**	**18.67**	**6.69**
LLM-Pruner	30%	16.18	74.09	17.09	12.01	98.56	13.37	4.97	20.97	7.59
SliceGPT	30%	11.64	172.78	60.43	9.84	166.0	54.05	6.67	79.59	38.31
FLAP	30%	8.92	42.85	13.05	7.14	51.69	11.57	4.84	20.63	7.56
Ours	30%	**7.92**	**42.28**	**12.98**	**6.95**	**49.71**	**11.50**	**4.43**	**20.32**	**7.55**
LLM-Pruner	40%	42.22	175.1	34.32	21.03	178.7	20.51	7.01	26.60	9.74
SliceGPT	40%	18.92	295.0	108.2	15.74	241.35	96.82	9.83	134.12	73.26
FLAP	40%	13.02	70.23	19.34	9.10	70.46	14.96	5.86	23.05	8.84
Ours	40%	**9.82**	**61.52**	**17.97**	**8.10**	**61.29**	**14.34**	**5.11**	**22.58**	**8.77**
LLM-Pruner	50%	165.1	375.3	149.0	41.71	308.5	43.42	9.95	41.62	13.96
SliceGPT	50%	25.94	391.8	151.0	21.27	319.66	133.08	13.20	186.35	109.7
FLAP	50%	25.14	190.5	39.87	14.39	126.8	22.36	7.09	28.62	10.58
Ours	50%	**12.92**	**100.6**	**27.03**	**9.71**	**89.20**	**19.26**	**5.91**	**27.78**	**10.51**
LLM-Pruner	60%	Nan	5389	Nan	169.9	803.0	157.5	35.72	139.7	48.60
SliceGPT	60%	43.95	602.7	305.0	35.52	488.7	214.9	22.87	317.2	192.9
FLAP	60%	58.60	271.4	150.5	31.32	559.4	52.21	9.33	40.29	14.64
Ours	60%	**19.73**	**249.6**	**53.35**	**13.19**	**203.3**	**32.44**	**7.06**	**36.22**	**13.88**

Table 6: Results of LLaMA-2 model family.

A.3 LLaMA-3 Results

We further show the detailed perplexity results of LLaMA-3 family models on three different datasets in Table 7. LLM-Pruner codebase does not support LLaMA-3 models. We achieve consistent better performance than all the other models on three datasets with all kinds of model sizes.

Method	Prune Ratio	LLaMA-3-8B			LLaMA-3-70B		
		Wiki	PTB	C4	Wiki	PTB	C4
Baseline	\	6.14	10.60	8.92	2.86	8.17	6.76
SliceGPT	10%	10.55	70.80	72.01	6.93	82.40	45.76
FLAP	10%	7.40	12.98	11.89	3.86	8.87	7.97
Ours	10%	**7.05**	**12.81**	**11.81**	**3.65**	**8.74**	**7.92**
SliceGPT	20%	15.61	147.5	122.9	8.57	143.9	73.17
FLAP	20%	9.30	16.26	16.82	5.49	9.82	9.57
Ours	20%	**8.58**	**16.23**	**16.69**	**5.15**	**9.51**	**9.53**
SliceGPT	30%	29.68	255.9	212.2	12.22	231.6	123.2
FLAP	30%	12.70	22.35	23.37	6.82	10.74	11.87
Ours	30%	**10.97**	**21.41**	**23.07**	**6.39**	**10.53**	**11.11**
SliceGPT	40%	54.94	472.8	334.0	23.35	393.5	263.0
FLAP	40%	19.04	37.86	37.61	8.23	12.37	12.96
Ours	40%	**13.95**	**28.53**	**31.07**	**7.52**	**12.00**	**12.94**
SliceGPT	50%	91.28	615.2	410.47	37.82	553.0	360.6
FLAP	50%	39.44	80.21	85.60	10.21	15.72	15.87
Ours	50%	**18.51**	**53.01**	**48.36**	**8.82**	**14.45**	**15.69**

Table 7: Results of LLaMA-3 model family.

A.4 Results on Generation and Math Datasets

We provide the results of our method on generation and math datasets in Table 8. The results demonstrate that our pruning method effectively preserves both the model's generation capabilities and its mathematical performance.

Sparsity	LLaMA-7B		LLaMA-13B		LLaMA-30B		LLaMA-2-7B		LLaMA-2-13B	
Dataset	MMLU	GSM8K	MMLU	GSM8K	MMLU	GSM8K	MMLU	GSM8K	MMLU	GSM8K
Dense	34.9	9.7	46.9	16.9	58.2	36.3	45.8	15.4	54.9	23.6
10%	31.8	4.3	40.5	11.0	53.1	17.9	39.3	9.2	48.3	15.6
20%	29.3	2.3	36.3	6.5	47.7	13.3	33.1	6.5	41.0	9.5
30%	27.4	1.5	33.8	3.4	42.5	6.5	30.1	3.1	36.1	3.9
40%	26.5	2.7	29.7	2.0	37.6	4.9	27.4	1.5	28.8	1.8
50%	25.2	1.5	29.1	2.3	35.3	1.6	25.6	1.6	27.7	1.6

Table 8: Results of LLaMA model family on MMLU and GSM8K datasets.

