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 **DReSS** (**D**ata-driven **Re**gularized **S**tructured **S**treamlining),

a novel framework that introduces pre-pruning regularization to structured pruning. To the best of our knowledge, we are the first to adopt this paradigm. As illustrated in Figure 1, DReSS operates in four steps: *First*, we randomly select a small subset of data from widely used benchmarks. Due to the small sample size, the computational overhead remains minimal. *Second*, we apply group regularization (ℓ_1 or ℓ_2) to the channels identified for pruning. The regularization process significantly suppresses the magnitude of the parameters in these channels, minimizing their information content and compelling the model to redistribute essential knowledge to the unpruned parts. This mechanism effectively mitigates the performance degradation typically caused by direct parameter removal. *Third*, we prune the regularized channels. *Finally*, we perform RFT on the pruned model using the data selected in the first step. Extensive experiments demonstrate that DReSS provides considerable acceleration and significantly outperforms existing methods in both perplexity and accuracy. 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; 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 introduce **DReSS**, a data-efficient structured pruning framework for Large Language Models (LLMs). As illustrated in Figure 1 and Algorithm 1, DReSS operates in three coherent stages: (1) **Regularization**, where we

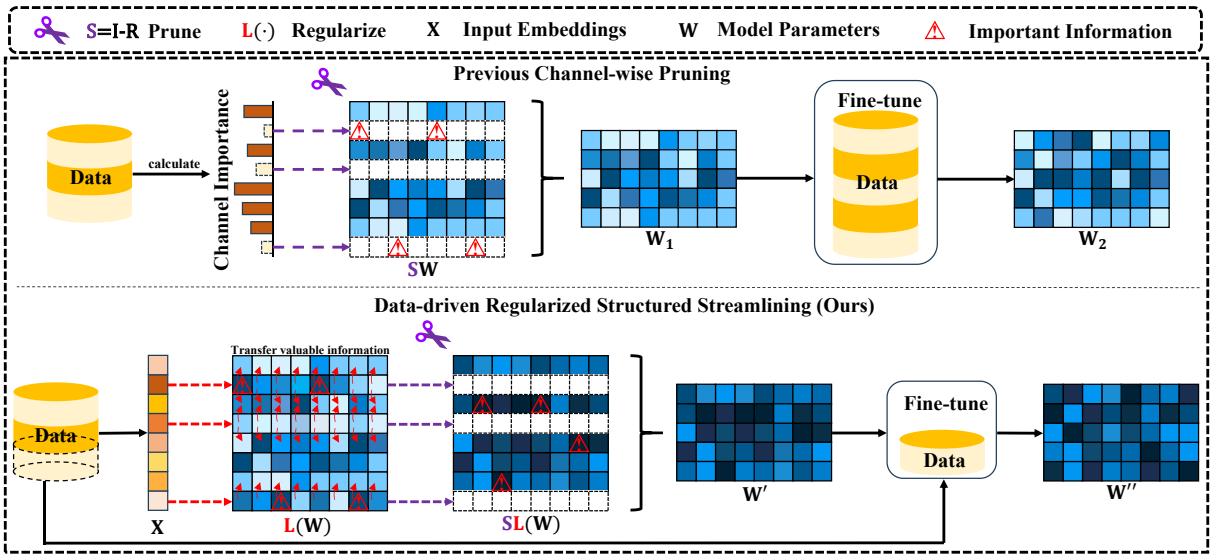

Figure 1: A comparison between previous channel-wise pruning methods and DReSS. Deeper blue square represents greater performance impact, the taller cylinder represents larger data volume. **Above**: Previous methods first select channels based on importance, followed by pruning and then do RFT. **Below**: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

