DReSS: Data-driven Regularized Structured Streamlining for Large Language Models

Anonymous ACL submission

Abstract

Large language models (LLMs) have achieved significant progress across various domains, but their increasing scale leads to high computational and memory costs. Recent studies show that LLMs exhibit sparsity, which can be exploited for pruning. Existing pruning methods typically follow a prune-then-finetune paradigm. Since the pruned components still contain valuable information, their direct removal often leads to irreversible performance degradation, causing expensive fine-tuning to recover performance. To address this, we propose a new paradigm: first apply regularization, then prune, and finally fine-tune. Based on this paradigm, we propose DReSS, a simple and effective **D**ata-driven **Re**gularized **S**tructured **S**treamlining method for LLMs. By using a small amount of data to regularize the components before pruning, DReSS transfers the important information to the remaining parts of the model in advance. Compared to direct pruning, this can reduce the information loss caused by parameter removal, thereby enhancing its language modeling capabilities. We evaluate our method on various LLMs, including Phi-2, OPT, LLaMA2, LLaMA3. Experimental results demonstrate DReSS even without recovery fine-tuning (RFT) achieves comparable performance to previous methods, drastically alleviating computational costs. Moreover, DReSS significantly outperforms existing powerful pruning methods even under extreme pruning ratios, significantly reducing latency and increasing throughput.

1 Introduction

Large language models (LLMs) have achieved significant advancements across a wide range of tasks, demonstrating their robust capabilities (Zhang et al., 2022; Achiam et al., 2023; Touvron et al., 2023; Wu et al., 2024). However, as the model size increases, the growing number of parameters leads to significant computational and memory requirements, which significantly hinder the practical deployment of LLMs. Consequently, it is critical to develop methods that can reduce model size while maintaining performance.

To address these challenges, several methods have been proposed, including pruning (Frantar and Alistarh, 2023; Sun et al., 2024; An et al., 2024), quantization (Frantar et al., 2023; Xiao et al., 2023), knowledge distillation (Shridhar et al., 2023; Hsieh et al., 2023), and low-rank decomposition (Saha et al., 2023). In this work, we mainly focus on pruning which is an efficient and highly generalizable approach that can be seamlessly integrated with other model compression strategies. Pruning techniques are generally classified into two primary categories: unstructured pruning (Frantar and Alistarh, 2023; Sun et al., 2024) and structured pruning (Ma et al., 2023; An et al., 2024). Compared to unstructured pruning, structured pruning offers the flexibility to do recovery fine-tuning (RFT) for specific downstream tasks without relying on specialized hardware (Zhu et al., 2024). The model obtained through structured pruning typically achieves much faster inference speed due to the regular data patterns.

Despite these advancements, existing structured pruning methods still have limitations. They all follow the paradigm of first selecting channels or layers to prune based on a designed metric, and then performing RFT (Chavan et al., 2024). However, they neglect that important information can also exist in the pruned parts (Dettmers et al., 2022; Xiao et al., 2023; Yin et al., 2024), and directly removing them leads to an irreversible decline in model performance. Thus, the pruned models often require extensive data for RFT to recover performance (Ma et al., 2023). Additionally, high pruning ratios often cause performance collapse, limiting their effectiveness in acceleration.

To tackle these limitations, we propose a new pruning paradigm that first applies regularization

to the components to be pruned, then prunes those parameters, finally performs RFT. To the best of our knowledge, we are the first to propose this paradigm. The regularization process explicitly transfers important information from the parts to be pruned to the remaining parts, thereby reducing the information loss caused by direct parameter removal. Based on this paradigm, we introduce DReSS, a **D**ata-driven **Re**gularized **S**tructured **S**treamlining method for LLMs. The comparison of the DReSS with existing channel-wise pruning approaches is shown in Figure 1. Specifically, the process of DReSS is divided into the following steps: (1) select data for pre-pruning regularization and post-pruning RFT, (2) apply regularization to the selected channels in the parameter matrices, (3) prune the regularized channels, finally (4) do RFT. *First*, we randomly select a small subset of data from widely used benchmark datasets. As the data size is small, the overhead of regularization and RFT remains minimal. *Second*, we determine the channels to be pruned based on the pruning ratio, and apply regularization (ℓ_1-norm or ℓ_2-norm) to these channels using the selected data. This redistributes important information to the unpruned parts, thereby significantly mitigating performance degradation caused by direct pruning. As a result, the model retains strong performance at high pruning ratios, enabling significant speedup. *Third*, we prune the regularized channels. *Finally*, we perform RFT on the pruned model using a small subset of samples selected in the first step. Extensive experiments show that DReSS significantly outperforms existing powerful pruning methods in perplexity and accuracy across various pruning ratios on different models and is able to provide considerable acceleration. The main contributions are summarized as follows:

- **Propose A New Paradigm:** By sequentially applying regularization (ℓ_1-norm or ℓ_2-norm), pruning, and RFT, DReSS minimizes information loss caused by direct parameter removal, thereby improving the model's language modeling capabilities.
- **High Performance:** DReSS surpasses competitive structured pruning methods in perplexity and accuracy, achieving notable improvements in throughput and reduced latency compared to the dense models.
- **Low Overhead:** By using only a small amount of data for regularization and optional RFT, DReSS incurs minimal overhead.

2 Related Works

2.1 Pruning Methods

Pruning redundant weights has been an effective way to reduce deep neural network complexity for decades. (LeCun et al., 1989; Hassibi et al., 1993; Han et al., 2015). Pruning methods can be broadly categorized into two types: unstructured pruning (Kurtic et al., 2022; Zhang et al., 2024b; Xu et al., 2024) and structured pruning (Xia et al., 2024; Gao et al., 2024b).

Unstructured pruning methods remove individual weights based on designed metrics. Magnitude (Han et al., 2015) removes smaller weights, Wanda (Sun et al., 2024) considers both weights and activations, and SparseGPT (Frantar and Alistarh, 2023) uses Hessian matrix. They require specialized hardware to accelerate (Xia et al., 2023) and a high sparsity to achieve substantial acceleration (Wang, 2020).

Structured pruning methods, including channel-wise pruning (An et al., 2024) and layer-wise pruning (Men et al., 2024). Channel-wise pruning methods remove less important channels of the parameter matrices. For instance, SliceGPT (Ashkboos et al., 2024) prune channels with smaller eigenvalues. LLM Surgeon (van der Ouderaa et al., 2024) periodically updates model weights and structures, pruning more aggressively in the initial layers. Layer-wise pruning methods, such as SLEB (Song et al., 2024), iteratively prune transformer layers based on their importance. These methods have a key limitation: even less important channels or layers may contain valuable information, and pruning them directly often leads to information loss. To address this, we investigate the way of 'shifting' important information from the pruned parts to the remaining parts, which could reduce information loss caused by pruning.

2.2 Regularization

Regularization is widely used in machine learning, such as feature selection (Tibshirani, 1996) and preventing model overfitting (Santos and Papa, 2022). The ℓ_1-norm drives certain coefficients to zero, while ℓ_2-norm encourages smoother solutions (Boyd and Vandenberghe, 2004). Both of them can significantly alter the distribution pattern of the data (Han et al., 2015; Liu et al., 2017;

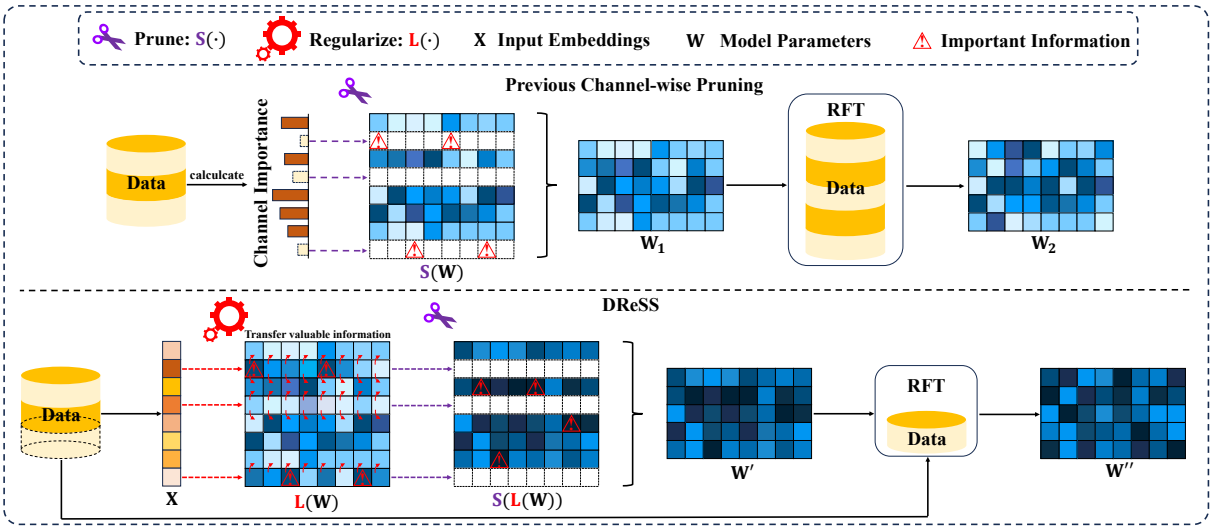

Figure 1: A comparison between previous channel-wise pruning methods and DReSS. Deeper blue square represents greater performance impact, the taller cylinder represents larger data volume. **Above**: Previous methods first select channels based on importance, followed by pruning and then do RFT. **Below**: DReSS first regularizes channels to transfer important information, prunes the channels, then do RFT with a small amount of data.