A.5 Full Results for Common Sense Reasoning Datasets

We provide the task performance on common sense reasoning dataset with LLaMA-7B and LLaMA-13B models with 10% and 20% sparsity in Table 9. The results show that our method can perform better than other three methods.

Method	Prune Ratio	BoolQ	PIQA	Hella Swag	Wino Grande	ARC-e	ARC-c	OBQA	Average Acc.
LLaMA-7B	/	73.18	78.35	72.99	67.01	67.45	41.38	42.40	63.25
LLM-Pruner(v)	10%	67.95	77.42	69.31	63.54	66.33	39.85	41.20	60.80
LLM-Pruner(e2)		68.29	76.88	70.25	64.33	65.28	40.10	39.60	60.68
LLM-Pruner(e1)		66.97	77.26	70.30	64.33	65.24	40.19	41.00	60.76
SliceGPT	10%	57.68	69.80	59.32	68.11	62.75	36.01	38.00	55.95
FLAP	10%	74.43	75.41	68.68	67.01	65.78	38.48	41.00	61.54
Ours	10%	71.07	77.53	71.75	68.11	61.45	39.59	41.6	**61.58**
LLM-Pruner(v)	20%	61.44	71.71	57.27	54.22	55.77	33.96	38.40	53.25
LLM-Pruner(e2)		59.39	75.57	65.34	61.33	59.18	37.12	39.80	56.82
LLM-Pruner(e1)		57.06	75.68	66.80	59.83	60.94	36.52	40.00	56.69
SliceGPT	20%	37.89	64.09	45.67	62.75	53.62	31.74	33.20	46.99
FLAP	20%	68.59	74.21	64.98	64.40	59.89	37.80	40.20	58.58
Ours	20%	67.92	74.76	67.31	66.54	58.80	36.77	39.40	**58.79**
LLaMA-13B	/	68.47	78.89	76.24	70.09	74.58	44.54	42.00	64.97
LLM-Pruner(c)	10%	68.47	74.76	66.99	66.38	66.58	35.24	38.20	59.52
LLM-Pruner(b)		70.64	78.40	75.00	69.46	72.82	41.47	41.40	64.17
SliceGPT	10%	61.74	69.97	60.74	69.38	66.79	40.70	41.80	58.73
FLAP	10%	63.76	78.07	73.69	69.61	69.53	39.93	41.60	62.31
Ours	10%	70.21	77.97	76.04	69.69	70.92	43.00	41.80	**64.23**
LLM-Pruner(c)	20%	62.39	66.87	49.17	58.96	49.62	31.83	33.20	50.29
LLM-Pruner(b)		67.68	77.15	73.41	65.11	68.35	38.40	42.40	61.79
SliceGPT	20%	50.34	66.00	53.37	68.11	60.56	36.35	38.20	53.27
FLAP	20%	62.23	76.50	70.59	68.35	65.66	38.99	41.60	60.56
Ours	20%	63.18	77.04	71.81	70.40	66.79	41.55	42.40	**61.88**

Table 9: Pruning results for LLaMA-7B and LLaMA-13B on common sense reasoning datasets. LLM-Pruner (v), (e2), and (e1) denote vector-wise and element-wise, while (c) and (b) represent channel and block pruning.

A.6 More Results of Image Generation Tasks

We visualize more image generation results in Figure 6.

Figure 6: Visualization of generated images through LlamaGen-3B in 384×384 resolution (cfg=1.65) with 10% sparsity.

A.7 Full Results with 128 Sequence Length

We provide the full perplexity results with 128 input sequence length on three different datasets in Table 10. The results show the effectiveness of our method for short input sequence.

Prune Ratio	10%			20%			30%			40%			50%		
Dataset	Wiki	PTB	C4	Wiki	PTB	C4	Wiki	PTB	C4	Wiki	PTB	C4	Wiki	PTB	C4
LLM-Pruner	15.37	64.54	12.27	19.09	82.03	15.51	30.64	116.9	21.29	52.28	212.6	33.69	122.8	419.7	67.67
SliceGPT	14.52	113.2	33.68	19.27	154.1	58.23	44.96	349.1	164.5	535.5	2005	1505	2241	7401	5224
FLAP	13.84	62.28	11.51	14.62	67.17	13.95	17.62	78.30	17.71	22.32	100.8	22.22	31.80	157.4	34.51
Ours	**13.31**	**61.16**	**11.45**	**14.47**	**65.96**	**14.20**	**16.40**	**74.52**	**17.52**	**19.04**	**89.62**	**21.69**	**23.32**	**125.9**	**29.87**

Table 10: Pruning results for LLaMA-7B model on WikiText2, PTB, and C4 with 128 sequence length.