Algorithm 1 DReSS Framework

Input: Dataset $\mathbf{X}$, Model $\mathbf{W}$, Pruning Ratio p, Layers L.
Construct global mask $\mathbf{R}$ based on p.
Split $\mathbf{X}$ into $\mathbf{X_{reg}}$ and $\mathbf{X_{rft}}$.
// $\mathbf{X_{reg}}$: *Regularization subset*
// $\mathbf{X_{rft}}$: *Recovery Fine-tuning subset*
// Stage 1: Regularization
while not converged **do**
  Compute $\mathcal{L}_{\text{LM}}(\mathbf{W}, \mathbf{X_{reg}})$
  Compute $\mathcal{L}_{\text{reg}}$ based on Eq. 2 using ℓ_1 or ℓ_2
  Update $\mathbf{W} \leftarrow \text{Optimizer}(\mathcal{L}_{\text{LM}} + \lambda\mathcal{L}_{\text{reg}})$
end while
// Stage 2: Pruning
$\mathbf{W}_{\text{pruned}} \leftarrow \text{Prune}(\mathbf{W}, \mathbf{R})$
// Stage 3: Optional RFT
$\mathbf{W}_{\text{final}} \leftarrow \text{LoRA\_Finetune}(\mathbf{W}_{\text{pruned}}, \mathbf{X_{rft}})$

identify coupled structures within the LLM and apply group regularization to suppress the parameters intended for pruning; (2) **Structured Pruning**, where the suppressed parameters are removed; and (3) **Optional Recovery Fine-tuning (RFT)**, which restores model performance.

3.1 Data Selection

To ensure efficiency and fairness, we utilize a small, randomly sampled subset from the widely used public dataset WikiText-2 (Merity et al., 2016) for both pre-pruning regularization and optional post-pruning RFT. Typically, only 1,000 samples drawn from the training set are required, making the process highly data-efficient.

3.2 Structural Dependency and Grouping

A key challenge in structured pruning is maintaining the validity of matrix multiplications after removing parameters. In a transformer architecture, removing a channel in one layer necessitates the removal of corresponding weights in connected layers. More details are in Appendix A.

Instead of treating each weight matrix independently, we define **dependency groups**. As shown in Figure 2, for a pruning ratio p, we define a binary diagonal mask matrix $\mathbf{R} \in \{0, 1\}^{d \times d}$, where indices corresponding to the pruned channels are set to 0. $\mathbf{R}$ is shared across all layers to ensure global consistency.

Based on the Transformer structure, the dependency requires that:

- **Attention Block:** If we regularize the columns of the output projection of the previous layer, we must simultaneously regularize the rows of the Query ($\mathbf{W_Q}$), Key ($\mathbf{W_K}$), and Value ($\mathbf{W_V}$) matrices this layer. Consequently, the output matrix $\mathbf{W_o}$ must be regularized column-wise to match the channel reduction.

- **FFN Block:** Similarly, the Up-projection ($\mathbf{W_{up}}$) and Down-projection ($\mathbf{W_{down}}$) are

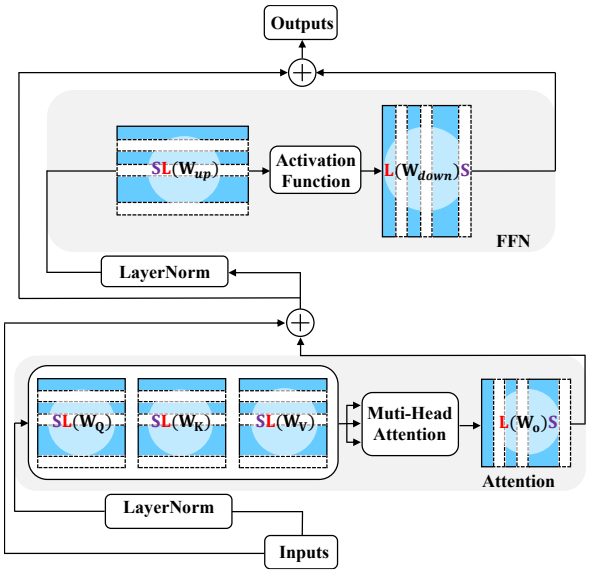

Figure 2: **Structural Dependency and Grouping.** Visualization of the coupled structures in a Transformer layer. The white rows and columns represent channels suppressed by the global mask $\mathbf{R}$, illustrating how regularization is applied consistently across Attention and FFN blocks to ensure structural integrity.

coupled. Regularizing rows in $\mathbf{W_{up}}$ necessitates regularizing columns in $\mathbf{W_{down}}$.