Tao et al., 2023). Motivated by this, we propose a new pruning paradigm, in which the regularization process can transfer important information from the pruned parameter space to the remaining parts of the model, thus reducing the information loss caused by parameter removal.

3 Methodology

In this section, we provide a detailed introduction to DReSS. The overall DReSS algorithm is summarized in Figure 1 and Algorithm 1, our method consists of four main steps:

- **Data Selection:** Select a small amount of data for pre-pruning regularization and post-pruning RFT.
- **Regularization:** Apply regularization to selected channels of weight matrices to transfer important information from pruned to retained parts, thereby reducing information loss and mitigating performance degradation.
- **Pruing:** Remove the channels that were regularized in the previous step.
- **RFT:** Perform RFT on the pruned model to compensate for the performance loss caused by pruning.

3.1 Data Selection

For fairness, the data is selected from widely used datasets. For instance, approximately 1,000 samples are randomly selected from the WikiText-2 (Merity et al., 2016) training set for pre-pruning regularization and post-pruning RFT.

3.2 Regularization

For clarity, we first define the notations. p is the pruning ratio, indicating $p\%$ of model parameters will be pruned. The model dimension is d, and the number of layers is l. Additionally, $\mathbf{W_{emb}} \in \mathbb{R}^{b\times n\times d}$ is the data feature map, and $\mathbf{W_{pos}} \in \mathbb{R}^{b\times n\times d}$ is the positional feature map, where b is the batch size and n is the number of tokens. The model weights are represented by $\mathbf{W} \in \mathbb{R}^{d_1\times d_2}$. Specifically, in the ith layer attention block, $\mathbf{W^i_Q} \in \mathbb{R}^{d\times d_1}$ is query matrix, $\mathbf{W^i_K} \in \mathbb{R}^{d\times d_1}$ is key matrix, $\mathbf{W^i_V} \in \mathbb{R}^{d\times d_1}$ is value matrix, and $\mathbf{W^i_o} \in \mathbb{R}^{d_1\times d}$ is output matrix. In the ith layer FFN block, $\mathbf{W^i_{up}} \in \mathbb{R}^{d\times d_2}$ is up projection matrix, and $\mathbf{W^i_{down}} \in \mathbb{R}^{d_2\times d}$ is down projection matrix. In the ith LayerNorm, $\alpha^{\mathbf{i}} \in \mathbb{R}^d$ is scaling vector and $\beta^{\mathbf{i}} \in \mathbb{R}^d$ is offset vector. $\mathbf{W_{lm}} \in \mathbb{R}^{d\times b\times n}$ denotes the language modeling matrix. $\mathbf{I} \in \mathbb{R}^{d\times d}$ is the identity matrix, let $\mathbf{R} \in \mathbb{R}^{d\times d}$ is a pseudo-index selection matrix, which is a diagonal matrix consisting of $(1-p\%)\times d$ zeros and $p\%\times d$ ones, where the ones correspond to the selected indices. $\mathbf{R}$ is identical across all layers.

Proposition 3.1 (Dependency between attention and FFN block. Proof in Appendix A). *If regularization is applied to certain columns of* $\mathbf{W^{i-1}_{down}}$ *in*

Algorithm 1 DReSS algorithm.

Input: selected data $\mathbf{X}$, number of layers in LLMs l, initial model $\mathbf{W}$, norm type: flag.
if flag==0 **then**
  continue // ℓ_2-norm
else if flag==1 **then**
  use Proposition 3.1 to transform // ℓ_1-norm
end if
$\mathbf{X_1}, \mathbf{X_2} \leftarrow divide(\mathbf{X})$
$\tilde{\mathcal{L}}_1 \leftarrow regularize(\mathbf{W_{emb}}, \mathbf{W_{pos}}, \mathbf{X_1})$
$\tilde{\mathcal{L}}_{\text{att}} \leftarrow 0$, $\tilde{\mathcal{L}}_{\text{FFN}} \leftarrow 0$, $\tilde{\mathcal{L}}_2 \leftarrow 0$
for $i = 1$ **to** l **do**
  $\tilde{\mathcal{L}}_{\text{att}} \leftarrow \tilde{\mathcal{L}}_{\text{att}} + regularize(Attention, \mathbf{X_1})$
  $\tilde{\mathcal{L}}_{\text{FFN}} \leftarrow \tilde{\mathcal{L}}_{\text{FFN}} + regularize(FFN, \mathbf{X_1})$
  $\tilde{\mathcal{L}}_2 \leftarrow \tilde{\mathcal{L}}_2 + regularize(LayerNorm, \mathbf{X_1})$
end for
$\tilde{\mathcal{L}}_3 \leftarrow regularize(\mathbf{W_{lm}}, \mathbf{X_1})$
$\tilde{\mathcal{L}}_{\text{remain}} \leftarrow \tilde{\mathcal{L}}_1 + \tilde{\mathcal{L}}_2 + \tilde{\mathcal{L}}_3$
$\mathcal{L}(\mathbf{W}, \mathbf{X}) \leftarrow forward(\mathbf{W}, \mathbf{X_1})$
$\mathcal{L}_{\text{sum}} \leftarrow \mathcal{L}(\mathbf{W}, \mathbf{X}) + \tilde{\mathcal{L}}_{\text{att}} + \tilde{\mathcal{L}}_{\text{FFN}} + \tilde{\mathcal{L}}_{\text{remain}}$
update $\mathbf{W}$ using backpropagation algorithm
$\mathbf{W} \leftarrow Prune(\mathbf{W}, \mathbf{R})$
$\mathbf{W} \leftarrow Optional_RFT(\mathbf{W}, \mathbf{X_2})$

the $(i-1)$th layer FFN block, then regularization should also be applied to the corresponding rows of $\mathbf{W^i_Q}$, $\mathbf{W^i_K}$, and $\mathbf{W^i_V}$ in the ith layer attention block. The same dependency applies to $\mathbf{W^i_o}$ and $\mathbf{W^{i-1}_{up}}$.

Figure 2 illustrates the process of applying regularization in a transformer layer. Following Proposition 1, in the attention block, regularization is applied to the rows of $\mathbf{W^i_Q}$, $\mathbf{W^i_K}$, and $\mathbf{W^i_V}$ corresponding to the one entries in the index matrix $\mathbf{R}$ and the columns of $\mathbf{W^i_o}$ associated with these entries. In the FFN block, regularization is applied on the rows of $\mathbf{W^i_{up}}$ and the columns of $\mathbf{W^i_{down}}$ that correspond to the one entries in $\mathbf{R}$. We also apply regularization to the remaining parts in LLMs. Specifically, for $\mathbf{W_{emb}}$, regularization is enforced on the columns corresponding to the one indices in $\mathbf{R}$, and the same for $\mathbf{W_{pos}}$. For each $\alpha^{\mathbf{i}}$ and $\beta^{\mathbf{i}}$ within every transformer layer, regularization is applied to the elements corresponding to the one indices in $\mathbf{R}$. Additionally, for the final $\mathbf{W_{lm}}$, due to its dependency on $\mathbf{W^l_{down}}$ in the FFN block of the last transformer layer, regularization is applied to the corresponding rows. Therefore, the overall loss function defined in Equation 1 consists of the language modeling loss $\mathcal{L}(\mathbf{W}, \mathbf{X})$ and the