B Derivation of Theorem 3

The Lagrange function for Problem (10) is

$$\begin{aligned}\mathscr{L}(\delta W, \lambda) =& \|X\delta W\|^2 + \sum_i \lambda_i^\top (M_p^\top \delta W + W_p) \cdot e_i \\ =& \operatorname{tr}[\delta W^\top X^\top X \delta W] + \sum_i \lambda_i^\top (M_p^\top \delta W + W_p) \cdot e_i,\end{aligned} \tag{13}$$

where $\lambda_i \in R^{k\times 1}$ denotes the Lagrange multiplier corresponding to the constraint for the i-th column in Eq. (10). e_i is a one-hot vector with the i-th element as 1 and all others as zero. Ae_i denotes selecting the i-th column of a matrix. Note that for the constraint in Eq. (10), $\delta M_p^\top W + W_p$ is a matrix and every element in the matrix should be 0. Thus, in the Lagrange function, we assign a Lagrange multiplier for each element in the constraint. Specifically, $\lambda_i = [\lambda_{i1}, \lambda_{i2}, ..., \lambda_{ik}]^\top$ and each λ_{ij} corresponds to the j-th row and i-th column of the constraint. And their sums are computed in the Lagrange function. The trace function $\operatorname{tr}[\cdot]$ computes the ℓ_2 norm of $X\delta W$.

The gradients with reference to δW should be 0, *i.e.*,

$$\frac{\delta \mathscr{L}(\delta W, \lambda)}{\delta(\delta W)} = 2X^\top X \delta W + \sum_i M_p \lambda_i e_i^\top = 0. \tag{14}$$

δW can be derived as below,

$$\delta W = -(2X^\top X)^{-1} \left(\sum_i M_p \lambda_i e_i^\top \right). \tag{15}$$

By applying Equation (15) in Eq. 13, we have the following,

$$\begin{aligned} g(\lambda) =& \operatorname{tr}\left[\left(\sum_i e_i \lambda_i^\top M_p^\top \right) (2XX^\top)^{-1} X^\top X (2X^\top X)^{-1} \left(\sum_i M_p \lambda_i e_i^\top \right) \right] \\ & - \sum_i \lambda_i^\top M_p^\top (2X^\top X)^{-1} \left(\sum_i M_p \lambda_i e_i^\top \right) e_i + \sum_i \lambda_i^\top W_p e_i \\ =& \operatorname{tr}\left[X(2X^\top X)^{-1} \left(\sum_i M_p \lambda_i e_i^\top \right) \left(\sum_i e_i \lambda_i^\top M_p^\top \right) (2XX^\top)^{-1} X^\top \right] \\ & - \sum_i \lambda_i^\top M_p^\top (2X^\top X)^{-1} M_p \lambda_i + \sum_i \lambda_i^\top W_p e_i \\ =& \operatorname{tr}\left[\sum_i X(2X^\top X)^{-1} M_p \lambda_i \lambda_i^\top M_p^\top (2XX^\top)^{-1} X^\top \right] \\ & - \sum_i \lambda_i^\top M_p^\top (2X^\top X)^{-1} M_p \lambda_i + \sum_i \lambda_i^\top W_p e_i \\ =& \sum_i \lambda_i^\top M_p^\top (2XX^\top)^{-1} X^\top X (2X^\top X)^{-1} M_p \lambda_i \\ & - \sum_i \lambda_i^\top M_p^\top (2X^\top X)^{-1} M_p \lambda_i + \sum_i \lambda_i^\top W_p e_i \\ =& -\frac{1}{2} \sum_i \lambda_i^\top M_p^\top (2X^\top X)^{-1} M_p \lambda_i + \sum_i \lambda_i^\top W_p e_i \end{aligned} \tag{16}$$

Note that $e_i^\top e_i = 1$ and $e_i^\top e_j = 0$, if $i \neq j$. Besides, we can switch the position of two terms in the trace function, such as $X(2X^\top X)^{-1} M_p \lambda_i$ and $\lambda_i^\top M_p^\top (2XX^\top)^{-1} X^\top$. Further, after switching the two terms in the trace function, if the output is a scale, we can omit the trace function.

