

SPSR: Achieving Superior Large Language Model Layer Pruning Performance by Super-fast Recovery

Zhiguo Yang¹, Changjian Deng¹, Qinke Chen¹, Zijing Zhou¹, Jian Cheng¹

¹School of Information and Communication Engineering,
University of Electronic Science and Technology of China

Correspondence: chengjian@uestc.edu.cn

Abstract

Layer pruning aims to accelerate the inference of Large Language Models on general hardware by removing less important Transformer layers. However, naively deleting layers typically degrades model performance. To reduce this degradation, prior work approximates removed layers with replacement components such as linear layers. These approaches often require large datasets and heavy replacement modules, which increase computational and memory costs. Their time complexity also restricts them to static, consecutive-layer pruning. In this work, we introduce a novel method, Superior layer Pruning performance by Super-fast Recovery (SPSR), which uses Pearson correlation coefficients to quantify a layer’s recovery ability and applies a super-fast fine-tuning procedure with a lightweight replacement component. SPSR readily extends to non-consecutive pruning and to dynamic pruning. Experimental results show that SPSR achieves performance competitive with state-of-the-art methods while substantially improving recovery efficiency. Code is available at <https://github.com/BeiYazi0/SPSR>.

1 Introduction

Large Language Models (LLMs) have revolutionized the field of Natural Language Processing (NLP) (Brown et al., 2020; OpenAI, 2024; Touvron et al., 2023a). However, their massive parameter counts create severe deployment constraints due to limited memory and compute resources. Parameter pruning (Hassibi et al., 1993; LeCun et al., 1989; Han et al., 2015) has emerged as an effective compression strategy to address these constraints. Recent work has investigated layer redundancy in LLMs (Men et al., 2024; Song et al., 2024; Yang et al., 2024), showing that successive Transformer layers often produce highly similar outputs. Motivated by this observation, layer pruning identifies redundant layers and removes them, yielding

a model that differs only in depth and is therefore better suited for deployment.

Prior layer pruning approaches (Men et al., 2024; Song et al., 2024; Sieberling et al., 2025; Gromov et al., 2025; Kim et al., 2024) ignore activation mismatches introduced by layer removal, leading to performance degradation. To address this, recent work (Chen et al., 2025a; Shopkhoev et al., 2025; Chen et al., 2025b) proposes consecutive layer pruning with a recovery stage, in which several adjacent layers are removed simultaneously and a single component is trained to replace them. Since only one new component requires training, these methods require significantly fewer resources than fine-tuning approaches like LoRA (Hu et al., 2021). To accurately measure redundancy among consecutive layers while remaining invariant to magnitude, most methods employ cosine similarity.

However, these methods replace the removed layers with a linear layer, or even a Transformer layer, which introduces substantial computational or memory burden during training or inference and thus limits their applicability to non-consecutive layer pruning. Moreover, although cosine similarity is widely used to estimate the similarity between layer outputs (Men et al., 2024; Chen et al., 2025a; Gromov et al., 2025), it still lacks a convincing explanation for why cosine similarity effectively identifies redundant layers that can be recovered by replacing them with a single component.

Therefore, we identify two important questions that prior work has insufficiently examined:

- *Is there a simpler component that has sufficient capacity to recover while being easily applicable to non-consecutive pruning?*
- *How can redundant layers be accurately identified by theoretically analyzing their potential for performance recovery?*

To answer these questions, we first analyze the

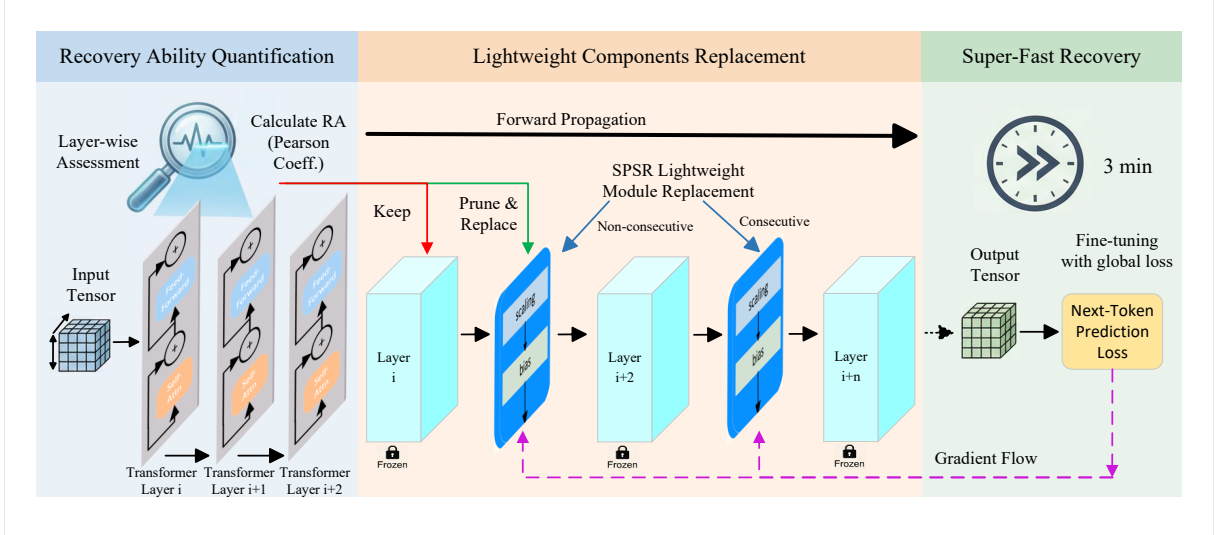


Figure 1: Illustration of SPSR. We first quantify each layer’s Recovery Ability (RA) using Pearson correlation coefficients. Layers with high RA are replaced by lightweight components containing few parameters. During recovery, all modules of the pruned model are frozen except for the SPSR replacement modules.

diagonalization of a linear layer that replaces consecutive layers identified by cosine similarity. This experiment demonstrates that off-diagonal weights have little effect on the layer’s transformation after training. Motivated by this observation, we simplify the replacement component to a single scaling vector and a bias vector. We then analyze the Recovery Ability (RA) of layers. By measuring the fit between output and input activations after transformation, we show that the difficulty of recovery is related to Pearson correlation coefficients.

In this paper, we introduce a novel method, Superior layer Pruning performance by Super-fast Recovery (SPSR), which uses the minimal Pearson correlation coefficient to measure recovery potential and employs a super-fast fine-tuning step with two vectors. The SPSR pipeline is illustrated in Figure 1. Since training is performed once using a global loss, SPSR naturally extends to non-consecutive pruning, including dynamic pruning.

The contributions of our work are as follows:

- We propose a novel metric, Recovery Ability (RA), that estimates a layer’s recovery potential by the minimum Pearson correlation coefficient across channels.
- We show that SPSR is a simple yet effective component for layer pruning. By optimizing a global loss, SPSR naturally extends to non-consecutive pruning. Comprehensive experiments show that SPSR achieves competitive performance while improving efficiency.