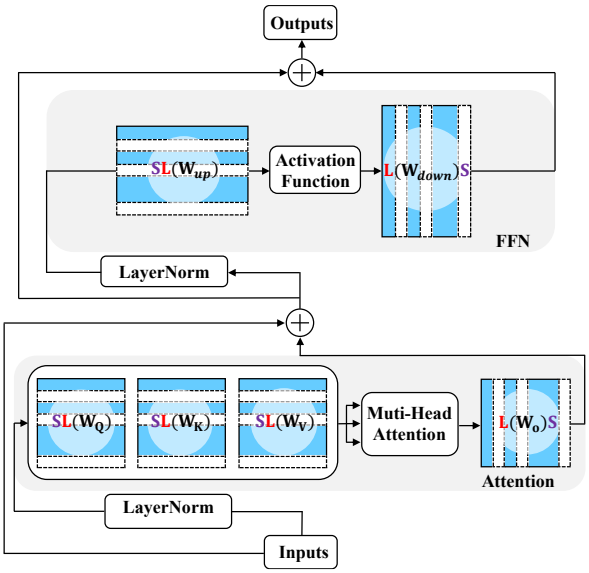

Figure 2: The regularization and pruning process of DReSS in a transformer layer.

regularization loss. $\mathbf{X}$ is the input data. Regularization loss has three parts: Attention loss: $\sum_{i=1}^{l} \tilde{\mathcal{L}}^i_{\text{att}}$, FFN loss: $\sum_{i=1}^{l} \tilde{\mathcal{L}}^i_{\text{FFN}}$, remain loss: $\tilde{\mathcal{L}}_{\text{remain}}$.

$$\mathcal{L}_{\text{sum}} = \mathcal{L}(\mathbf{W}, \mathbf{X}) + \sum_{i=1}^{l} \tilde{\mathcal{L}}^i_{\text{att}} + \sum_{i=1}^{l} \tilde{\mathcal{L}}^i_{\text{FFN}} + \tilde{\mathcal{L}}_{\text{remain}} \tag{1}$$

$\tilde{\mathcal{L}}^i_{\text{att}}$ denotes the ith layer attention block regularization loss:

$$\tilde{\mathcal{L}}^i_{\text{att}} = \lambda(\|\mathbf{RW^i_Q}\| + \|\mathbf{RW^i_K}\| + \|\mathbf{RW^i_V}\| + \|\mathbf{W^i_o R}\|) \tag{2}$$

$\tilde{\mathcal{L}}^i_{\text{FFN}}$ is the ith layer FFN block regularization loss:

$$\tilde{\mathcal{L}}^i_{\text{FFN}} = \lambda\|\mathbf{RW^i_{up}}\| + \lambda\|\mathbf{W^i_{down}R}\| \tag{3}$$

$\tilde{\mathcal{L}}_{\text{remain}}$ denotes the regularization loss of the remaining parts of the LLM:

$$\tilde{\mathcal{L}}_{\text{remain}} = \lambda(\|\mathbf{W_{emb}R}\| + \|\mathbf{W_{pos}R}\| + \sum_{i=1}^{l} \|\mathbf{R}\alpha^{\mathbf{i}}\| + \sum_{i=1}^{l} \|\mathbf{R}\beta^{\mathbf{i}}\| + \|\mathbf{RW_{lm}}\|) \tag{4}$$

In Equations 2, 3, and 4, λ is the same, $\|\cdot\|$ is a certain vector norm applied to the matrix column-wise.

Proposition 3.2 (If the loss function includes the ℓ_1-norm, it can also be solved using backpropagation. Proof in Appendix B). *The following two*

problems are equivalent, where $\|\cdot\|_1$ denotes the ℓ_1-norm.

$$\min \quad ||x||_1 \quad \Longleftrightarrow \quad \min_{x,y} \quad \mathbf{1}^T y \quad \text{s.t.} \quad -y \leq x \leq y, \quad y \geq 0. \tag{5}$$

We formalize the minimization of the overall loss as an optimization problem. When ℓ_2-norm is used, the problem can be directly solved using backpropagation (BP) algorithm (Rumelhart et al., 1986). In contrast, when the ℓ_1-norm is employed, as the regularization losses introduced by various components in LLMs are similar, we consider only $\tilde{\mathcal{L}}^i_{\text{FFN}}$ as an example, the complete objective function transformation is in Appendix C. According to Proposition 2, the optimization problem in Equation 6 can be equivalently transformed into a constrained formulation in Equation 7, enabling its solution by BP algorithm.

During forward and backward propagation, regularization on certain channels in LLMs significantly reduces the values in those rows or columns. Intuitively, this value reduction could decrease the impact of these parameters on the model's performance thus potentially transfer important information to the unregularized parts.

$$\min_{\mathbf{W}} \quad \mathcal{L}(\mathbf{W}, \mathbf{X}) + \lambda ||\mathbf{R}\mathbf{W}^{\mathbf{i}}_{\mathbf{up}}||_1 + \lambda ||\mathbf{W}^{\mathbf{i}}_{\mathbf{down}}\mathbf{R}||_1 \tag{6}$$

$$\begin{aligned} \min_{\mathbf{W},\mathbf{Y_1},\mathbf{Y_2}} \quad & \mathcal{L}(\mathbf{W}, \mathbf{X}) + \lambda(\mathbf{1}^T\mathbf{Y_1}\mathbf{1} + \mathbf{1}^T\mathbf{Y_2}\mathbf{1}) \\ \text{s.t.} \quad & -\mathbf{Y_1} \leq \mathbf{R}\mathbf{W}^{\mathbf{i}}_{\mathbf{up}} \leq \mathbf{Y_1}, \\ & -\mathbf{Y_2} \leq \mathbf{W}^{\mathbf{i}}_{\mathbf{down}}\mathbf{R} \leq \mathbf{Y_2}, \\ & \mathbf{Y_1} \geq 0, \quad \mathbf{Y_2} \geq 0. \end{aligned} \tag{7}$$

3.3 Pruning

According to Figure 2, we prune the regularized rows and columns using the pseudo-index selection matrix $\mathbf{R}$. For clarity, we define $\mathbf{S} = \mathbf{I} - \mathbf{R}$. The pruning of the ith layer attention block and FFN block are performed as:

$$\begin{aligned} \mathbf{W}^{\mathbf{i}}_{\mathbf{Q}}{}' &= \mathbf{S}\mathbf{W}^{\mathbf{i}}_{\mathbf{Q}}, & \mathbf{W}^{\mathbf{i}}_{\mathbf{K}}{}' &= \mathbf{S}\mathbf{W}^{\mathbf{i}}_{\mathbf{K}}, \\ \mathbf{W}^{\mathbf{i}}_{\mathbf{V}}{}' &= \mathbf{S}\mathbf{W}^{\mathbf{i}}_{\mathbf{V}}, & \mathbf{W}^{\mathbf{i}}_{\mathbf{o}}{}' &= \mathbf{W}^{\mathbf{i}}_{\mathbf{o}}\mathbf{S}, \\ \mathbf{W}^{\mathbf{i}}_{\mathbf{up}}{}' &= \mathbf{S}\mathbf{W}^{\mathbf{i}}_{\mathbf{up}}, & \mathbf{W}^{\mathbf{i}}_{\mathbf{down}}{}' &= \mathbf{W}^{\mathbf{i}}_{\mathbf{down}}\mathbf{S} \end{aligned} \tag{8}$$

The remaining parts are pruned as below:

$$\begin{aligned} \mathbf{W}_{\mathbf{emb}}{}' &= \mathbf{W}_{\mathbf{emb}}\mathbf{S}, \quad \mathbf{W}_{\mathbf{pos}}{}' = \mathbf{W}_{\mathbf{pos}}\mathbf{S}, \\ \alpha^{\mathbf{i}'} &= \mathbf{S}\alpha^{\mathbf{i}}, \quad \beta^{\mathbf{i}'} = \mathbf{S}\beta^{\mathbf{i}}, \quad \mathbf{W}_{\mathbf{lm}}{}' = \mathbf{S}\mathbf{W}_{\mathbf{lm}} \end{aligned} \tag{9}$$

The pruning process is straightforward and simple.

3.4 Optional RFT

After pruning, we perform RFT using a small subset of the data selected in Section 3.1, leveraging LoRA (Hu et al., 2022). This step is optional, as experimental results in section 4.7 show that DReSS without RFT still achieves competitive performance in perplexity and zero-shot accuracy.

4 Experiments

4.1 Experimental Setup

Implementation: All methods are implemented in PyTorch (Paszke et al., 2019), using the Hugging Face Transformers library (Wolf, 2019). All experiments are conducted on 80GB NVIDIA A100 GPUs. For fairness, we use llm-eval-harness (Gao et al., 2024a) to evaluate.