- **Residuals & Norms:** The embeddings ($\mathbf{W_{emb}}, \mathbf{W_{pos}}$), LayerNorm parameters ($\boldsymbol{\alpha}, \boldsymbol{\beta}$), and the final language modeling head ($\mathbf{W_{lm}}$) are also aligned with global mask $\mathbf{R}$.

3.3 Regularization Objective

To facilitate pruning, we enforce a regularization penalty on the channels selected by the pseudo-index selection matrix $\mathbf{R}$. Specifically, the non-zero diagonal entries in $\mathbf{R}$ indicate the channels intended for suppression. By penalizing the weights associated with these indices, we encourage the model to shift essential information to the remaining unregularized channels.

The total training objective combines the standard language modeling loss $\mathcal{L}_{\text{LM}}$ with the structured regularization loss. The overall loss function is defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{LM}}(\mathbf{W}, \mathbf{X}) + \lambda \left(\mathcal{L}_{\text{Attn}} + \mathcal{L}_{\text{FFN}} + \mathcal{L}_{\text{Misc}}\right) \tag{1}$$

where $\mathbf{X}$ is the input data, and λ is a hyperparameter controlling the regularization strength.

Based on the structural dependencies identified in Section 3.2, the regularization terms for the Attention blocks ($\mathcal{L}_{\text{Attn}}$), FFN blocks ($\mathcal{L}_{\text{FFN}}$), and remaining components ($\mathcal{L}_{\text{Misc}}$) are aggregated across all L layers:

$$\mathcal{L}_{\text{Attn}} = \sum_{i=1}^{L} \left(\sum_{\text{x} \in \mathbf{Q},\mathbf{K},\mathbf{V}} \|\mathbf{R}\mathbf{W}_{\text{x}}^{i}\| + \|\mathbf{W}_{\mathbf{o}}^{\mathbf{i}}\mathbf{R}\| \right) \tag{2}$$

$$\mathcal{L}_{\text{FFN}} = \sum_{i=1}^{L} \left(\|\mathbf{R}\mathbf{W}_{\mathbf{up}}^{\mathbf{i}}\| + \|\mathbf{W}_{\mathbf{down}}^{\mathbf{i}}\mathbf{R}\| \right) \tag{3}$$

$$\mathcal{L}_{\text{Misc}} = \|\mathbf{W}_{\mathbf{emb}}\mathbf{R}\| + \|\mathbf{W}_{\mathbf{pos}}\mathbf{R}\| + \|\mathbf{R}\mathbf{W}_{\mathbf{lm}}\| + \sum_{i=1}^{L} \left(\|\mathbf{R}\boldsymbol{\alpha}^{i}\| + \|\mathbf{R}\boldsymbol{\beta}^{i}\| \right) \tag{4}$$

Here, $\|\cdot\|$ denotes a vector norm applied to the matrix column-wise (or row-wise as indicated by the position of $\mathbf{R}$).

Optimization. We formalize the minimization of Eq. 1 as an optimization problem. Our framework supports both ℓ_2-norm and ℓ_1-norm for the regularization terms. When the ℓ_2-norm is used, the problem is directly differentiable. When the sparsity-inducing ℓ_1-norm is employed, the objective function is non-differentiable at zero. To address this, we transform the problem into a constrained formulation that can be efficiently solved using standard backpropagation. The detailed mathematical derivation and the equivalent constrained form are provided in Appendix B and Appendix C.

3.4 Pruning and Optional RFT

According to Figure 2, we prune the regularized rows and columns using the pseudo-index selection matrix $\mathbf{R}$. For clarity, we define the complementary binary mask $\mathbf{S} = \mathbf{I} - \mathbf{R}$, where diagonal entries of 1 denote parameters to be retained.