2 Related Work

2.1 Layer Pruning

Layer pruning, also known as depth pruning, is a special pruning strategy. Unlike general structured pruning (Ma et al., 2023; Khaki and Plataniotis; Kurtić et al., 2023; Meng et al., 2024), which target coupled components such as attention heads or neurons, layer pruning removes entire transformer layers from LLMs. Layer pruning approaches can be categorized as static or dynamic. Static methods (Men et al., 2024; Song et al., 2024; Sieberling et al., 2025; Gromov et al., 2025; Kim et al., 2024; Yang et al., 2024) use various metrics, such as cosine similarity, perplexity, and gradient magnitude, to assess layer redundancy. By contrast, dynamic layer pruning (Wee et al., 2025; Zhao et al., 2025) trains a router to produce removing layers at inference phase. PuDDing (Wee et al., 2025) prunes layers conditioned on specific prompts and trains a router to predict task likelihood over a limited set of candidate omission sets.

2.2 Lightweight Replacement Component Recovery

To mitigate the performance degradation caused by directly removing, common layer pruning methods (Gromov et al., 2025; Kim et al., 2024) employ Parameter-Efficient Fine-Tuning (PEFT) to recover model quality. However, RestoreLCC (Feng et al., 2025) indicates that PEFT are designed for dense models and neglect the distinct characteristics of

pruned models, often yielding suboptimal recovery. LLM-Streamline (Chen et al., 2025a) further demonstrates empirically that replacing removed layers outperforms LoRA despite using far fewer training resources. More recently, layer pruning with lightweight replacement component recovery (Chen et al., 2025a; Shopkhoev et al., 2025; Chen et al., 2025b) has emerged as a promising approach for compressing LLMs. Methods in this category identify the least important consecutive layers and replace them with a lightweight component. LLM-Streamline replaces consecutive layers with a Transformer layer and trains the replacement using MSE loss. ReplaceMe (Shopkhoev et al., 2025) estimates a linear transformation to approximate the pruned layers. Linear-Patch (Chen et al., 2025b) proposes a Hadamard transformation to suppress large outliers and a channel-wise scaling to align activation statistics, then fuses them into a single matrix to replace pruning layers. Despite these advances, prior work has focused primarily on consecutive layer pruning since recovery for non-consecutive remains more complex to train.

3 Preliminary

3.1 Layer Pruning metrics

LLMs typically use a Transformer decoder architecture with pre-norm normalization, comprising an embedding layer, a stack of Transformer decoder layers, and an LM head. We denote the l -th Transformer layer by $f(\mathbf{X}^l, \theta^l)$, where $\mathbf{X}^l$ and θ^l are the input activation and the parameters of the l -th layer, and $f(\cdot)$ denotes the layer transformation. Under the pre-norm architecture, the input of the $(l + 1)$ -th Transformer layer can be expressed as:

$$\mathbf{X}^{l+1} = \mathbf{X}^l + f(\mathbf{X}^l, \theta^l) \quad (1)$$

To identify redundant layers, researchers commonly use metrics such as cosine similarity, perplexity, and gradient-based scores.

Cosine Similarity. ShortGPT (Men et al., 2024) proposes measuring layer redundancy using Block Influence (BI), which quantifies the similarity between a layer’s input and output activations. The BI score for the l -th layer is calculated as follows:

$$\text{BI}^l = \mathbb{E}_{\mathbf{X}, t} \frac{(\mathbf{X}_t^l)^\top \mathbf{X}_t^{l+1}}{\|\mathbf{X}_t^l\|_2 \|\mathbf{X}_t^{l+1}\|_2} \quad (2)$$

where $\mathbf{X}_t^l$ denotes the hidden state of the t -th token in the l -th layer. A high BI score indicates

a large cosine similarity between layer’s input and output, thus suggest the layer’s redundancy.

Perplexity. Since Perplexity (PPL) directly reflects an LLM’s text generation capability, SLEB (Song et al., 2024) measures layer redundancy by evaluating PPL’s change introduced by layer removal. Layers that cause smaller increases in PPL are assumed to contribute less to performance and can therefore be prioritized for removal. The PPL importance score of the l -th layer is defined as:

$$I_{\text{PPL}}^l = \exp \left\{ -\frac{1}{ST} \sum_{s=1}^S \sum_{t=1}^T \log p_{\theta^l}(x_t^{(s)} | x_{<t}^{(s)}) \right\} \quad (3)$$

where p_{θ^l} denotes the pruned model with layer l removed, s and t index sequences and tokens, respectively. SLEB masks the remaining layers one at a time and removes the layer whose removal produces the smallest change in PPL. This procedure is repeated to remove multiple layers, making SLEB more time-consuming than most pruning methods.

LLM-Streamline (Chen et al., 2025a) demonstrates that PPL is highly sensitive to the data distribution. Methods that achieve low PPL on the calibration data perform poorly on other datasets. Chen et al. (2025c) supports this conclusion, showing that models pruned using cosine similarity perform better on downstream NLP tasks.

Taylor. Taylor metrics are commonly used to estimate a parameter’s importance by measuring the change in training loss $\mathcal{L}$ on calibration dataset $\mathbf{D}$ when the parameter is removed.

Shortened-LLaMA (Kim et al., 2024) propose the Taylor+ metric, the l -th layer’s importance can be formulated as:

$$I_{\text{Taylor}}^l = \sum_{\mathbf{W} \in \theta^l} \sum_i \sum_j \left| \frac{\partial \mathcal{L}(\mathbf{D})}{\partial \mathbf{W}_{i,j}} \mathbf{W}_{i,j} \right| \quad (4)$$

where second-order derivatives are omitted. A lower score indicates that a layer is less important and therefore a better candidate for removal.

3.2 Replacement Component Recovery

While most layer pruning methods adopt PEFT to recover performance of pruned models, recent works (Chen et al., 2025a; Shopkhoev et al., 2025; Chen et al., 2025b) show that replacing pruning layers with lightweight components is more efficient. These replacement approaches freeze the original

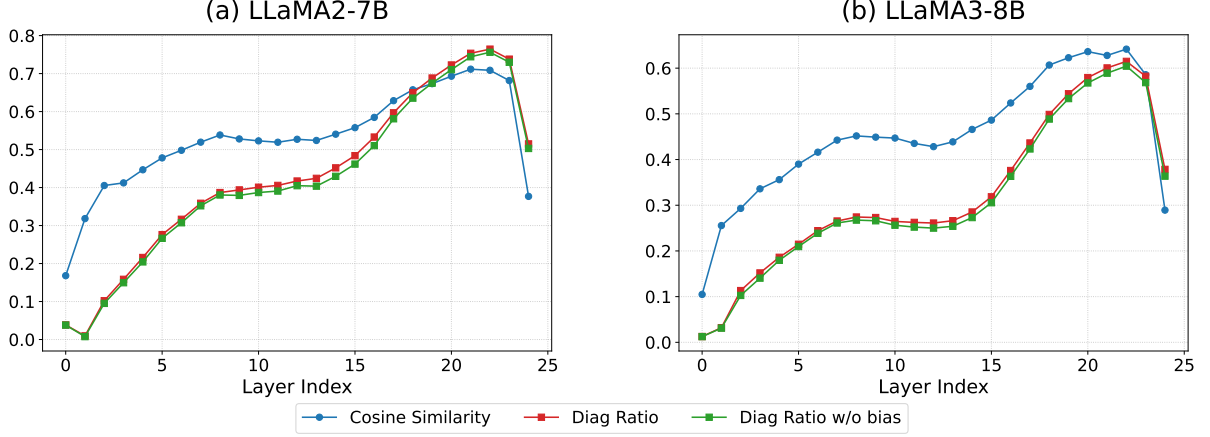


Figure 2: Comparison of cosine similarity and diagonal ratio when removing 8 layers. The layer index denotes the start index of the consecutive pruning layers.

model parameters and train only the lightweight components. Although they require fewer training resources compared to PEFT, they achieve comparable or better performance recovery.