Datasets: For generation task, we evaluate the model's perplexity on WikiText-2 (Ashkboos et al., 2024). For zero-shot task, the benchmarks are PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2021), HellaSwag (Zellers et al., 2019), ARC-e and ARC-c (Clark et al., 2018). In Appendix H we use data from Alpaca (Taori et al., 2023), WikiText-2 (Merity et al., 2016), PTB (Marcus et al., 1993), and C4 (Raffel et al., 2020).

Models: We use models commonly adopted in the pruning domain including the LLaMA models (LLaMA2-7B, LLaMA3-8B, LLaMA2-13B) (Touvron et al., 2023; Grattafiori et al., 2024), OPT model (OPT-13B) (Zhang et al., 2022), and Phi-2 (Javaheripi et al., 2023).

Baselines: We evaluate DReSS against competitive structured pruning methods: LLM Surgeon (van der Ouderaa et al., 2024), SliceGPT (Ashkboos et al., 2024), and SLEB (Song et al., 2024).

Evaluation Metrics: The performance on generation task is measured by *perplexity*(Yao et al., 2022), while zero-shot tasks performance is evaluated using *accuracy* (Dong et al., 2024). The acceleration effects are represented by *throughput* and *latency* (Song et al., 2024; Zhang et al., 2024a).

4.2 Performance Comparison

To ensure a fair comparison, all methods were evaluated on 1,000 samples randomly selected from the WikiText-2 training set with a sequence length of 2048. We used a consistent data pool for DReSS's regularization, the calibration for LLM Surgeon,

Table 1: Performance comparison of different pruning methods. 'PPL' refers to the perplexity on WikiText-2. The accuracy (%) is reported on five zero-shot benchmarks. The best result is highlighted in **bold**, and the second-best is underlined. DReSS-ℓ_2 denotes the use of the ℓ_2-norm, while DReSS-ℓ_1 denotes the use of the ℓ_1-norm. The results below are all obtained after RFT.

Model	Method	PR	PPL(↓)	PIQA(↑)	WinoGrande(↑)	HellaSwag(↑)	ARC-e(↑)	ARC-c(↑)	Avg_Acc(↑)
Phi-2	Dense	0%	5.28	79.11	75.77	73.83	78.32	54.18	72.24
	LLM Surgeon	25%	7.26	67.28	63.25	54.24	51.62	35.85	54.44
	SliceGPT	25%	7.06	**69.32**	65.39	52.57	**53.78**	31.89	54.59
	SLEB	25%	7.82	67.94	62.79	49.80	48.55	29.34	51.68
	DReSS-ℓ_2	25%	**6.25**	68.52	**66.71**	56.73	52.78	**37.63**	**56.47**
	DReSS-ℓ_1	25%	6.28	68.67	65.32	**57.16**	52.39	37.28	56.16
LLaMA2-7B	Dense	0%	5.47	79.11	69.06	75.99	74.58	46.25	69.00
	LLM Surgeon	25%	7.38	70.59	65.87	58.66	63.65	38.33	59.42
	SliceGPT	25%	7.49	68.15	64.13	56.22	55.39	34.74	55.73
	SLEB	25%	10.24	63.25	62.36	53.77	55.82	32.24	53.49
	DReSS-ℓ_2	25%	5.86	73.18	**66.49**	61.73	**65.42**	**40.86**	**61.54**
	DReSS-ℓ_1	25%	**5.81**	**73.42**	65.73	**61.92**	65.26	39.68	61.20
LLaMA3-8B	Dense	0%	5.76	85.56	77.94	79.27	78.84	56.49	75.62
	LLM Surgeon	25%	7.62	76.34	70.18	71.46	69.65	49.22	67.37
	SliceGPT	25%	8.14	74.37	67.81	69.56	69.83	45.53	65.42
	SLEB	25%	11.37	71.68	62.96	66.37	64.61	43.28	61.78
	DReSS-ℓ_2	25%	**6.09**	78.29	**71.17**	73.28	**72.64**	**53.62**	**69.80**
	DReSS-ℓ_1	25%	6.12	**78.86**	70.28	**73.85**	72.37	53.02	69.68
OPT-13B	Dense	0%	10.12	76.82	64.80	69.81	61.87	35.67	61.79
	LLM Surgeon	25%	11.02	**74.18**	64.33	65.37	60.96	34.98	59.96
	SliceGPT	25%	10.90	73.72	64.28	63.33	60.59	34.66	59.32
	SLEB	25%	12.02	72.62	63.96	62.79	59.21	34.12	58.50
	DReSS-ℓ_2	25%	**10.38**	74.06	**64.57**	**66.82**	**61.56**	**35.33**	**60.47**
	DReSS-ℓ_1	25%	10.56	73.65	64.38	66.46	61.15	34.74	60.07
LLaMA2-13B	Dense	0%	4.88	80.47	72.22	79.39	77.48	49.23	71.76
	LLM Surgeon	25%	5.75	**77.75**	69.62	74.31	72.83	43.52	67.61
	SliceGPT	25%	6.16	69.69	68.45	63.73	63.46	39.90	61.05
	SLEB	25%	7.39	66.93	65.87	55.48	60.06	35.14	56.70
	DReSS-ℓ_2	25%	**5.12**	76.14	**71.22**	76.51	73.45	**46.87**	**68.84**
	DReSS-ℓ_1	25%	5.16	77.24	70.61	**76.75**	**73.84**	45.46	68.78

SliceGPT, and SLEB, and the RFT data for all methods, split at a 3:1 ratio (justified in Section 4.7). A uniform 25% pruning ratio was applied to all methods, implemented by setting the last 25% of the diagonal matrix R to 1. The choice of R is discussed in Section 4.4.

As shown in Table 1, DReSS excels in both *generation* and *zero-shot* tasks. On LLaMA2-7B, its perplexity is 20% lower than the second-best method, LLM Surgeon; on OPT-13B, its accuracy drops by only 1% compared to the dense model. Since the performance difference between the ℓ_2-norm and ℓ_1-norm versions is tiny, DReSS hereafter refers to the ℓ_2-norm version.

4.3 Acceleration Effectiveness

Language processing in LLMs comprises two primary stages: prompt processing, which is compute-bound, and token generation, which is memory-bound. We separately analyze the speedup achieved in each stage. Table 2 presents the throughput and latency results for OPT-13B and LLaMA2-13B, evaluated using a single 80GB NVIDIA A100 GPU. Following the methodology of previous work (Song et al., 2024), the token generation test involves generating sentences of 128 tokens with a batch size of 64, whereas for prompt processing, latency is measured by processing an input sequence of 2048 tokens.

At a pruning ratio of 50% on OPT-13B, DReSS delivers a 35% improvement in throughput and a 30% reduction in latency compared to the dense model. These results highlight the superior efficiency of DReSS in acceleration.

4.4 The Impact of Pruning Different Channels

Matrix $\mathbf{R}$ is used to select the channels on which regularization is applied (followed by pruning). We choose three options of $\mathbf{R}$ under 25% sparsity: (1) remove the matrix's last 25% rows or columns, (2) remove the first 25% rows or columns , (3) divide the matrix into 5 segments, removing each

Table 2: Comparison of throughput and latency under different ratios on OPT-13B and LLaMA2-13B. 'PPL' refers to the perplexity on Wikitext2, 'PR' represents 'pruning ratio', 'TI' represents 'throughput increase'.

Model	Method	PR	PPL(↓)	Tokens/s(↑)	TI(↑)	Latency(↓)	Speedup(↑)
OPT-13B	Dense	0%	10.12	1029	1.00×	386.5	1.00×
	DReSS	25%	10.38	1194	1.16×	319.42	1.21×
	DReSS	50%	13.85	1389	1.35×	274.11	1.41×
LLaMA2-13B	Dense	0%	4.88	1066	1.00×	396.9	1.00×
	DReSS	25%	5.12	1215	1.14×	330.8	1.20×
	DReSS	50%	7.59	1407	1.32×	285.5	1.39×

Table 3: The impact of pruning different channels on LLaMA2-7B. 'PPL' is the perplexity on WikiText-2. Avg_Acc is on five zero-shot benchmarks.

Cases	(1)	(2)	(3)
PPL(↓)	5.86	5.89	5.87
Avg_Acc(%)	61.54	61.39	61.46

segment's 5% of the last rows or columns. The results on LLaMA2-7B are shown in Table 3. The performance differences among (1), (2), and (3) are minimal, demonstrating that DReSS is insensitive to pruning different channels.

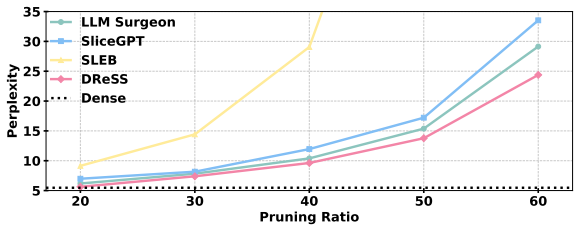

Figure 3: Perplexity of LLaMA2-7B on WikiText-2 using various pruning methods and ratios.