The gradients with reference to λ should be 0, *i.e.*,

$$\frac{\delta g(\lambda)}{\delta \lambda_i} = -M_p^\top (2X^\top X)^{-1} M_p \lambda_i + W_p e_i = 0, \tag{17}$$

We can obtain the optimal λ as below,

$$\lambda_i^* = (M_p^\top (2X^\top X)^{-1} M_p)^{-1} W_p e_i, \tag{18}$$

The optimal δW can be derived as below,

$$\begin{aligned}
\delta W^* =& -(2XX^\top)^{-1} \left(\sum_i M_p (M_p^\top (2X^\top X)^{-1} M_p)^{-1} W_p e_i e_i^\top \right) \\
=& -(2XX^\top)^{-1} \left(\sum_i M_p (M_p^\top (2X^\top X)^{-1} M_p)^{-1} M_p^\top W e_i e_i^\top \right).
\end{aligned} \tag{19}$$

The minimal loss/error corresponding to the optimal δW can be obtained by

$$\begin{aligned}
L^* &= \frac{1}{2} \sum_i \lambda_i^\top M_p^\top (2X^\top X)^{-1} M_p \lambda_i \\
&= \frac{1}{2} \sum_i e_i^\top W_p^\top (M_p^\top (2X^\top X)^{-1} M_p)^{-1} M_p^\top (2X^\top X)^{-1} M_p (M_p^\top (2X^\top X)^{-1} M_p)^{-1} W_p e_i \\
&= \frac{1}{2} \sum_i e_i^\top W_p^\top (M_p^\top (2X^\top X)^{-1} M_p)^{-1} W_p e_i \\
&= \frac{1}{2} \sum_i (W^\top M_p (M_p^\top (2XX^\top)^{-1} M_p)^{-1} M_p^\top W)_{i,i}.
\end{aligned} \tag{20}$$

Optimal Solution. As demonstrated in Eq. (19), since e_i is a one-hot vector, $Ae_ie_i^\top$ only has non-zero values in the i-th column with all zeros for all other columns. Thus, in the sum of Eq. (19), each term indexed by i just computes the i-th column of the output. Furthermore, the computation of the i-th column does not affect the j-th column, $\forall j \neq i$. For each column, we have the following,

$$\begin{aligned}
(\delta W^*)_{*,i} =& -(2XX^\top)^{-1} \left(M_p (M_p^\top (2X^\top X)^{-1} M_p)^{-1} M_p^\top W e_i \right), \\
=& \left(-(2XX^\top)^{-1} \left(M_p (M_p^\top (2X^\top X)^{-1} M_p)^{-1} M_p^\top W \right) \right)_{*,i}
\end{aligned} \tag{21}$$

Thus we can obtain the following optimal solution,

$$\delta W^* = -(2XX^\top)^{-1} \left(M_p (M_p^\top (2X^\top X)^{-1} M_p)^{-1} M_p^\top W \right). \tag{22}$$

C Preliminary

In this section, we present preliminary concepts, several basic facts, and definitions for our paper.

C.1 Notations

For two vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$, we use $\langle x, y \rangle$ to denote the inner product between x, y, i.e., $\langle x, y \rangle = \sum_{i=1}^n x_i y_i$. We use e_i to denote a vector where only i-th coordinate is 1, and other entries are 0. For each $a, b \in \mathbb{R}^n$, we use $a \circ b \in \mathbb{R}^n$ to denote the vector where i-th entry is $(a \circ b)_i = a_i b_i$ for all $i \in [n]$. We use $\mathbf{1}_n$ to denote a length-n vector where all the entries are ones. We denote the i-th row of a matrix A as A_i. We use $x_{i,j}$ to denote the j-th coordinate of $x_i \in \mathbb{R}^n$. We use $\|x\|_p$ to denote the ℓ_p norm of a vector $x \in \mathbb{R}^n$, i.e. $\|x\|_1 := \sum_{i=1}^n |x_i|$, $\|x\|_2 := (\sum_{i=1}^n x_i^2)^{1/2}$, and $\|x\|_\infty := \max_{i \in [n]} |x_i|$. For $k > n$, for any matrix $A \in \mathbb{R}^{k \times n}$, we use $\|A\|$ to denote the spectral norm of A, i.e. $\|A\| := \sup_{x \in \mathbb{R}^n} \|Ax\|_2 / \|x\|_2$. For a tensor $X \in \mathbb{R}^{B \times N \times D}$ and a matrix $U \in \mathbb{R}^{D \times d_1}$, we define $Y = X \cdot U \in \mathbb{R}^{B \times N \times d_1}$. For a matrix $V \in \mathbb{R}^{d_2 \times B}$ and a tensor $X \in \mathbb{R}^{B \times N \times D}$, we define $Z = V \cdot X \in \mathbb{R}^{d_2 \times N \times D}$. For a square matrix A, we use $\mathrm{tr}[A]$ to denote the trace of A, *i.e.*, $\mathrm{tr}[A] = \sum_{i=1}^n A_{i,i}$.

C.2 Facts

Here we provide facts we use.

Fact 6 (Norm bounds)**.** *For $a, b \in \mathbb{R}^d$ and $A \in \mathbb{R}^{n \times d}$, we have*

- $\|a \circ b\|_2 \leq \|a\|_2 \cdot \|b\|_2$.
- $\|Ab\|_2 \leq \|A\| \cdot \|b\|_2$.