Most layer pruning methods that use replacement component perform consecutive layer pruning, since removing non-consecutive layers requires more replacement components. Formally, the optimal pruning index i^* is determined by minimizing the distance between the hidden states before and after removing n consecutive layers:

$$i^* = \arg \min_i h(\mathbf{X}^l, \mathbf{X}^{l+n}) \quad (5)$$

where $h(\cdot)$ represents the importance metric, which is typically the cosine similarity. Rather than directly removing pruning layers, these methods replace them with a lightweight component and recover performance by minimizing the distance.

LLM-Streamline (Chen et al., 2025a) is a representative layer pruning method that follows this procedure. It replaces consecutive pruned layers with a Transformer layer, and trains that replacement using 30,000 samples randomly drawn from SlimPajama-6B. The resulting pruned model outperforms LoRA trained with 300,000 SlimPajama-6B samples. ReplaceMe (Shopkhoev et al., 2025) adopts a more flexible lightweight component, linear transformation, to minimize the cosine distance between $(\mathbf{X}^{i^*+n} - \mathbf{P}^{i^*})$ and $\mathbf{M}^{i^*}$, where $\mathbf{P}^{i^*}$ and $\mathbf{M}^{i^*}$ denote the input and output activation of the FFN in layer i^* . Linear-Patch (Chen et al., 2025b) applies a Hadamard transform and channel-wise scaling to suppress large outliers and align activation statistics, and fuses these operations into a

single linear layer. The lightweight component can be further refined by distillation, aligning the top-K output logit probability distributions.

Although ReplaceMe and Linear-Patch are faster, their performance lags behind LLM-Streamline, which limits their practical applicability. In addition, LLM-Streamline and ReplaceMe are unsuitable for non-consecutive layer pruning since their training objectives are local losses and fine-tuning time increases substantially for non-consecutive. Linear-Patch claims support for non-consecutive via training-free patches, but it overlooks the additional computation and memory costs imposed by multiple linear layers that cannot be absorbed by the remaining transformer layers. In the worst case, the number of added linear layers can equal the number of removed transformer layers.

4 Methodology

4.1 A simpler Lightweight Component

Prior work on replacement components has focused on using a transformation to approximate removed layers. Formally, let $\mathbf{X} \in \mathbb{R}^{N \times d_s, d}$ and $\mathbf{Y} \in \mathbb{R}^{N \times d_s, d}$ denote the input and output of the removed layers, where N is the batch size, d_s is the sequence length and d is the hidden size. A linear transformation $\mathbf{W} \in \mathbb{R}^{d, d}$ is then learned to minimize the discrepancy between $\mathbf{XW}$ and $\mathbf{Y}$. We further introduce a bias $\mathbf{b} \in \mathbb{R}^d$ to centralize $\mathbf{X}$ and $\mathbf{Y}$ and minimize the MSE:

$$\min_{\mathbf{W}, \mathbf{b}} \|\mathbf{XW} + \mathbf{b} - \mathbf{Y}\|_2^2 \quad (6)$$

The closed-form solution of Eq. (6) is $\mathbf{W}^* = (\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^\top \tilde{\mathbf{Y}}$ and $\mathbf{b}^* = \bar{\mathbf{y}} - \bar{\mathbf{x}} \mathbf{W}^*$, where $\tilde{\mathbf{X}}$ is

obtained by centering $\mathbf{X}$ per hidden dimension and $\bar{x} \in \mathbb{R}^d$ is the mean of $\mathbf{X}$. We introduce the notion of diagonal ratio to quantify the diagonal degree of $\mathbf{W}$, defined as the proportion of diagonal weight:

$$\text{Dr}(\mathbf{W}) = \|\text{diag}(\mathbf{W})\|_2^2 / \|\mathbf{W}\|_2^2 \quad (7)$$

where $\text{diag}(\cdot)$ extracts a matrix's diagonal elements. Intuitively, if the diagonal ratio of $\mathbf{W}$ is high, which shows the off-diagonal elements have only a minor effect, a radical but feasible approximation is to represent $\mathbf{W}$ using only a single vector.

To validate this approach, we prune eight layers of LLaMA models and compute the diagonal ratio using 128 samples randomly drawn from C4 for each pruning configuration. As shown in Figure 2, the diagonal ratio generally increases with depth but exhibits a sharp drop when the final layer is removed. The overall trend of the diagonal ratio curve matches the cosine similarity curve of consecutive layers (CL), suggesting that layers with high similarity can be approximated by a scaling vector. Furthermore, introducing b improves the diagonal ratio by roughly 1–2%, highlighting the bias's important role. Motivated by these observations, we propose a simpler component:

$$\min_{\mathbf{w}, \mathbf{b}} \|\mathbf{X} \times \mathbf{w} + \mathbf{b} - \mathbf{Y}\|_2^2 \quad (8)$$

where $\mathbf{w} \in \mathbb{R}^d$ is a vector that scales channels. To obtain more reliable results, we trained a linear layer replacement within the LLM-Streamline framework using the same experimental settings. **After training, the diagonal ratios are 90.68% for LLaMA2-7B and 88.65% for LLaMA3-8B.** This finding further supports our method of reconstructing removed layers using a single scaling vector and a bias vector.

We hypothesize that this phenomenon is not coincidental but rooted in the nature of residual architectures. In deep layers, the non-residual branch tends to degenerate into a "fine-tuning" mechanism rather than a feature extractor. To maintain the stability of the features learned in shallow layers, the transformation often converges to a simple affine transformation, which yields a diagonal-dominant weight matrix, where off-diagonal elements (representing complex channel mixing) become negligible compared to the diagonal scaling factors.