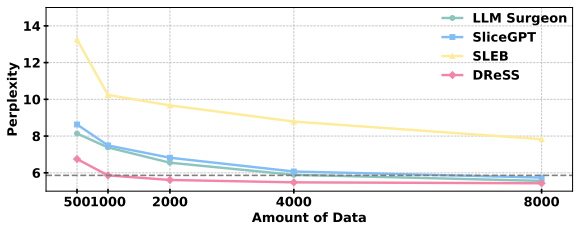

Figure 4: Perplexity of DReSS and other methods on WikiText-2 under different amount of data.

4.5 Robustness to Different Pruning Ratios

Keeping all other settings consistent with Section 4.2, we extend the pruning ratio from 20% to 60%. The perplexity of LLaMA2-7B under different methods are shown in Figure 3. DReSS significantly outperforms other methods across various pruning ratios. When the pruning ratio is up to 60%, SLEB collapses, while DReSS maintains relatively low perplexity compared to other pruning methods. This demonstrates that DReSS maintains robust performance even under extreme pruning ratios, enabling structured pruning to unlock significant potential for model acceleration. More detailed results are in Appendix G.

4.6 Minimal Overhead

We evaluated the perplexity of each method on LLaMA2-7B using varying amounts of data, keeping all other conditions consistent with Section 4.2 and only change the data size from 500 to 8,000. As shown in Figure 4, when only 1,000 samples were used for regularization and RFT, DReSS outperformed the other methods that used 4,000 samples. This further highlights the effectiveness and efficiency of applying regularization prior to pruning, as it transfers critical information in advance, enabling DReSS to get strong performance with minimal data overhead.

4.7 Ablation Study

As shown in Table 5, both pruning without regularization and regularizing all parameters degrade performance, increasing perplexity and reducing accuracy at 25% sparsity. This underscores the need to apply regularization specifically to the pruned components. Furthermore, RFT after pruning offers minimal performance impact, suggesting it is an optional step.

Data Ratio of Regualrization and RFT: With other settings consistent with Section 4.2, we only vary the data ratio $[4:1, 3:1, 2:1, 1:1, 1:2, 1:3]$. The lowest perplexity is achieved under 3:1 ratio. We speculate that pre-pruning regularization may be more critical than post-pruning RFT.

Trade-Off Between Language Modeling and Regularization: Keeping all other settings consistent with Section 4.2, we evaluate the model's performance by varying $\lambda \in [10^{-5}, 10^{-4}, 10^{-3}, 5 \times$

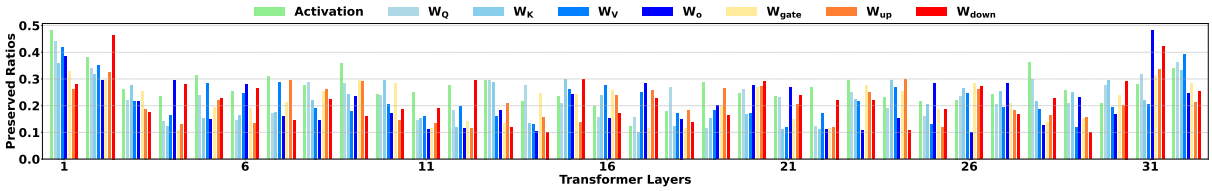

Figure 5: The ratio of the sum of absolute parameter values after regularization to that before regularization on LLaMA2-7B. For example, the absolute sum of the last 25% of rows in $\mathbf{W_Q}$ after regularization is divided by the sum of the corresponding rows before regularization to obtain the preserved ratios. This is applied similarly to other matrices $\mathbf{W_K}$, $\mathbf{W_V}$, $\mathbf{W_o}$, $\mathbf{W_{gate}}$, $\mathbf{W_{up}}$, $\mathbf{W_{down}}$, and layer activations.

Table 4: The model performance under different regularization settings. 'PPL' is the perplexity on WikiText-2. Avg_Acc is on five zero-shot benchmarks. Pruning ration is 25% and the LLM is LLaMA2-7B. We didn't perform RFT after pruning.

Cases	N&N&N	R&N&N	R&N&R	N&N&R	R&P&N	R&P&R	N&P&N	N&P&R
PPL(↓)	5.47	5.06	4.86	5.06	5.97	5.63	22.38	11.45
Avg_Acc(%)	69.00	70.39	71.23	70.39	59.56	62.35	47.25	52.73

$10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}]$. Optimal λ is 10^{-3}, demonstrating the importance of balancing language modeling loss and regularization loss. We listed the best λ for each model in Appendix E.

Table 5: Ablation results on LLaMA2-7B, LLaMA2-13B. 'FPR' is full parameter regularization, 'R' is regularization on selected channels. 'w/o' means removing specific parts.

Model	**Setting**	**PPL(↓)**	**AVG_ACC**
LLaMA2-7B	DReSS	5.86	61.54
	w FPR	12.68	50.26
	w/o R	22.38	47.25
	w/o RFT	5.97	59.56
LLaMA2-13B	DReSS	5.12	68.84
	w FPR	10.53	53.28
	w/o R	18.94	50.17
	w/o RFT	5.39	67.96

4.8 Impact of Regularization

Figure 5 shows that regularization reduces the weights and activations in selected portions to 30% of their original values, signifying a decrease in the information they contain. Conversely, as detailed in Appendix F, the values in the unregularized parts increase. This opposing trend confirms that regularization successfully transfers information from the pruned components to the remaining ones.

To evaluate regularization effectiveness, we divided into three stages: whether to regularize before pruning, whether to prune, and whether to regularize after pruning, totally 8 cases. For example,"N&P&R" denotes no regularization, pruning, then regularized. We used 25% sparsity LLaMA2-7B."N&N&R" and "R&N&N" are the same. The 750 (0.75×1,000) samples used for regularization are consistent with Section 4.2. As shown in Table 4, "R&P&N" and "N&P&R" outperform "N&P&N", "R&P&R" is the best, showing that regularization improves performance before, after pruning, and both of them. "R&P&N" outperforms "N&P&R", indicating that the paradigm of applying regularization before pruning may transfer important information to the remaining parts, thereby preserving model capacity.

5 Conclusion

In this paper, we propose a new pruning paradigm: apply regularization, prune, and finally RFT. Unlike previous paradigm that first prune and then apply RFT, DReSS transfers critical information from the pruned parameter space to the remaining parts during regularization, effectively mitigating the irreversible performance degradation caused by information loss. DReSS demonstrates superior performance in generation and zero-shot tasks, significantly outperforming existing methods. For instance, DReSS surpasses the powerful LLM Surgeon by 21% in perplexity on LLaMA2-7B. On OPT-13B, under 25% sparsity, the average accuracy drops by only 1%, while achieving 1.21× speedup compared to the dense model. Moreover, DReSS requires only 25% of the data to achieve comparable performance to previous methods, substantially reducing computational costs and the reliance on RFT. The new paradigm may provide insights for structured pruning in LLMs.

Limitations

In this paper, we only explore pruning at the channel level (i.e., rows or columns of parameter matrices). In future work, we plan to extend the pruning units to entire transformer layers. Using transformer layer as pruning unit is expected to yield more significant acceleration.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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

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

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

Arnav Chavan, Raghav Magazine, Shubham Kushwaha, Mérouane Debbah, and Deepak Gupta. 2024. Faster and lighter llms: a survey on current challenges and way forward. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 7980–7988.

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

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332.

Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Xinglin Pan, Qiang Wang, and Xiaowen Chu. 2024. Pruner-zero: Evolving symbolic pruning metric from scratch for large language models. In *Proceedings of the 41st International Conference on Machine Learning*. PMLR.

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

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024a. A framework for few-shot language model evaluation.

Shangqian Gao, Chi-Heng Lin, Ting Hua, Zheng Tang, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. 2024b. Disp-llm: Dimension-independent structural pruning for large language models. *Advances in Neural Information Processing Systems*, 37:72219–72244.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

Babak Hassibi, David G Stork, and Gregory J Wolff. 1993. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, and 1 others. 2023. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 1(3):3.

Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. 2022. The optimal bert surgeon:

Scalable and accurate second-order pruning for large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4163–4181.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. *Advances in neural information processing systems*, 2.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744.

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

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and B Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. *URL: https://github. com/huggingface/peft*.

Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

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

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

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

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Rajarshi Saha, Varun Srivastava, and Mert Pilanci. 2023. Matrix compression via randomized low rank and low precision factorization. *Advances in Neural Information Processing Systems*, 36.