Fact 7 (Linear algebra)**.** *For $a, b \in \mathbb{R}^d$ and $A, B \in \mathbb{R}^{n \times d}$ and $C \in \mathbb{R}^{d \times d}$, we have*

- $a \circ b = b \circ a = \mathrm{diag}(a) b = \mathrm{diag}(b) a$.
- $\mathrm{diag}(a) C \, \mathrm{diag}(b) = (a b^\top) \circ C$.
- $A B^\top = \sum_{i \in [d]} A_{*,i} B_{*,i}^\top$.

Fact 8 (Calculus)**.** *For $a, b \in \mathbb{R}^d$ and $A \in \mathbb{R}^{n \times d}$, we have*

- $\frac{\mathrm{d} a \circ b}{\mathrm{d} a_j} = b_j e_j$ *for* $j \in [d]$.
- $\frac{\mathrm{d} \langle a, b \rangle}{\mathrm{d} a_j} = b_j$ *for* $j \in [d]$.
- $\frac{\mathrm{d} A b}{\mathrm{d} b} = A$.
- $\frac{\mathrm{d} 0.5 \|b\|_2^2}{\mathrm{d} a} = a^\top \frac{\mathrm{d} b}{\mathrm{d} a}$.

C.3 Internal Computation Alignment

Assume the input $X \in \mathbb{R}^{B \times N \times D}$ where B denotes the batch size, N denotes the number of tokens, D denotes the dimension of hidden states, and the pruned weights can be denoted as $W_q, W_k, W_v \in \mathbb{R}^{D \times D_p}$ and $W_o \in \mathbb{R}^{D_p \times D}$ where $D_p \leq D$ denotes the pruned dimension size. The computation in the self-attention mechanism can be presented as follows,

$$\mathrm{Attention}(Q, K, V) := \mathrm{Softmax}(Q K^\top / \sqrt{D_k}) \cdot V \ \in \mathbb{R}^{B \times N \times D_p},$$

where $Q = X \cdot W_q, K = X \cdot W_k, V = X \cdot W_v$. Multiplying W_o, we have the final output:

$$Z^{\mathrm{attn}} = \mathrm{Attention}(Q, K, V) \cdot W_o \ \in \mathbb{R}^{B \times N \times D}.$$

Similarly, for the MLP module, we define the up, gate, and down projection weights as $W_{\mathrm{up}}, W_{\mathrm{gate}} \in \mathbb{R}^{D \times D_p}$ and $W_{\mathrm{down}} \in \mathbb{R}^{D_p \times D}$, respectively. With the same input $X \in \mathbb{R}^{B \times N \times D}$, the MLP module can be expressed as follows,

$$H_{\mathrm{Up}} := X \cdot W_{\mathrm{up}} \in \mathbb{R}^{B \times N \times D_p} \ \text{ and } \ H_{\mathrm{Gate}} := X \cdot W_{\mathrm{gate}} \in \mathbb{R}^{B \times N \times D_p}.$$

The final output then is

$$Z^{\mathrm{mlp}} = (H_{\mathrm{Up}} \circ H_{\mathrm{Gate}}) \cdot W_{\mathrm{down}} \in \mathbb{R}^{B \times N \times D}.$$

D Theory

D.1 More Related Work

Optimization. Optimization is a cornerstone of computer science and mathematics, encompassing various techniques to find optimal solutions. Linear Programming (LP) and Semi-Definite Programming (SDP) form a fundamental basis in this field, addressing problems with linear/quadratic objective functions and constraints, offering powerful tools for diverse applications (Anstreicher 2000; d'Aspremont et al. 2006; Amini and Wainwright 2009; Diakonikolas et al. 2019; Dong, Hopkins, and Li 2019; Cohen, Lee, and Song 2021; Jambulapati, Li, and Tian 2020; Jiang et al. 2020a; Gu and Song 2022; Song, Ye, and Zhang 2023). The concept of dynamic maintenance has gained prominence, focusing on efficient solution updates as input data changes (Cohen, Lee, and Song 2021; Lee, Song, and Zhang 2019; Brand 2020; Jiang et al. 2020b; Brand et al. 2020a; Jiang et al. 2020c; Song and Yu 2021; Dong, Lee, and Ye 2023; van den Brand 2020; Jiang et al. 2020a; Huang et al. 2021a; Gu and Song 2022). In machine learning, optimization techniques have significantly impacted Support Vector Machines (SVMs) (Chang and Lin 2001; Joachims 2006; Gu, Song, and Zhang 2023; Gao et al. 2023a; Tarzanagh et al. 2023; Brand, Song, and Zhou 2023; Li, Song, and Zhou 2023; Gao, Mahadevan, and Song 2023) and Empirical Risk Minimization (ERM) (Nesterov 1983; Vapnik 1991; Polyak and Juditsky 1992; Bartlett, Bousquet, and Mendelson 2005; Bottou and Bousquet 2007; Nemirovski et al. 2009; Moulines and Bach 2011; Feldman et al. 2012; Nesterov 2013; Johnson and Zhang 2013; Vapnik 2013; Shalev-Shwartz and Zhang 2013; Défossez and Bach 2014; Frostig et al. 2015; Zhang and Xiao 2017; Jin et al. 2018; Lee, Song, and Zhang 2019; Lianke et al. 2023; Bian, Song, and Yin 2023), enhancing their efficiency and performance. Recent work continues to expand the application of optimization across various domains, including graph algorithms, numerical methods, and advanced machine learning architectures (Li and Liang 2018; Du et al. 2018; Arora et al. 2019a,b; Song and Yang 2019; Cai et al. 2019; Zhang, Martens, and Grosse 2019; Cao and Gu 2019; Zou and Gu 2019; Oymak and Soltanolkotabi 2020; Ji and Telgarsky 2019; Lee et al. 2020; Huang et al. 2021b; Zhang et al. 2020b; Brand et al. 2020b; Zhang et al. 2020a; Song, Zhang, and Zhang 2021; Alman et al. 2022; Zhang 2022; Gao, Mahadevan, and Song 2023; Li, Song, and Zhou 2023; Qin, Song, and Yang 2023; Chu, Song, and Yang 2024; Shrivastava, Song, and Xu 2023; Qin, Song, and Sun 2023; Deng, Mahadevan, and Song 2023; Liang et al. 2024a; Li et al. 2024c), demonstrating its ongoing significance in advancing computational capabilities.