The pruning operations for the i-th layer Attention and FFN blocks are performed as follows:

$$\begin{aligned} \mathbf{W}_{\mathbf{Q}}^{\mathbf{i}\,\prime} &= \mathbf{S}\mathbf{W}_{\mathbf{Q}}^{\mathbf{i}}, & \mathbf{W}_{\mathbf{K}}^{\mathbf{i}\,\prime} &= \mathbf{S}\mathbf{W}_{\mathbf{K}}^{\mathbf{i}}, \\ \mathbf{W}_{\mathbf{V}}^{\mathbf{i}\,\prime} &= \mathbf{S}\mathbf{W}_{\mathbf{V}}^{\mathbf{i}}, & \mathbf{W}_{\mathbf{o}}^{\mathbf{i}\,\prime} &= \mathbf{W}_{\mathbf{o}}^{\mathbf{i}}\mathbf{S}, \\ \mathbf{W}_{\mathbf{up}}^{\mathbf{i}\,\prime} &= \mathbf{S}\mathbf{W}_{\mathbf{up}}^{\mathbf{i}}, & \mathbf{W}_{\mathbf{down}}^{\mathbf{i}\,\prime} &= \mathbf{W}_{\mathbf{down}}^{\mathbf{i}}\mathbf{S} \end{aligned} \tag{5}$$

Similarly, the remaining components, including embeddings, positional encodings, LayerNorm parameters, and the language modeling head, are pruned to maintain consistency:

$$\begin{aligned} \mathbf{W}_{\mathbf{emb}}{}' &= \mathbf{W}_{\mathbf{emb}}\mathbf{S}, & \mathbf{W}_{\mathbf{pos}}{}' &= \mathbf{W}_{\mathbf{pos}}\mathbf{S}, \\ \alpha^{\mathbf{i}\prime} &= \mathbf{S}\alpha^{\mathbf{i}}, & \beta^{\mathbf{i}\prime} &= \mathbf{S}\beta^{\mathbf{i}}, & \mathbf{W}_{\mathbf{lm}}{}' &= \mathbf{S}\mathbf{W}_{\mathbf{lm}} \end{aligned} \tag{6}$$

After pruning, we perform **Optional RFT** (Recovery Fine-Tuning) on the pruned model using the subset data $\mathbf{X}$ selected in Section 3.1, leveraging LoRA (Hu et al., 2022).

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 test set (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 used 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, 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 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.

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

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

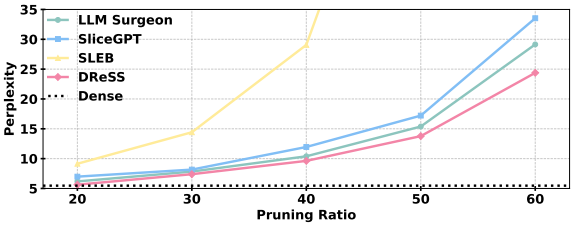

Figure 3: Perplexity of LLaMA2-7B on WikiText-2 using various pruning methods and ratios.

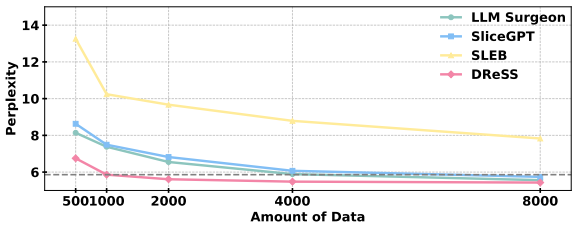

Figure 4: Perplexity of DReSS and other methods on WikiText-2 under different amount of data.