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

Claudio Filipi Gonçalves Dos Santos and João Paulo Papa. 2022. Avoiding overfitting: A survey on regularization methods for convolutional neural networks. *ACM Computing Surveys (CSUR)*, 54(10s):1–25.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7059–7073.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. 2024. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. In *International Conference on Machine Learning*, pages 46136–46155. PMLR.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*.

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

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

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Tycho F. A. van der Ouderaa, Markus Nagel, Mart Van Baalen, and Tijmen Blankevoort. 2024. The LLM surgeon. In *The Twelfth International Conference on Learning Representations*.

Ziheng Wang. 2020. Sparsert: Accelerating unstructured sparsity on gpus for deep learning inference. In *Proceedings of the ACM international conference on parallel architectures and compilation techniques*, pages 31–42.

T Wolf. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Jinyang Wu, Mingkuan Feng, Shuai Zhang, Feihu Che, Zengqi Wen, and Jianhua Tao. 2024. Beyond examples: High-level automated reasoning paradigm in in-context learning via mcts. *Preprint*, arXiv:2411.18478.

Haojun Xia, Zhen Zheng, Yuchao Li, Donglin Zhuang, Zhongzhu Zhou, Xiafei Qiu, Yong Li, Wei Lin, and Shuaiwen Leon Song. 2023. Flash-llm: Enabling cost-effective and highly-efficient large generative model inference with unstructured sparsity. *Proceedings of the VLDB Endowment*, 17(2):211–224.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2024. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *The Twelfth International Conference on Learning Representations*.

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

Peng Xu, Wenqi Shao, Mengzhao Chen, Shitao Tang, Kaipeng Zhang, Peng Gao, Fengwei An, Yu Qiao, and Ping Luo. 2024. BESA: Pruning large language models with blockwise parameter-efficient sparsity allocation. In *The Twelfth International Conference on Learning Representations*.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Kumar Jaiswal, Mykola Pechenizkiy, Yi Liang, and 1 others. 2024. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. In *International Conference on Machine Learning*, pages 57101–57115. PMLR.

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

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, and 1 others. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Yang Zhang, Yawei Li, Xinpeng Wang, Qianli Shen, Barbara Plank, Bernd Bischl, Mina Rezaei, and Kenji Kawaguchi. 2024a. Finercut: Finer-grained interpretable layer pruning for large language models. *arXiv preprint arXiv:2405.18218*.

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

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1556–1577.

A Proof of the Proposition 1

Let us first assume that $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{B} \in \mathbb{R}^{n\times k}$. We can express $\mathbf{A}$ as $\begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n\end{bmatrix}$, where $\mathbf{a}_i \in \mathbb{R}^{m\times 1}$, and $\mathbf{B}$ as $\begin{bmatrix}\mathbf{b}_1^T \\ \mathbf{b}_2^T \\ \vdots \\ \mathbf{b}_n^T\end{bmatrix}$, where $\mathbf{b}_i \in \mathbb{R}^{k\times 1}$. Then, we obtain $\mathbf{C} \in \mathbb{R}^{m\times k}$ as shown in Equation 10. It is important to note that the final summation in Equation 10 refers to the summation of n rank-1 matrices, not a scalar summation.

$$\begin{aligned}\mathbf{C} = \mathbf{AB} &= \begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n\end{bmatrix} \times \begin{bmatrix}\mathbf{b}_1^T \\ \mathbf{b}_2^T \\ \vdots \\ \mathbf{b}_n^T\end{bmatrix} \\ &= \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i^T\end{aligned} \tag{10}$$

According to Equation 10, To keep the elements of $\mathbf{a}_i\mathbf{b}_i^T$ small, both $\mathbf{a}_i$ and $\mathbf{b}_i^T$ must be small. So, if regularization is applied to $\mathbf{a}_i$ and it should also be applied to the corresponding row $\mathbf{b}_i^T$ in $\mathbf{B}$.

Applying the above conclusion to the parameters of a transformer layer, if regularization is applied to certain columns of $\mathbf{W^{i-1}_{down}}$ in the FFN layer of the $(i-1)$th transformer layer, then, since $\mathbf{W^i_Q}$, $\mathbf{W^i_K}$, and $\mathbf{W^i_V}$ in the ith transformer layer are immediately multiplied by the result of $\mathbf{W^{i-1}_{down}}$, regularization should also be applied to the corresponding rows in $\mathbf{W^i_Q}$, $\mathbf{W^i_K}$, and $\mathbf{W^i_V}$. Additionally, if regularization is applied to certain columns of $\mathbf{W^i_o}$ in the ith transformer layer, the $\mathbf{W^i_{up}}$ in the FFN block will immediately multiply with it, so regularization should also be applied to the corresponding rows in $\mathbf{W^i_{up}}$. Thus, we have completed the proof of Proposition 1.

B Proof of Proposition 2

Step 1: Expressing ℓ_1-norm Using Elements. The objective function in the unconstrained prob-

lem is the ℓ_1-norm of the vector x, which is defined as:

$$||x||_1 = \sum_{i=1}^{n} |x_i| \tag{11}$$

This function aims to minimize the sum of the absolute values of the components of x.

Step 2: Reformulating the Constrained Problem

The constrained optimization problem introduces an auxiliary variable y, where for each element i:

$$x_i \geq -y_i \quad \text{and} \quad x_i \leq y_i \tag{12}$$

This implies that $y_i \geq |x_i|$, meaning each element of y serves as an upper bound for the absolute value of the corresponding element in x. Consequently, minimizing $|x|_1$ is equivalent to minimizing the sum of the elements in y. Thus, the objective function is defined as:

$$\mathbf{1}^T y \tag{13}$$

Thus, minimizing $\mathbf{1}^T y$ is equivalent to minimizing the sum of the absolute values of x, which is the ℓ_1-norm of x.

This transformation allows the optimization problem to be solved without directly involving the absolute value function, resulting in an equivalent constrained optimization problem that can be addressed via backpropagation. Thus, the proof of the Proposition 2 is complete.

C Completed Training Objective Function

The final training objective has two parts: language modeling loss $\mathcal{L}(\mathbf{W}, \mathbf{X})$, regularization loss. Regularization loss has three parts: Attention loss, FFN loss, remain loss. In Equation 7, for clarity, we only list one FFN layer, and principles for the others are similar. By using Proposition 2, the problem can be equivalently transformed into a constrained and differentiable optimization problem, which can be directly solved using the BP algorithm. The final objective function and constraints are as follows:

$$\begin{aligned}
\mathcal{L}_{\text{sum}} &= \mathcal{L}(\mathbf{W}, \mathbf{X}) + \sum_{i=1}^{l} \tilde{\mathcal{L}}_{\text{att}}^{i} \\
&\quad + \sum_{i=1}^{l} \tilde{\mathcal{L}}_{\text{FFN}}^{i} + \tilde{\mathcal{L}}_{\text{remain}} \\
\tilde{\mathcal{L}}_{\text{att}}^{i} &= \lambda(\mathbf{1}^T \mathbf{Y}_3^i \mathbf{1} + \mathbf{1}^T \mathbf{Y}_4^i \mathbf{1} \\
&\quad + \mathbf{1}^T \mathbf{Y}_5^i \mathbf{1} + \mathbf{1}^T \mathbf{Y}_6^i \mathbf{1}) \\
\tilde{\mathcal{L}}_{\text{FFN}}^{i} &= \lambda(\mathbf{1}^T \mathbf{Y}_1^i \mathbf{1} + \mathbf{1}^T \mathbf{Y}_2^i \mathbf{1}) \\
\tilde{\mathcal{L}}_{\text{remain}} &= \lambda\Big(\mathbf{1}^T \mathbf{Y_7} \mathbf{1} + \mathbf{1}^T \mathbf{Y_8} \mathbf{1} + \sum_{i=1}^{l} \mathbf{1}^T y_9^i \mathbf{1} \\
&\quad + \sum_{i=1}^{l} \mathbf{1}^T y_{10}^i \mathbf{1} + \mathbf{1}^T \mathbf{Y_{11}} \mathbf{1}\Big) \\
\text{s.t.} \quad & -\mathbf{Y}_1^i \leq \mathbf{R}\mathbf{W}_{\mathbf{up}}^{\mathbf{i}} \leq \mathbf{Y}_1^i, \\
& -\mathbf{Y}_2^i \leq \mathbf{W}_{\mathbf{down}}^{\mathbf{i}} \mathbf{R} \leq \mathbf{Y}_2^i, \\
& -\mathbf{Y}_3^i \leq \mathbf{R}\mathbf{W}_{\mathbf{Q}}^{\mathbf{i}} \leq \mathbf{Y}_3^i, \\
& -\mathbf{Y}_4^i \leq \mathbf{R}\mathbf{W}_{\mathbf{K}}^{\mathbf{i}} \leq \mathbf{Y}_4^i, \\
& -\mathbf{Y}_5^i \leq \mathbf{R}\mathbf{W}_{\mathbf{V}}^{\mathbf{i}} \leq \mathbf{Y}_5^i, \\
& -\mathbf{Y}_6^i \leq \mathbf{W}_{\mathbf{o}}^{\mathbf{i}} \mathbf{R} \leq \mathbf{Y}_6^i, \\
& -\mathbf{Y}_7 \leq \mathbf{W}_{\mathbf{emb}} \mathbf{R} \leq \mathbf{Y}_7, \\
& -\mathbf{Y}_8 \leq \mathbf{W}_{\mathbf{pos}} \mathbf{R} \leq \mathbf{Y}_8, \\
& -y_9^i \leq \mathbf{R}\alpha^{\mathbf{i}} \leq y_9^i, \\
& -y_{10}^i \leq \mathbf{R}\beta^{\mathbf{i}} \leq y_{10}^i, \\
& -\mathbf{Y}_{11} \leq \mathbf{R}\mathbf{W}_{\mathbf{lm}} \leq \mathbf{Y}_{11} \\
& \mathbf{Y}_1^i, \mathbf{Y}_2^i, \mathbf{Y}_3^i, \mathbf{Y}_4^i, \mathbf{Y}_5^i, \mathbf{Y}_6^i \geq 0, \\
& \mathbf{Y}_7, \mathbf{Y}_8, y_9^i, y_{10}^i, \mathbf{Y}_{11} \geq 0
\end{aligned} \tag{14}$$