Attention Theory. Attention mechanisms have been extensively studied and developed over the years, becoming a fundamental component in various neural network architectures. (Deng, Li, and Song 2023; Li et al. 2023a; Gu et al. 2024a) investigate the softmax regression problem, while (Song, Ye, and Zhang 2023) propose and analyze the attention kernel regression problem. The rescaled hyperbolic functions regression is examined by (Gao, Song, and Yin 2023). Following this, (Song, Wang, and Yin 2023; Li et al. 2023b) delve into two-layer attention regression problems. (Gu et al. 2024b) demonstrate that attention layers in transformers learn two-dimensional cosine functions. Additionally, (Deng et al. 2023) explore data recovery using attention weights, and (Kacham, Mirrokni, and Zhong 2023; Song, Xu, and Yin 2023) investigate the replacement of the softmax unit with a polynomial unit. Moreover, some works theoretically explore variations or combinations of the attention mechanism with other techniques, such as quantum attention (Gao et al. 2023b, 2024), tensor attention (Alman and Song 2024; Liang et al. 2024e), and differentially private attention (Liang et al. 2024d; Gao et al. 2023c; Gu et al. 2024c) and other applications such as (Clark et al. 2019b; Tenney, Das, and Pavlick 2019; Hewitt and Liang 2019; Vig and Belinkov 2019; Belinkov 2022; Brand, Song, and Zhou 2023; Chen et al. 2024c,b,a; Liang et al. 2024b; Li et al. 2024b; Liang et al. 2024c,f,e).

D.2 Error Bound for Masked Weight

In this section, we show how to derive the error bound for masked weight. We assume $B = 1$ for simplicity of proofs.

Lemma 9 (formal version of Lemma 1)**.** *If the following conditions hold:*

- *Let $W \in \mathbb{R}^{D \times D'}$, $X \in \mathbb{R}^{N \times D}$.*
- *Let $M \in \{0,1\}^D$ and $\rho \in [0,1]$ be ratio of number of zeros to number of entries in M.*
- *Assume $\|X\| \leq R$.*

We can show that for $i \in [D']$ we have

$$\|XW_{*,i} - X(M \circ W_{*,i})\|_2 \leq \rho R \|W_{*,i}\|_2$$

Proof. We can show

$$\begin{aligned} \|XW_{*,i} - X(M \circ W_{*,i})\|_2 &= \|X(W_{*,i} - M \circ W_{*,i}))\|_2 \\ &\leq \|X\| \cdot \|(\mathbf{1}_D - M) \circ W_{*,i}\|_2 \\ &\leq \|X\| \cdot \|(\mathbf{1}_D - M)\|_2 \cdot \|W_{*,i}\|_2 \\ &\leq \rho R \|W_{*,i}\|_2 \end{aligned}$$

where the first step follows from basic algebra, the second step and the third step follow from Fact 6, the last step follows from we assume $\|X\| \leq R$ and the definition of ρ. □

D.3 Find Optimal Mask

In this section, we show how to use Newton's method to find the numerical score defined in Section 3.2.