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

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×

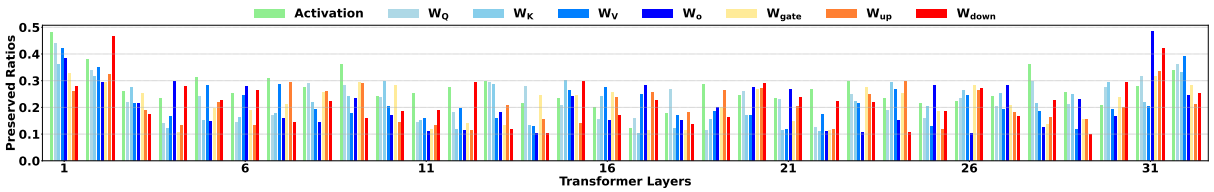

Figure 5: The ratio of the sum of absolute parameter values after regularization to that before regularization on LLaMA2-7B. For example, the absolute sum of the last 25% of rows in $\mathbf{W_Q}$ after regularization is divided by the sum of the corresponding rows before regularization to obtain the 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 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

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 4, 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.

Table 4: 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

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

4.8 Regularization Cost

To evaluate the computational overhead of our pre-pruning regularization, we conducted a comprehensive experiment on a single NVIDIA A100 (80 GB)

Table 5: 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

Table 6: Comparison of regularization cost and performance on Llama2-7B (25% Sparsity). All methods were evaluated on the same A100-80GB GPU. DReSS trades higher VRAM usage for the fastest training time and best performance.

Method	**Time** (↓)	**PPL** (↓)	**Avg_Acc** (↑)	**Peak VRAM** (↓)
LLM Surgeon	9h 08m	7.38	59.42	28 GB
SliceGPT	24.35m	7.49	56.20	**16 GB**
SLEB	19.82m	10.24	53.49	22 GB
DReSS	**13.44m**	**5.97**	**59.56**	69 GB

GPU. We report the peak VRAM usage for all evaluated methods. As shown in Table 6, although DReSS requires higher memory (69 GB) for optimizer states compared to inference-only baselines, it achieves remarkable efficiency, converging in just **13.44 minutes**. This corresponds to an orders-of-magnitude acceleration over metric-based methods like LLM Surgeon (which exceeds 9 hours due to expensive Hessian calculations) and a 1.8× speedup over SliceGPT. We contend that leveraging modern high-end GPU memory to achieve such substantial speed gains and superior performance (lowest PPL of 5.97) represents a highly advantageous trade-off for practical deployment.

4.9 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 5, "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

While DReSS demonstrates significant advantages in training efficiency and model performance, we acknowledge several limitations in its current form:

High Peak Memory Consumption. A primary limitation is the memory overhead during the regularization phase. Since DReSS involves full-parameter updates (requiring storage for gradients and optimizer states), it consumes significantly more VRAM (e.g., ∼69GB for Llama2-7B) compared to calibration-based methods that operate in inference mode. Consequently, our method relies on high-performance hardware (e.g., A100/A800 GPUs), which may limit accessibility for researchers with constrained computational resources.

Pruning Granularity. In this work, we focus exclusively on structured pruning at the channel level (i.e., removing rows or columns of parameter matrices). While this is effective for accelerating GEMM operations, we have not yet explored coarser-grained pruning strategies, such as removing entire transformer heads or layers. Extending DReSS to layer-level pruning could potentially yield larger reductions in inference latency and memory footprint, representing a promising direction for future work.

Scope of Model Architectures. Our current evaluation primarily focuses on dense Large Language Models (LLMs) like Llama-2. The applicability of DReSS to other specialized architectures, such as Mixture-of-Experts (MoE) where parameter redundancy manifests differently, requires further investigation.

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.

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 Structural Dependency

To formally establish the dependency between connected layers, let us consider two adjacent linear transformations represented by matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times k}$. Here, n represents the channel dimension (or hidden dimension) to be pruned.