D Detailed Implementation

In this part, we first introduce several hyperparameter settings, with the detailed results shown in Table 6. In our experiments, we employ FP16 precision for all evaluated models, including Phi-2, OPT-2.7B, LLaMA3-8B, OPT-13B, LLaMA2-7B, and LLaMA2-13B. For all RFT configurations, we set the LoRA rank r to 32, the scaling factor α to 10, and the sequence length to 2048. All other hyperparameters follow the default settings provided in the Hugging Face PEFT package (Mangrulkar et al., 2022). We set the batch size to 64. In future work, we will further explore a broader range of batch sizes. To ensure a fair comparison between DReSS and other methods, we maintain consistency in the data used across all approaches.

Table 6: Implementation Details

LoRA Rank	Scaling Factor	Max Sequence Length	Batch Size	Learning Rate	Early Stop Threshold
32	10	2048	64	2e-5	5

Specifically, the data used by DReSS for regularization, by LLM Surgeon for periodic updates of model weights and structures, by SliceGPT for selecting channel importance, and by SLEB for identifying crucial transformer layers are identical. Furthermore, we ensure that the data employed during the RFT process is consistent across all methods, thereby enabling a controlled and equitable evaluation framework. Following previous works (Ashkboos et al., 2024; Song et al., 2024), for the comparison unstructured pruning methods like Mangnitude, Wanda, and SparseGPT, we ensure that the data used to compute the importance of individual weights is the same as the data used by DReSS for regularization.

E Optimal λ for Different Models

Keeping all other settings consistent with Section 4.2, we evaluate the model's performance by varying $\lambda \in [10^{-5}, 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}]$.

we list λ for the best performance of each model in the Table 7:

F Changes in the Parts Without Regularization

As illustrated in Figure 6, the magnitude of the unregularized parameters exhibits an increase after the application of regularization, suggesting a redistribution of model capacity. This phenomenon indicates that during regularization, critical information, initially encoded in the regularized portion of the model, is partially transferred to the unregularized portion. In contrast, Figure 3 shows a reduction in the magnitude of the regularized parameters after regularization, implying that the imposed constraints effectively suppress the corresponding parameter values, enforcing sparsity or compression in that region.

Collectively, these observations suggest that the regularization process facilitates an implicit redistribution of information across different parameter subsets. Specifically, the regularization term promotes a shift of important model characteristics from the constrained (regularized) portion to the unconstrained (unregularized) portion, thereby preserving essential model knowledge despite the imposed sparsity constraints. Based on this insight, we posit that pruning the regularized portion post-regularization could mitigate information loss, as the core knowledge has already been migrated to the unregularized segment. This pruning strategy effectively reduces parameter redundancy while retaining the model's language modeling capacity, thereby achieving a more compact and efficient representation without compromising performance.

G Performance of DReSS under Different Pruning Ratios and Datasets

G.1 The Perplexity of DReSS under Different Pruning Ratios and Datasets

In Section 4.2, we utilize 1,000 samples randomly selected from the WikiText-2 dataset to guide the regularization process. Subsequently, we evaluate multiple large language models (LLMs) by measuring changes in perplexity across various generative task datasets, including WikiText-2, Alpaca, PTB, and C4, under pruning rates of 10%, 20%, 30%, 40%, 50%, and 60%. The detailed results, presented in Table 8, indicate that DReSS exhibits greater robustness as model scale increases, suggesting that the proposed method effectively mitigates performance degradation in larger architectures. This highlights the scalability of DReSS and its potential to maintain model efficiency under varying levels of sparsity.

G.2 The Accuracy of DReSS under Different Pruning Ratios on Zero-shot Tasks

To systematically evaluate the performance of DReSS on zero-shot tasks under varying pruning rates, we adopt the experimental setup outlined in Section 4.2, where 1,000 samples are randomly selected from the WikiText-2 dataset to guide the regularization process. We assess the accuracy of different model configurations at pruning rates of 10%, 20%, 30%, 40%, 50%, and 60% across a diverse set of benchmark datasets, including PIQA, WinoGrande, HellaSwag, ARC-e, and ARC-c. The results, summarized in Table 9, provide

Table 7: The optimal λ for each model.

Model	Phi-2	OPT-2.7B	OPT-6.7B	OPT-13B	LLaMA2-7B	LLaMA2-13B
λ	10^{-3}	5×10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}

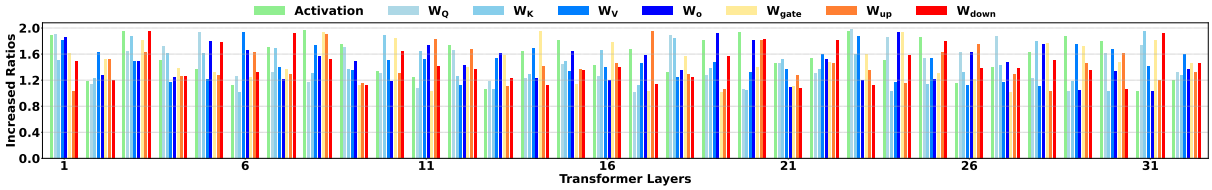

Figure 6: Ratio of the sum of absolute values of unregularized parameters after LLaMA2-7B regularization to the sum of absolute values of the corresponding parameters before regularization. For example, regularization is not applied to the first 75% of rows in $\mathbf{W_Q}$, and the absolute sum of the rows after regularization is divided by the sum of the corresponding rows before regularization to obtain the Increased Ratios. This is applied similarly to other matrices such as $\mathbf{W_K}$, $\mathbf{W_V}$, $\mathbf{W_o}$, $\mathbf{W_{gate}}$, $\mathbf{W_{up}}$, $\mathbf{W_{down}}$, and for layer activations, the sum of the unregularized columns is compared to the original sum.

insights into the impact of sparsity on zero-shot generalization. Notably, the analysis reveals that DReSS maintains competitive performance even at higher pruning rates, demonstrating its effectiveness in preserving reasoning and commonsense understanding across different tasks.

H Dependency on Calibration Dataset

Since DReSS relies on data-driven regularization, we investigate its dataset dependency. We evaluated perplexity of four methods on WikiText-2, using calibration and RFT data selected from Alpaca, WikiText-2, PTB, and C4. For fairness, we randomly selected 1,000 samples from each dataset, with other settings consistent with Section 4.2. As shown in Figure 7, DReSS consistently outperforms the other methods across datasets, demonstrating its robustness. When using Alpaca, WikiText-2, C4, and PTB as calibration and RFT data, the perplexity of various methods on WikiText-2, Alpaca, C4, and PTB is shown as follows:

Table 8: Perplexity comparison of DReSS with different pruning ratios. We set the pruning ratios to 10%, 20%, 30%, 40%, 50%, and 60%, and test the perplexity of the OPT and LLaMA2 models on the generation task datasets Alpaca, WikiText-2, PTB, and C4. For DReSS, we use the ℓ_2-norm.