Theorem 10 (Mask optimization, formal version of Theorem 2)**.** *If the following conditions hold:*

- *Let $W \in \mathbb{R}^{D \times D'}$, $X \in \mathbb{R}^{N \times D}$.*
- *Let $z \in [0,1]^D$.*
- *Let $r \in [0, D]$ denote the number of ones (it can be a fractional number).*
- *Let $\lambda > 0$ denote a regularization co-efficients.*
- *Assume $\|X\| \leq R$.*

There exist an algorithm (Algorithm 1) that can get the optimal z such that

$$\arg\min_{z \in [0,1]^D} \frac{1}{2} \sum_{i \in [D']} \|XW_{*,i} - X(z \circ W_{*,i})\|_2^2 + \frac{1}{2}\lambda \cdot (\langle \mathbf{1}_D, z \rangle - r)^2$$

Proof. We can use Newton's method to solve this problem. To use Newton's method we need to compute the gradient and Hessian. After calculation in Lemma 12, 13, 14, and 15, we use Algorithm 1 to get optimal z. □

Gradient Calculation In this section, we compute the gradient for $L(z)$ and $L_{\mathrm{reg}}(z)$. First, we define $f(z)$ for simplicity to calculate the gradient.

Definition 11. *We define $f(z)_i := X(z \circ W_{*,i}) \in \mathbb{R}^D$ for $i \in [D']$.*

Then, we calculate the gradient for each loss.

Lemma 12 (Gradient of loss function)**.** *If the following conditions hold*

- *Let $L(z)_i := 0.5\|XW_{*,i} - X(z \circ W_{*,i})\|_2^2$ for $i \in [D']$.*

Then, we can show

- *Part 1. For each $i \in [D'], j \in [D]$*

$$\underbrace{\frac{\mathrm{d}f(z)_i}{\mathrm{d}z_j}}_{\mathrm{D}} = X_{*,j} W_{j,i}$$

- *Part 2. For each $i \in [D'], j \in [D]$*

$$\underbrace{\frac{\mathrm{d}L(z)_i}{\mathrm{d}z_j}}_{\mathrm{scalar}} = (X(z \circ W_{*,i}) - XW_{*,i})^\top X_{*,j} W_{j,i}$$

- *Part 3. For each $i \in [D']$*

$$\underbrace{\frac{\mathrm{d}L(z)_i}{\mathrm{d}z}}_{D} = ((W_{*,i} W_{*,i}^\top) \circ (X^\top X))(z - \mathbf{1}_D)$$

- *Part 4.*

$$\underbrace{\frac{\mathrm{d}L(z)}{\mathrm{d}z}}_{D} = ((WW^\top) \circ (X^\top X))(z - \mathbf{1}_D)$$

Proof. **Proof of part 1.** We can show

$$\begin{aligned} \frac{\mathrm{d}f(z)_i}{\mathrm{d}z_j} &= \frac{\mathrm{d}X(z \circ W_{*,i})}{\mathrm{d}z_j} \\ &= X(e_j W_{j,i}) \\ &= X_{*,j} W_{j,i} \end{aligned}$$

where the first step follows from Definition 11, the second step follows from Fact 8, and the last step follows from simple algebra.

Proof of part 2. We can show

$$\begin{aligned}\frac{\mathrm{d}L(z)_i}{\mathrm{d}z_j} &= \frac{\mathrm{d}0.5\|XW_{*,i} - X(z \circ W_{*,i})\|_2^2}{\mathrm{d}z_j} \\ &= (XW_{*,i} - f(z)_i)^\top \frac{\mathrm{d}(XW_{*,i} - f(z)_i)}{\mathrm{d}z_j} \\ &= (f(z)_i - XW_{*,i})^\top \frac{\mathrm{d}f(z)_i}{\mathrm{d}z_j} \\ &= (X(z \circ W_{*,i}) - XW_{*,i})^\top X_{*,j} W_{j,i}\end{aligned}$$

where the first step follows from definition of $L(z)_i$, the second step follows from Fact 8, the third step follows from simple algebra, and the last step follows from **Part 1**.

Proof of part 3. We can show

$$\begin{aligned}\frac{\mathrm{d}L(z)_i}{\mathrm{d}z} &= ((X(z \circ W_{*,i}) - XW_{*,i})^\top X \operatorname{diag}(W_{*,i}))^\top \\ &= \operatorname{diag}(W_{*,i}) X^\top (X(z \circ W_{*,i}) - XW_{*,i}) \\ &= \operatorname{diag}(W_{*,i}) X^\top X (W_{*,i} \circ (z - \mathbf{1}_D)) \\ &= \operatorname{diag}(W_{*,i}) X^\top X \operatorname{diag}(W_{*,i})(z - \mathbf{1}_D) \\ &= ((W_{*,i} W_{*,i}^\top) \circ (X^\top X))(z - \mathbf{1}_D)\end{aligned}$$

where the first step follows from **Part 2**, the second step follows from property of transpose, the third step follows from simple algebra, the fourth step follows from Fact 7, and the last step follows from Fact 7.