4.2 Recovery Ability (RA) Estimation

In the previous section, we evaluate the feasibility of a simpler replacement component but do not ad-

dress how to select layers for removal. Prior work commonly measures layer importance using cosine similarity. Although these studies report that cosine similarity metric is effective, it is not guaranteed to be appropriate for our replacement component. Hence, we introduce a new metric, Recovery Ability (RA), which estimate each channel's potential for performance recovery by measuring how well the centralized activation can be fitted:

$$\text{RA}_k = 1 - \frac{\|\tilde{\mathbf{X}}_{:,k} \times \mathbf{w}_k^* - \tilde{\mathbf{Y}}_{:,k}\|_2^2}{\|\tilde{\mathbf{Y}}_{:,k}\|_2^2} \quad (9)$$

where $\mathbf{w}^*$ and $\mathbf{b}^*$ (not appear due to the centered activations) denote the optimal solution of Eq. (8):

$$\mathbf{w}_k^* = \frac{\tilde{\mathbf{X}}_{:,k}^\top \tilde{\mathbf{Y}}_{:,k}}{\|\tilde{\mathbf{X}}_{:,k}\|_2^2}, \mathbf{b}_k^* = \bar{y}_k - \bar{x}_k \times \mathbf{w}_k^* \quad (10)$$

Hence, Eq. (9) can be simplified as:

$$\begin{aligned} \text{RA}_k &= \frac{-\|\tilde{\mathbf{X}}_{:,k}\|_2^2 \times (\mathbf{w}_k^*)^2 + 2\tilde{\mathbf{X}}_{:,k}^\top \tilde{\mathbf{Y}}_{:,k} \times \mathbf{w}_k^*}{\|\tilde{\mathbf{Y}}_{:,k}\|_2^2} \\ &= \frac{(\tilde{\mathbf{X}}_{:,k}^\top \tilde{\mathbf{Y}}_{:,k})^2}{\|\tilde{\mathbf{X}}_{:,k}\|_2^2 \|\tilde{\mathbf{Y}}_{:,k}\|_2^2} \\ &= \rho_k^2 \end{aligned} \quad (11)$$

where ρ_k denotes the Pearson correlation coefficient between $\mathbf{X}_{:,k}$ and $\mathbf{Y}_{:,k}$. The lower bound of RA can be formulated as:

$$\begin{aligned} \text{RA}_l &= 1 - \frac{\|\tilde{\mathbf{X}} \times \mathbf{w}^* - \tilde{\mathbf{Y}}\|_2^2}{\|\tilde{\mathbf{Y}}\|_2^2} \\ &= 1 - \frac{\sum_k \|\tilde{\mathbf{X}}_{:,k} \times \mathbf{w}_k^* - \tilde{\mathbf{Y}}_{:,k}\|_2^2}{\sum_k \|\tilde{\mathbf{Y}}_{:,k}\|_2^2} \\ &= 1 - \frac{\sum_k \|\tilde{\mathbf{Y}}_{:,k}\|_2^2 (1 - \rho_k^2)}{\sum_k \|\tilde{\mathbf{Y}}_{:,k}\|_2^2} \\ &\geq \rho_{\min}^2 \end{aligned} \quad (12)$$

where $\rho_{\min}^2$ denotes the minimum squared Pearson correlation coefficient across all channels.

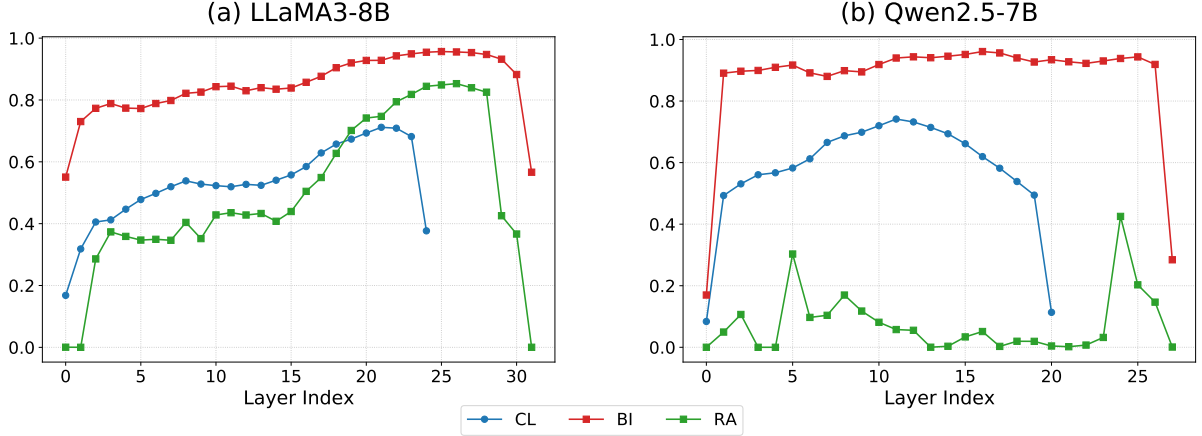


Figure 3: Comparison of cosine similarity and RA. The layer index denotes the start index of the consecutive pruning layers for CL. BI denotes the cosine similarity of each layer.

Table 1: Ablation of RA aggregation strategies on Qwen2.5-7B (8 layers pruned). Mean and median variants select identical layer.

RA Variant	Winogrande	HellaSwag	BoolQ	PIQA	OpenBookQA	ARC-E	ARC-C	Avg.
Min (ours)	62.12	58.01	74.53	66.81	31.80	61.28	37.20	55.96
Mean/median	62.19	58.33	62.51	70.08	36.00	57.62	37.12	54.83
P10	55.56	61.00	58.35	75.35	40.80	62.08	37.54	55.81

Even if all other channels are fully recovered, failure to recover the most difficult channel can still significantly alter the outputs due to layer normalization. We therefore **define RA as ρ_{min}^2** and remove layers with the largest RA in one shot. Since each SPSR component contains $2d$ parameters and all components are trained simultaneously using a global loss, we **do not require removals to be consecutive**. Nevertheless, when RA selects consecutive layers, they are replaced by a single component. Further analysis based on Eq. (6) is provide in Appendix A. The results indicate that this metric, which is based on a linear transformation, cannot measure recovery potential as accurately as RA.

Since ρ_{min}^2 may appear sensitive to noisy low-variance channels, we ablate this choice against mean, median, and 10th-percentile aggregations in Table 1. The minimum-based RA achieves the highest average accuracy, outperforming all variants. This supports our design intuition: the minimum statistic is more sensitive to worst-case, poorly recoverable channels, thereby better identifying layers that are truly safe to prune.

4.3 Training Loss

To simultaneously optimize multiple replacement components, we adopt a global training loss rather

than the methods used by LLM-Streamline or ReplaceMe. Although Linear-Patch demonstrates that minimizing Kullback–Leibler (KL) divergence is feasible, it requires restoring parts of the original output logits and is not well suited to low data regimes. Next-token Prediction loss is widely employed in training base LLMs since the complexity of compressing massive datasets facilitates emergent intelligence. We find that minimizing perplexity alone is sufficient to fine-tune our components. To accelerate training, we **cache the hidden input of the first removed layer**, thereby skipping part of the forward propagation. Training settings for SPSR are provided in Appendix F.

We further compare three alternative losses: MSE loss and cosine similarity loss computed between the original model output and the pruned model output at the final layer, as well as the KL divergence loss used in Linear-Patch. As shown in Table 2, perplexity achieves the best performance.

4.4 Empirical Study

Comparison between Cosine Similarity and RA.