Model	Pruning Ratio	WikiText-2	Alpaca	PTB	C4
OPT-2.7B	Dense	12.46	11.64	17.97	14.32
	10%	12.48	11.71	18.16	14.71
	20%	12.50	11.93	19.45	15.42
	30%	14.49	12.88	23.58	18.93
	40%	18.92	15.46	31.30	24.33
	50%	24.57	21.37	45.32	32.15
	60%	33.83	31.22	58.73	45.56
OPT-6.7B	Dense	10.85	10.27	15.77	12.71
	10%	10.91	10.45	16.05	13.18
	20%	11.02	10.83	17.58	14.45
	30%	12.32	11.46	19.65	16.68
	40%	14.26	12.71	25.52	20.94
	50%	19.63	15.66	33.78	28.22
	60%	28.75	21.39	47.29	39.47
OPT-13B	Dense	10.12	9.46	14.52	12.06
	10%	10.22	9.65	14.88	12.37
	20%	10.31	9.97	15.64	13.15
	30%	10.99	10.56	18.81	15.63
	40%	11.62	11.25	23.07	19.55
	50%	13.85	13.23	29.26	26.21
	60%	27.63	20.12	38.59	35.58
LLaMA2-7B	Dense	5.47	5.25	7.92	7.26
	10%	5.54	5.29	8.06	7.34
	20%	5.63	5.37	8.29	7.79
	30%	7.39	7.32	9.13	8.46
	40%	9.62	8.62	12.37	10.56
	50%	13.77	12.59	18.85	15.18
	60%	24.38	20.25	29.34	27.37
LLaMA2-13B	Dense	4.88	4.63	7.16	6.73
	10%	4.94	4.69	7.23	6.96
	20%	5.08	4.83	7.61	7.53
	30%	5.61	5.42	8.33	8.14
	40%	6.25	6.14	9.27	9.05
	50%	7.59	7.28	11.58	10.88
	60%	12.77	11.78	15.16	13.62

Table 9: Accuracy comparison of DReSS with different pruning ratios. We set the pruning ratios to 10%, 20%, 30%, 40%, 50%, and 60%, and test the accuracy of the OPT and LLaMA2 models on the zero-shot task datasets PIQA, WinoGrande, HellaSwag, ARC-e and ARC-c. For DReSS, we use the ℓ_2-norm. 'Avg_Acc' represents the average accuracy.

Model	Pruning Ratio	PIQA	WinoGrande	HellaSwag	ARC-e	ARC-c	Avg_Acc
OPT-2.7B	Dense	74.81	61.01	60.58	54.42	31.14	56.39
	10%	70.38	60.12	51.79	52.46	29.78	52.91
	20%	69.96	59.47	50.45	51.87	28.62	52.07
	30%	64.52	58.76	48.25	50.23	27.52	49.86
	40%	61.32	56.27	47.39	48.26	25.57	47.76
	50%	59.38	53.46	44.63	46.19	21.66	45.06
	60%	52.99	48.74	40.03	43.35	16.25	40.27
OPT-6.7B	Dense	76.39	65.19	67.16	60.14	34.64	60.70
	10%	75.89	64.61	64.69	58.53	33.47	59.44
	20%	75.12	64.23	62.56	57.34	32.95	58.44
	30%	72.52	62.63	58.27	54.48	29.99	55.58
	40%	67.37	58.59	52.12	49.38	26.87	50.87
	50%	61.48	55.62	46.46	47.68	22.97	46.84
	60%	55.73	51.52	43.38	45.85	18.62	43.02
OPT-13B	Dense	76.82	64.80	69.81	61.87	35.67	61.79
	10%	75.46	64.69	68.56	61.79	35.58	61.22
	20%	74.78	64.61	67.25	61.66	35.43	60.75
	30%	72.62	63.26	65.69	59.64	32.57	58.76
	40%	68.67	61.49	62.74	55.98	29.26	55.63
	50%	62.19	56.46	58.55	52.15	24.53	50.78
	60%	57.43	52.72	51.23	47.62	21.05	46.01
LLaMA2-7B	Dense	79.11	69.06	75.99	74.58	46.25	69.00
	10%	77.38	68.16	71.28	69.26	43.73	65.96
	20%	76.42	67.35	68.26	66.73	41.68	64.09
	30%	72.29	64.87	58.53	63.48	39.27	59.69
	40%	68.66	61.49	55.42	58.61	36.52	56.14
	50%	61.32	55.68	51.79	52.35	31.55	50.54
	60%	56.45	51.79	48.96	48.98	27.26	46.69
LLaMA2-13B	Dense	80.47	72.22	79.39	77.48	49.23	71.76
	10%	79.35	72.15	77.42	76.92	48.57	70.88
	20%	78.27	71.98	76.89	75.43	47.62	70.04
	30%	75.49	69.73	73.54	72.87	45.41	67.41
	40%	73.56	64.46	67.75	68.45	41.28	63.10
	50%	68.73	59.82	62.48	60.12	36.14	57.46
	60%	62.25	56.27	59.93	53.37	29.77	52.32

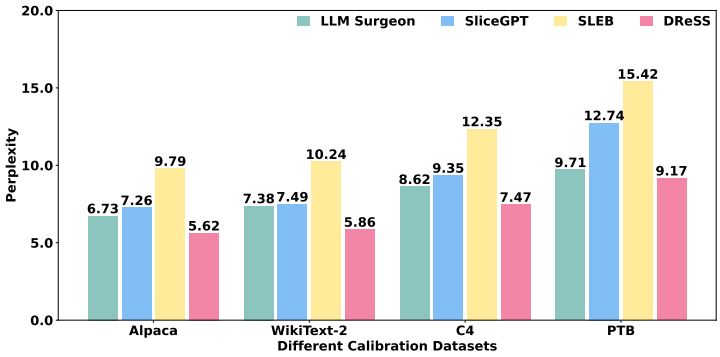

Figure 7: Comparison of perplexity on Wikitext-2 using different calibration datasets at a pruning ratio of 25% on LLaMA2-7B.

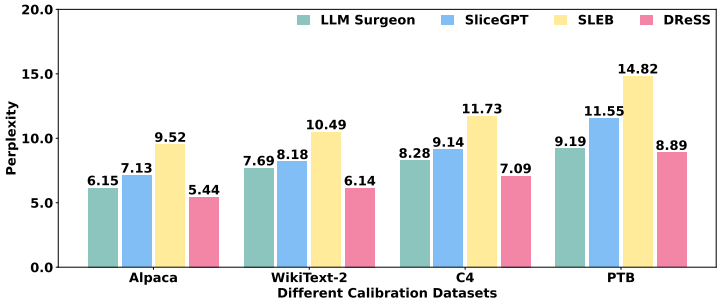

Figure 8: Comparison of perplexity on Alpaca using different calibration datasets at a pruning ratio of 25% on LLaMA2-7B.

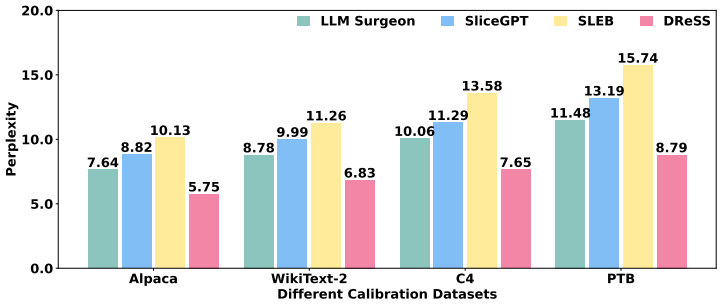

Figure 9: Comparison of perplexity on C4 using different calibration datasets at a pruning ratio of 25% on LLaMA2-7B.

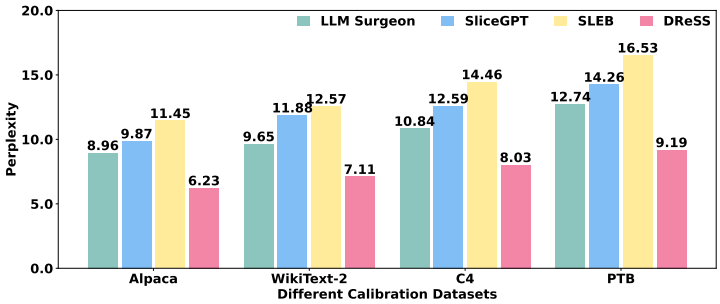

Figure 10: Comparison of perplexity on PTB using different calibration datasets at a pruning ratio of 25% on LLaMA2-7B.