Proof of part 4. We can show

$$\begin{aligned}\frac{\mathrm{d}L(z)}{\mathrm{d}z} &= \frac{\mathrm{d}}{\mathrm{d}z} \sum_{i \in [D']} L(z)_i \\ &= \sum_{i \in [D']} \frac{\mathrm{d}L(z)_i}{\mathrm{d}z} \\ &= \sum_{i \in [D']} ((W_{*,i} W_{*,i}^\top) \circ (X^\top X))(z - \mathbf{1}_D) \\ &= ((WW^\top) \circ (X^\top X))(z - \mathbf{1}_D)\end{aligned}$$

where the first step follows from definition of $L(z)$, the second step follows from basic calculus, the third step follows from **Part 3**, and the last step follows from Fact 7. □

Lemma 13 (Gradient of regularization)**.** *If the following conditions hold*

- *Let* $L_{\mathrm{reg}}(z) := 0.5\lambda \cdot (\langle \mathbf{1}_D, z \rangle - r)^2$.

Then, we can show

- *Part 1. For each* $j \in [D]$

$$\underbrace{\frac{\mathrm{d}L_{\mathrm{reg}}(z)}{\mathrm{d}z_j}}_{\mathrm{scalar}} = \lambda(\langle \mathbf{1}_D, z \rangle - r)$$

- *Part 2.*

$$\underbrace{\frac{\mathrm{d}L_{\mathrm{reg}}(z)}{\mathrm{d}z}}_{D} = \lambda(\langle \mathbf{1}_D, z \rangle - r) \cdot \mathbf{1}_D$$

Proof. **Proof of part 1.** We can show

$$\begin{aligned}\frac{\mathrm{d}L_{\mathrm{reg}}(z)}{\mathrm{d}z_j} &= \frac{\mathrm{d}0.5\lambda \cdot (\langle \mathbf{1}_D, z \rangle - r)^2}{\mathrm{d}z_j} \\ &= \lambda(\langle \mathbf{1}_D, z \rangle - r) \frac{\mathrm{d}\langle \mathbf{1}_D, z \rangle - r}{\mathrm{d}z_j} \\ &= \lambda(\langle \mathbf{1}_D, z \rangle - r)\end{aligned}$$

where the first step follows from definition of $L_{\mathrm{reg}}(z)$, the second step follows from basic calculus, and the last step follows from Fact 8.

Proof of part 2. Using Part 1, we can show

$$\frac{\mathrm{d}L_{\mathrm{reg}}(z)}{\mathrm{d}z} = \lambda(\langle \mathbf{1}_D, z \rangle - r) \cdot \mathbf{1}_D$$

□

Hessian Calculation In this section, we calculate the hessian for $L(z)$ and $L_{\mathrm{reg}}(z)$.

Lemma 14 (Hessian of loss function)**.** *If the following conditions hold*

- *Let* $L(z)_i := 0.5\|XW_{*,i} - X(z \circ W_{*,i})\|_2^2$ *for* $i \in [D']$.

Then, we can show

$$\underbrace{\frac{\mathrm{d}^2 L(z)}{\mathrm{d}z^2}}_{D \times D} = (WW^\top) \circ (X^\top X)$$

Proof. We can show

$$\begin{aligned}
\frac{\mathrm{d}^2 L(z)}{\mathrm{d}z^2} = & \frac{\mathrm{d}}{\mathrm{d}z} \frac{\mathrm{d}L(z)}{\mathrm{d}z} \\
= & \frac{\mathrm{d}}{\mathrm{d}z}((WW^\top) \circ (X^\top X))(z - \mathbf{1}_D) \\
= & (WW^\top) \circ (X^\top X)
\end{aligned}$$

where the first step follows from basic calculus, the second step follows from Lemma 12, and the last step follows from Fact 8. □

Lemma 15 (Hessian of regularization)**.** *If the following conditions hold*

- *Let* $L_{\mathrm{reg}}(z) := 0.5\lambda \cdot (\langle \mathbf{1}_D, z \rangle - r)^2$.

Then, we can show

$$\underbrace{\frac{\mathrm{d}^2 L_{\mathrm{reg}}(z)}{\mathrm{d}z^2}}_{D \times D} = \lambda \mathbf{1}_{D \times D}$$

Proof. We can show

$$\begin{aligned}
\frac{\mathrm{d}^2 L_{\mathrm{reg}}(z)}{\mathrm{d}z^2} = & \frac{\mathrm{d}}{\mathrm{d}z} \frac{\mathrm{d}L_{\mathrm{reg}}(z)}{\mathrm{d}z} \\
= & \frac{\mathrm{d}}{\mathrm{d}z} \lambda(\langle \mathbf{1}_D, z \rangle - r) \cdot \mathbf{1}_D \\
= & \frac{\mathrm{d}}{\mathrm{d}z} \lambda \langle \mathbf{1}_D, z \rangle \cdot \mathbf{1}_D \\
= & \lambda \mathbf{1}_{D \times D}
\end{aligned}$$

where the first step follows from basic calculus, the second step follows from Lemma 13, the third step follows from basic calculus, and the last step follows from basic calculus. □