As shown in Figure 3, RA exhibits clearer significance than cosine similarity: its values at the first and final layers are zero, indicating that model performance cannot be recovered without these lay-

Table 2: Mean accuracy under different training losses after pruning eight layers of LLaMA3-8B.

Loss	Accuracy(%)
MSE	58.36
Cosine similarity	59.14
KL Divergence	58.73
Perplexity	60.84

Table 3: Ablation study on the ingredients of SPSR over LLaMA3-8B with eight layers pruned.

	Accuracy(%)
SPSR	60.84
w/o training	58.53
w/o b	58.15
fixed w	60.29

ers. This observation is consistent with prior studies (Ma et al., 2023; Gromov et al., 2025), and consequently, most layer pruning methods avoid removing such layers. Moreover, the second subfigure shows that **layers with high RA are not necessarily consecutive**, revealing a limitation of existing replacement component recovery methods. Layers in LLaMA3-8B exhibit higher RA than those in Qwen2.5-7B, suggesting that LLaMA3-8B may benefit more from replacement-based recovery. This hypothesis is supported by the results in Table 6. To further examine the impact of different metrics, we combine SPSR with CL in subsequent experiments and apply the RA metric to other methods in Appendix B. The results show that selecting layers for removal based on RA can further improve the performance of other methods in some specific cases.

Resource Consumption Comparison. Table 4 compares the training configurations and resource consumption of SPSR with other replacement recovery approaches. Since our recovery component is extremely lightweight, the required training dataset can be substantially reduced, thereby lowering CPU memory consumption. Notably, SPSR achieves the shortest training time and moderate GPU memory usage due to its efficient training strategy. Although 3 minutes of training is sufficient and robust, experiments in Appendix F shows that **the training time of SPSR can be further reduced without performance degradation**. LLM-Streamline requires 30,000 training samples, leading to CPU memory consumption exceeding 1TB, which surpasses our hardware limitations. Consequently, we reduce its data size. However, it still incurs the highest training time and memory usage among the compared methods. For a detailed breakdown of training budgets, hyperparameters, resource consumption, and fairness validation, please refer to Appendix K. We emphasize that the differing sample sizes reflect intrinsic parameter

efficiency rather than unfair comparison.

Ablation Study. We ablate the contribution of individual components of SPSR, as reported in Table 3. Without training, the performance of pruned model drops by more than 2%, indicating the critical role of the fast recovery procedure. Without bias b , the assumptions in the previous analysis no longer hold, which substantially degrades performance. Without updating w , optimizing only b is insufficient to recover performance.

5 Experiments

Experimental Setup. To demonstrate the effectiveness of SPSR, we evaluate its performance across various LLMs, including LLaMA2 (Touvron et al., 2023b), LLaMA3 (Grattafiori et al., 2024), and Qwen (Bai et al., 2023). Our evaluation encompasses language modeling proficiency and zero-shot capabilities of pruned LLMs. Language modeling is measured by perplexity on the WikiText (Merity et al., 2017), PTB (Marcus et al., 1994), and C4 (Raffel et al., 2020) validation datasets. For zero-shot evaluation, we report accuracy on seven commonsense benchmarks from EleutherAI LM Harness (Gao et al., 2023), including BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), ARC-easy (Boratto et al., 2018), ARC-challenge (Boratto et al., 2018) and OpenbookQA (Mihaylov et al., 2018).

Implementation Details. Following prior work, we restrict the pruning ratio to below 30%. We denote the combination of SPSR with CL metric (consecutive layers cosine similarity) as SPSR (CL). All evaluation metrics are computed using the LM Harness. For BoolQ, PIQA, and WinoGrande, we report accuracy (Acc). For HellaSwag, ARC-Easy, ARC-Challenge, and OpenBookQA, we report normalized accuracy (Acc-Norm). To ensure statistical reliability, each experiment is run at least five times with different seeds, and the average result is reported.

5.1 Comparison of Replacement Component Recovery Methods

Table 5 and 6 report the average accuracy of pruned models across seven zero-shot tasks. Additionally, we provide detailed accuracy for each task in Appendix L. Overall, SPSR achieves the best performance in most settings, demonstrating its promise for more challenging downstream tasks. For exam-

Table 4: Comparison of different approaches in terms of resource consumption. Training time and memory usage were measured on LLaMA3-8B when removing eight layers. The training configurations for other methods are adopted from their open-source code.

Method	Metric	Training Data	Data size	Training Module	Trainig Method	Training Time	Memory Usage (GPU/CPU)
LLM-Streamline	Cosine Similarity	SlimPajama	3000	Transformer Layer	MSE Loss	227m	29.5g/106.7g
ReplaceMe	Cosine Similarity	SlimOrca	8000	Linear Layer	Cosine Similarity Loss	38m	15.6g/55.9g
Linear-Patch	Cosine Similarity	WikiText-2	2048	Linear Layer	KL-divergence	42m	17.4g /6.9g
SPSR	RA	C4	128	Vector	Next-token Prediction	3m	17.0g/2.2g

Table 5: Mean accuracy (%) on seven zero-shot tasks with various layer pruning methods over several models with six layers pruned. The best performance is set in bold, and the second best is set in underline.

Method	LLaMA 2-7B	LLaMA 3-8B	Qwen 2.5-7B	Qwen 3-8B
Dense	58.96	69.87	70.20	69.28
ShortGPT	54.19	60.59	56.57	54.86
CL	54.19	60.00	55.94	55.09
ReplaceMe	56.57	60.87	58.75	56.57
Linear-Patch	<u>56.09</u>	63.57	58.24	<u>61.03</u>
LLM-Streamline	55.50	64.80	<u>59.70</u>	59.64
SPSR(CL)	55.60	63.00	58.27	60.46
SPSR	55.53	<u>63.58</u>	60.31	61.56

Table 6: Mean accuracy (%) on seven zero-shot tasks with various layer pruning methods over several models with eight layers pruned. The best performance is set in bold, and the second best is set in underline.

Method	LLaMA 2-7B	LLaMA 3-8B	Qwen 2.5-7B	Qwen 3-8B
Dense	58.96	69.87	70.20	69.28
ShortGPT	51.41	39.92	51.58	48.80
CL	51.81	39.92	52.03	49.04
ReplaceMe	53.53	56.37	54.75	50.65
Linear-Patch	54.44	60.52	55.00	<u>54.80</u>
LLM-Streamline	<u>54.42</u>	61.83	55.15	50.49
SPSR(CL)	53.90	58.29	<u>56.22</u>	54.23
SPSR	54.16	<u>60.84</u>	56.34	55.42

ple, SPSR improves average accuracy by 1.19% on Qwen2.5-7B with 8 layers pruned and by 0.53% on Qwen3-8B with 6 layers pruned compared with other methods.

While SPSR consistently offers substantial gains, we recognize it may not always top the leaderboard in every single measure. Its performance is highly competitive with LLM-Streamline and it is most useful when users need a strong pruned checkpoint quickly with minimal recovery overhead, before direct use or fine-tuning. In this case, we view SPSR as a point on the efficiency–quality Pareto frontier, rather than universally dominating heavier methods. As shown in Appendix D, this advantage persists when coupled with fine-tuning: SPSR+LoRA

matches LLM-Streamline+LoRA in quality while saving a massive amount of total training time.