We can decompose the matrix multiplication $\mathbf{C} = \mathbf{AB}$ into the sum of outer products. Let $\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \end{bmatrix}$, where $\mathbf{a}_i$ denotes the i-th column of $\mathbf{A}$, and $\mathbf{B} = \begin{bmatrix} \mathbf{b}_1^T \\ \mathbf{b}_2^T \\ \vdots \\ \mathbf{b}_n^T \end{bmatrix}$, where $\mathbf{b}_i^T$ denotes the i-th row of $\mathbf{B}$. The product is derived as:

$$\mathbf{C} = \mathbf{AB} = \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i^T \tag{7}$$

Equation 7 demonstrates that the information flow through the i-th channel is encapsulated in the rank-1 matrix $\mathbf{a}_i\mathbf{b}_i^T$.

Structural Pruning Constraint: To strictly reduce the channel dimension from n to $n-1$ (i.e., physical removal), one must simultaneously delete the i-th column of $\mathbf{A}$ and the i-th row of $\mathbf{B}$. Deleting only one side would result in a dimension mismatch for the subsequent matrix multiplication. Therefore, to induce sparsity consistent with this structural constraint, regularization must be applied jointly to the coupled group $\{\mathbf{a}_i, \mathbf{b}_i^T\}$.

Application to Transformer Architecture: In a Transformer block, the output of one component becomes the input of the next, linked via the residual stream.

1. **Inter-Layer Dependency:** Consider the connection between the FFN of the $(i-1)$-th layer and the Attention block of the i-th layer. The output is projected by $\mathbf{W_{down}^{i-1}}$ (acting as $\mathbf{A}$) and subsequently projected by $\mathbf{W_Q^i}, \mathbf{W_K^i}, \mathbf{W_V^i}$ (acting as $\mathbf{B}$). To prune a channel in the residual stream, we must regularize and remove the columns of $\mathbf{W_{down}^{i-1}}$ and the corresponding rows of $\mathbf{W_Q^i}, \mathbf{W_K^i}, \mathbf{W_V^i}$.

2. **Intra-Layer Dependency:** Similarly, within the i-th layer, the output of the Attention block is projected by $\mathbf{W_o^i}$. Due to the residual addition $\mathbf{x} + \text{Attn}(\mathbf{x})$, the output dimensions must match the input dimensions of the subsequent FFN block (specifically $\mathbf{W_{up}^i}$). Thus, the columns of $\mathbf{W_o^i}$ and the rows of $\mathbf{W_{up}^i}$ form a dependency group.

This logic extends recursively via residual connections, necessitating the global mask $\mathbf{R}$ defined in Section 3.

B ℓ_1-norm based approach

Step 1: Expressing ℓ_1-norm Using Elements. The objective function in the unconstrained problem is the ℓ_1-norm of the vector x, which is defined as:

$$||x||_1 = \sum_{i=1}^{n} |x_i| \tag{8}$$

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:

$$x_i \geq -y_i \quad \text{and} \quad x_i \leq y_i \tag{9}$$

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:

$$\mathbf{1}^T y \tag{10}$$

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.

C Completed Training Objective Function

The final training objective has two parts: language modeling loss $\mathcal{L}_{\text{LM}}(\mathbf{W}, \mathbf{X})$, regularization loss. Regularization loss has three parts: Attention loss, FFN loss, remain loss. By using the conclusions in Appendix B, 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:

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

D Detailed Implementation

In this part, we first introduce several hyperparameter settings, with the detailed results shown in Table 7. 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. 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 8:

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

Table 7: Implementation Details

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

Table 8: 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}

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 9, 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 10, provide 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 9: 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

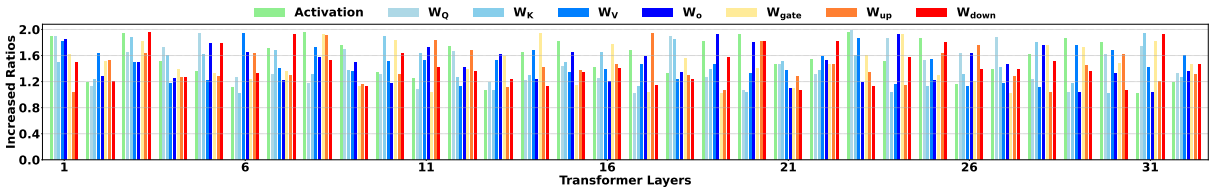

Figure 6: Ratio of the sum of absolute values of unregularized parameters after LLaMA2-7B regularization to the sum of absolute values of the corresponding parameters before regularization. For example, regularization is not applied to the first 75% of rows in $\mathbf{W_Q}$, and the absolute sum of 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.