5.2 Combination with Non-consecutive Layer Pruning

Both LLM-Streamline and ReplaceMe are unsuitable for the non-consecutive pruning due to their local fine-tuning stage. Although Linear-Patch introduces additional computational and memory overhead, and its extra parameters can be comparable to those of a transformer layer in the worst case, we still include it as a baseline. For non-consecutive layer pruning methods, we select SLEB and Taylor+, which represent perplexity-based and gradient-based metrics, respectively. Although ShortGPT selects layers individually, the removed layers are always consecutive. Therefore, we do not include it in our comparison. Experiments are conducted on LLaMA2-7B and LLaMA3-8B, as reported in Table 7. Overall, SPSR yields significant improvements over all baseline methods. For example, when combined with SLEB On LLaMA3-8B with 8 layers pruned, SPSR decreases the WikiText PPL from 15.27 to 11.85, and improve the average accuracy from 51.54% to 57.28%.

We further apply SPSR to dynamic layer pruning. Pudding (Wee et al., 2025) trains a router to predict the task likelihood of prompts and applies the corresponding candidate omission set. Following its setup, we conduct experiments on Llama3.1-8B with 7 pruned layers, containing 10 candidates combined with SPSR. As shown in Table 8, SPSR successfully improve performance. These results demonstrate the promise of SPSR for non-consecutive layer pruning.

6 Conclusion

In this paper, we propose a novel method that replaces removed layers with simple components. Based on a theoretical analysis of fitting capacity, we compute channel-wise Pearson correlation coefficients and adopt the minimum value as the

Table 7: Perplexity on different validation datasets and accuracy (%) on seven zero-shot tasks with various non-consecutive layer pruning methods. L_p denotes the number of pruned layers and L_t denotes the total number of layers of the model. The best performance result is indicated in bold.

Model	L_p/L_t	Method	Perplexity↓			Accuracy(%)↑							
			WikiText	PTB	C4	WinoGrande	HellaSwag	BoolQ	PIQA	OBQA	ARC-e	ARC-c	Mean
LLaMA2-7B	8/32	SLEB	10.00	44.92	12.59	55.01	57.55	60.89	70.13	36.60	43.43	32.68	50.90
	8/32	+Linear-Patch	9.46	44.13	12.09	54.54	56.44	60.24	71.16	36.80	42.09	32.00	50.47
	8/32	+SPSR	8.93	38.52	11.47	55.80	60.51	61.74	72.14	36.80	45.79	32.34	52.16
	8/32	Taylor+	19.28	71.85	21.60	58.01	57.23	62.78	69.48	35.80	42.26	32.51	51.15
	8/32	+Linear-Patch	13.81	47.17	14.59	58.09	58.72	62.57	70.67	35.20	43.90	33.53	51.81
	8/32	+SPSR	10.90	37.61	11.89	58.72	61.29	65.81	72.36	36.40	44.15	33.28	53.14
	6/32	SLEB	8.07	36.89	10.70	60.30	62.64	63.30	72.96	38.20	47.05	33.70	54.02
	6/32	+Linear-Patch	7.86	36.53	10.52	59.35	61.76	65.75	73.12	37.80	46.25	33.53	53.94
	6/32	+SPSR	7.74	33.24	10.29	61.01	64.43	65.26	75.90	39.60	48.78	33.96	55.56
	6/32	Taylor+	14.07	51.99	18.10	61.96	63.02	63.24	71.16	36.60	43.14	35.07	53.46
	6/32	+Linear-Patch	11.32	40.11	13.36	62.98	63.90	63.00	72.31	38.00	44.65	35.24	54.30
	6/32	+SPSR	9.74	33.76	11.07	63.22	65.43	69.63	73.72	38.60	46.30	35.75	56.09
	8/32	SLEB	15.27	22.02	21.20	56.75	60.31	46.94	71.06	33.20	57.87	34.64	51.54
	8/32	+Linear-Patch	13.60	20.29	19.79	56.83	59.91	49.97	71.71	35.60	60.19	34.39	52.65
	8/32	+SPSR	11.74	17.35	17.59	58.96	63.97	63.27	73.78	37.00	65.99	37.97	57.28
	8/32	Taylor+	561.52	656.17	824.25	56.12	25.74	62.17	55.71	32.60	33.71	28.84	42.13
	8/32	+Linear-Patch	154.04	199.13	216.29	57.93	42.14	62.17	64.69	35.80	44.74	32.17	48.52
	8/32	+SPSR	16.17	29.90	20.42	64.64	66.29	66.54	74.10	37.60	64.94	39.16	59.04
LLaMA3-8B	6/32	SLEB	11.45	17.29	16.85	56.27	63.29	49.39	74.32	35.80	61.32	35.41	53.69
	6/32	+Linear-Patch	10.77	16.60	16.30	55.96	63.13	51.01	74.65	37.00	63.89	35.84	54.50
	6/32	+SPSR	9.59	14.71	14.76	58.96	68.52	64.25	77.20	40.40	69.87	40.96	60.02
	6/32	Taylor+	484.71	569.75	793.20	57.54	25.28	62.05	57.07	35.20	34.26	29.10	42.93
	6/32	+Linear-Patch	156.11	193.48	230.17	61.96	44.61	62.20	65.40	38.80	45.54	34.47	50.43
	6/32	+SPSR	17.20	32.58	20.60	70.88	70.16	78.01	74.43	38.60	68.27	45.14	63.64

Table 8: Accuracy (%) on seven zero-shot tasks using Pudding over LLaMA3.1-8B with 7 out of 32 layers pruned. The best performance result is indicated in bold.

Method	WinoGrande	HellaSwag	BoolQ	PIQA	OBQA	ARC-e	ARC-c	Mean
Pudding	64.40	63.61	67.40	72.80	34.80	66.37	41.64	58.72
+SPSR	65.51	65.47	64.71	74.10	39.40	65.53	40.02	59.25

Recovery Ability (RA) metric. By selecting layers with high recovery potential, our method achieves a favorable balance between performance and efficiency, achieving competitive performance with only minutes of additional training overhead. Moreover, the proposed approach generalizes effectively to non-consecutive layer pruning settings, including dynamic pruning. These results highlight the strong potential of SPSR for broad applicability across diverse layer pruning strategies.

Limitations

While SPSR achieves competitive performance compared with other replacement component recovery methods, its performance still lags behind that of the original dense models. Therefore, fine-tuning is necessary for pruned models using SPSR.