Table 10: 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

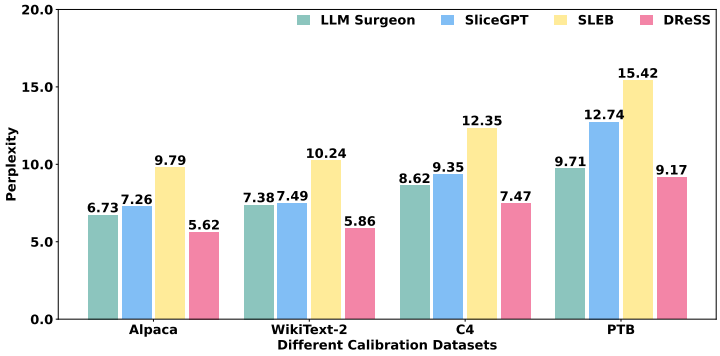

Figure 7: Comparison of perplexity on Wikitext-2 using different calibration datasets at a pruning ratio of 25% on LLaMA2-7B.

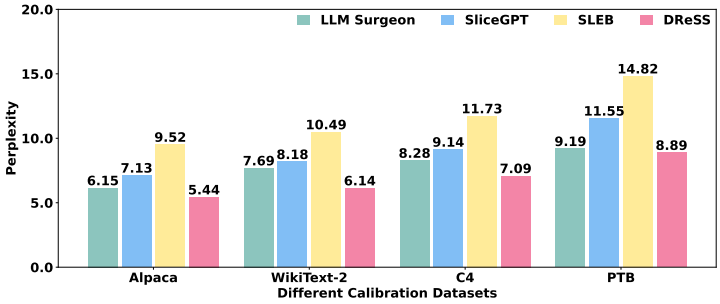

Figure 8: Comparison of perplexity on Alpaca using different calibration datasets at a pruning ratio of 25% on LLaMA2-7B.

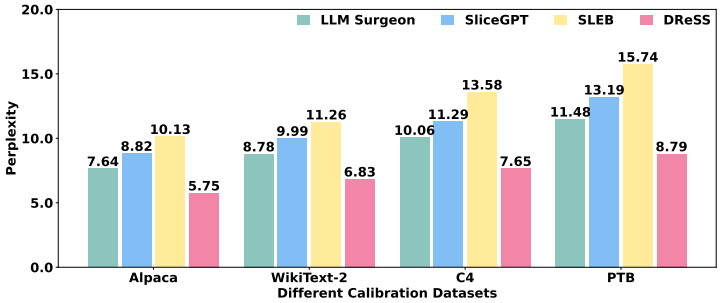

Figure 9: Comparison of perplexity on C4 using different calibration datasets at a pruning ratio of 25% on LLaMA2-7B.

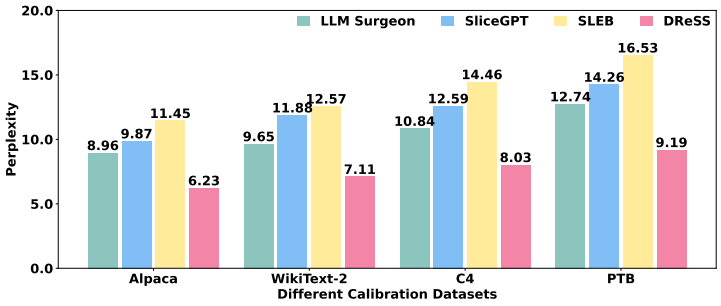

Figure 10: Comparison of perplexity on PTB using different calibration datasets at a pruning ratio of 25% on LLaMA2-7B.