References

- Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.
- Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.
- Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, and 29 others. 2023. [Qwen technical report](#). *Preprint*, arXiv:2309.16609.
- Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. [Piqa: Reasoning about physical commonsense in natural language](#). *Preprint*, arXiv:1911.11641.
- Michael Boratko, Harshit Padigela, Divyendra Mikilineni, Pritish Yuvraj, Rajarshi Das, Andrew McCal-

- lum, Maria Chang, Achille Fokoue-Nkoutche, Pavan Kapanipathi, Nicholas Mattei, Ryan Musa, Kartik Talamadupula, and Michael Witbrock. 2018. [A systematic classification of knowledge, reasoning, and context within the ARC dataset](#). In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 60–70, Melbourne, Australia. Association for Computational Linguistics.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. [Language models are few-shot learners](#). *Preprint*, arxiv:2005.14165 [cs].
- Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. 2025a. [Streamlining redundant layers to compress large language models](#). In *The Thirteenth International Conference on Learning Representations*.
- Xinrui Chen, Haoli Bai, Tao Yuan, Ruikang Liu, Kang Zhao, Xianzhi Yu, Lu Hou, Tian Guan, Yonghong He, and Chun Yuan. 2025b. [A simple linear patch revives layer-pruned large language models](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Xinrui Chen, Hongxing Zhang, Fanyi Zeng, Yongxian Wei, Yizhi Wang, Xitong Ling, Guanghao Li, and Chun Yuan. 2025c. [Prune&comp: Free lunch for layer-pruned llms via iterative pruning with magnitude compensation](#). *arXiv preprint arXiv:2507.18212*.
- Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. [BoolQ: Exploring the surprising difficulty of natural yes/no questions](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.
- Zijian Feng, Hanzhang Zhou, Zixiao Zhu, Tianjiao Li, Chua Jia Jim Deryl, Mak Lee Onn, Gee Wah Ng, and Kezhi Mao. 2025. [Restoring pruned large language models via lost component compensation](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- 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. 2023. [A framework for few-shot language model evaluation](#).
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 90 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Gloriosi, and Dan Roberts. 2025. [The unreasonable ineffectiveness of the deeper layers](#). In *The Thirteenth International Conference on Learning Representations*.
- Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.
- Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. [Learning both weights and connections for efficient neural networks](#). *Preprint*, arxiv:1506.02626 [cs].
- Babak Hassibi, David Stork, and Gregory Wolff. 1993. [Optimal Brain Surgeon: Extensions and performance comparisons](#). In *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. [Lora: Low-rank adaptation of large language models](#). *CoRR*, abs/2106.09685.
- Samir Khaki and Konstantinos N. Plataniotis. [The need for speed: Pruning transformers with one recipe](#). *Preprint*, arxiv:2403.17921 [cs].
- Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. 2024. [Shortened LLaMA: A simple depth pruning for large language models](#). In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*.
- Eldar Kurtić, Elias Frantar, and Dan Alistarh. 2023. [ZipLM: Inference-aware structured pruning of language models](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 65597–65617. Curran Associates, Inc.
- Yann LeCun, John Denker, and Sara Solla. 1989. [Optimal Brain Damage](#). In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.
- Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, and Beidi Chen. 2023. [Deja vu: Contextual sparsity for efficient LLMs at inference time](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 22137–22176. PMLR.

- 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.
- Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. [The Penn Treebank: Annotating predicate argument structure](#). In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.
- 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](#). *Preprint*, arXiv:2403.03853.
- Xiang Meng, Shibal Ibrahim, Kayhan Behdin, Hussein Hazimeh, Natalia Ponomareva, and Rahul Mazumder. 2024. OSSCAR: One-shot structured pruning in vision and language models with combinatorial optimization.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. [Pointer sentinel mixture models](#). In *International Conference on Learning Representations*.
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. [Can a suit of armor conduct electricity? a new dataset for open book question answering](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.
- OpenAI. 2024. [GPT-4 technical report](#). *Preprint*, arxiv:2303.08774 [cs].
- 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.
- Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. [Yara parser: A fast and accurate dependency parser](#). *Computing Research Repository*, arXiv:1503.06733. Version 2.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. [Winogrande: An adversarial winograd schema challenge at scale](#). *Preprint*, arXiv:1907.10641.
- Dmitriy Shopkoev, Ammar Ali, Magauiya Zhussip, Valentin Malykh, Stamatios Lefkimiatis, Nikos Komodakis, and Sergey Zagoruyko. 2025. [Replaceme: Network simplification via depth pruning and transformer block linearization](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Oliver Sieberling, Denis Kuznedelev, and Dan Alistarh. 2025. [Evopress: Accurate dynamic model compression via evolutionary search](#). In *Sparsity in LLMs (SLLM): Deep Dive into Mixture of Experts, Quantization, Hardware, and Inference*.
- 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 *Proceedings of the 41st International Conference on Machine Learning*.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. [LLaMA: Open and efficient foundation language models](#). *Preprint*, arxiv:2302.13971 [cs].
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023b. [Llama 2: Open foundation and fine-tuned chat models](#). *Preprint*, arXiv:2307.09288.
- Juyun Wee, Minjae Park, and Jaeho Lee. 2025. [Prompt-based depth pruning of large language models](#). In *Forty-second International Conference on Machine Learning*.
- Yifei Yang, Zouying Cao, and Hai Zhao. 2024. [Laco: Large language model pruning via layer collapse](#). *arXiv preprint arXiv:2402.11187*.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. [HellaSwag: Can a machine really finish your sentence?](#) In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.
- Anhao Zhao, Fanghua Ye, Yingqi Fan, Junlong Tong, Jing Xiong, Zhiwei Fei, Hui Su, and Xiaoyu Shen. 2025. [SkipGPT: Each token is one of a kind](#). In *Forty-second International Conference on Machine Learning*.

A The Fitting Ability of Linear Transformation

A.1 Coefficient of Determination

Linear Transformation is widely used to replace removed layers in previous studies. Building on the analyses in Sections 4.1 and 4.2, we further derive a fitting capacity metric based on linear transformation.

The closed-form solution of Eq. (6) is

$$\begin{aligned} \mathbf{W}^* &= (\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^\top \tilde{\mathbf{Y}} \\ \mathbf{b}^* &= \bar{y} - \bar{x} \mathbf{W}^* \end{aligned} \quad (13)$$

where $\tilde{\mathbf{X}}$ is obtained by centering $\mathbf{X}$ per hidden dimension and $\bar{x} \in \mathbb{R}^d$ is the mean vector of $\mathbf{X}$. The centralized optimal prediction is given by $\hat{\mathbf{Y}} = \tilde{\mathbf{X}} \mathbf{W}^* = \tilde{\mathbf{X}} (\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^\top \tilde{\mathbf{Y}}$. We define the projection matrix $H = \tilde{\mathbf{X}} (\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^\top$, which orthogonally projects $\tilde{\mathbf{Y}}$ onto the column space of $\tilde{\mathbf{X}}$. Let $E = \tilde{\mathbf{Y}} - \hat{\mathbf{Y}} = (I - H) \tilde{\mathbf{Y}}$. Since E and $\hat{\mathbf{Y}}$ are orthogonal, it follows that $\|\tilde{\mathbf{Y}}\|_2^2 = \|\hat{\mathbf{Y}}\|_2^2 + \|E\|_2^2$. Accordingly, the fitting ability can be formulated as:

$$\begin{aligned} r &= 1 - \frac{\|\tilde{\mathbf{X}} \mathbf{W}^* - \tilde{\mathbf{Y}}\|_2^2}{\|\tilde{\mathbf{Y}}\|_2^2} \\ &= 1 - \frac{\|(I - H) \tilde{\mathbf{Y}}\|_2^2}{\|\tilde{\mathbf{Y}}\|_2^2} \\ &= 1 - \frac{\|\tilde{\mathbf{Y}}\|_2^2 - \|H \tilde{\mathbf{Y}}\|_2^2}{\|\tilde{\mathbf{Y}}\|_2^2} \\ &= R^2 \end{aligned} \quad (14)$$

where R^2 denotes the coefficient of determination, representing the proportion of $\mathbf{Y}$ that can be linearly explained by $\mathbf{X}$. A smaller R^2 indicates that the removed layer is more difficult to recover. We further compare this measure with the Recovery Ability (RA) metric computed using Eq. (12) and (14). As shown in Figure 4, similar to BI, R^2 exhibits little variation across layers. Moreover, R^2 tends to select shallow layers for removal, which contradicts the conclusions drawn from both BI and RA. This phenomenon can be attributed to the strong fitting capacity of linear transformation, which obscures the accurate assessment of recovery potential.

By contrast, the RA metric employs a simple formulation based on two vectors to avoid overfitting. With such a limited number of parameters, layers with high recovery potential can be estimated more accurately, as fitting becomes more challenging. The experimental results in Table 9 further support this assumption: when pruning eight layers from LLaMA3-8B, the layers selected by R^2 cannot be

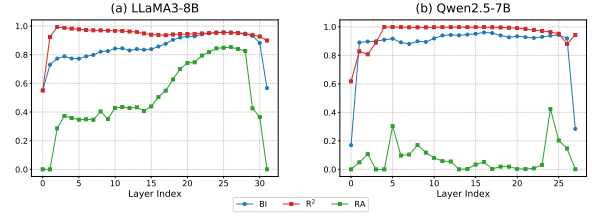


Figure 4: Comparison of R^2 and RA. BI denotes the cosine similarity of each layer.

effectively recovered, whereas those selected by RA can. Moreover, computing R^2 is more complex than computing the Pearson correlation coefficient. Therefore, we do not adopt R^2 as metric.

A.2 SPSR Modules vs. Full Linear Transformation

A full linear layer theoretically possesses higher expressive capacity than our scale-and-bias module. However, this capacity comes at the cost of requiring substantially more data and training time to estimate its parameters reliably. Under our designed "super-fast" recovery setting, which uses only 128 samples, a full linear matrix is heavily under-determined and prone to overfitting the noise in the small calibration set, ultimately leading to inferior recovery.

Furthermore, a practical advantage of our module emerges in non-consecutive pruning. SPSR often identifies disparate layers for removal, requiring multiple replacement components. Inserting numerous full linear layers would incur non-negligible inference latency and memory overhead. In contrast, our scale+bias structure is designed for extreme efficiency, both in terms of parameter count and data requirements, making it highly suitable for this scenario.

As shown in Table 10, under the identical 128-sample budget, the scale+bias structure consistently outperforms the full linear layer. These results highlight that under strict data and time constraints, a simpler, well-regularized structure is more effective. Therefore, we recommend the scale+bias structure for the rapid recovery phase to obtain a high-quality checkpoint at minimal cost. If further performance gains are desired and a larger dataset is available for a full fine-tuning phase, one can optionally substitute the scale vector with a full linear layer, warm-starting its diagonal from the learned scale values.

Table 9: Perplexity on different validation datasets and accuracy (%) on seven zero-shot tasks using SPSR combined with RA and R^2 metric. The best performance result is indicated in bold.

Method	Perplexity↓			Accuracy(%)↑							
	WikiText	PTB	C4	WinoGrande	HellaSwag	BoolQ	PIQA	OBQA	ARC-e	ARC-c	Mean
R^2	16.36	25.80	22.91	52.96	53.27	57.71	71.27	36.00	51.30	28.75	50.18
RA	14.16	18.88	20.23	71.67	64.56	78.23	70.73	35.20	64.02	41.47	60.84

Table 10: The mean accuracy(%) of zero-shot tasks and recovery time on LLaMA3-8B with 8 layers pruned under different samples setting.

Structure	Samples	Accuracy(%)	time
Dense	-	69.87	-
SPSR Modules	128	60.84	3m
Full Linear	128	59.75	5m
Full Linear	256	59.61	11m
Full Linear	512	60.63	26m

Table 11: Relative reconstruction error for affected layers (LLaMA3-8B, 8 layers pruned).

Setting	L29	L30	L31
No replace ($s=1, b=0$)	1.048	1.033	1.046
SPSR init (untrained)	0.891	0.982	0.917
SPSR (trained)	0.860	0.873	0.760

Our channel-wise affine replacement is motivated by the empirical observation that trained linear replacements are diagonally dominant. To provide direct evidence, we report the relative reconstruction error (L2 norm of output difference normalized by original output L2 norm) for the affected layers when pruning 8 layers from LLaMA3-8B. Table 11 shows that the trained SPSR module substantially reduces reconstruction error compared to both no-replacement and untrained initialization.

B Recovery Ability Metric Integration with Other Replacement Component Recovery Methods

To further validate the effectiveness of the RA metric, we integrate it with other replacement component recovery methods. In this setting, the layers to be removed are selected based on the RA metric. However, since RA evaluates layers independently, the selected layers may be non-consecutive. As a result, multiple linear transformations or even multiple transformer layers must be trained to recover

the pruned models. Moreover, the linear layers used in Linear-Patch are independent, which introduces higher computational and memory overhead than pruning consecutive layers. Similarly, LLM-StreamLine must remove additional transformer layers to offset the large parameter count of its replacement components.

Fortunately, experimental results show that the layers selected for removal by RA are consecutive in some LLMs, such as LLaMA3-8B. Therefore, we adopt these models for our experiments. As shown in Table 12, when selecting eight layers using RA, nearly all methods achieve performance gains. Notably, on LLaMA3-8B with ReplaceMe and Linear-Patch, RA yields substantial improvements, achieving 2.95% and 1.19% gains in zero-shot performance, respectively.

C Comparison with BI

To isolate the impact of the layer selection metric, we conducted a controlled experiment by integrating BI (Block Influence, the cosine similarity metric adapted for non-consecutive pruning) with our SPSR recovery module. As reported in Table 13, the RA metric consistently outperforms the cosine-based metric in non-consecutive pruning settings. This result validates that RA’s design, grounded in a per-channel recoverability analysis, provides a superior signal for identifying layers whose functionality can be effectively restored.

D Fine-tuning Performance

One motivation for SPSR is to provide a superior starting point for subsequent fine-tuning, not necessarily a fully restored final model. By rapidly bridging the activation gap, SPSR enables users to reach the fine-tuning stage almost immediately. As shown in Table 14, the entire SPSR+LoRA pipeline (taking approximately 4 hours in total) achieves fine-tuned performance competitive with LLM-Streamline+LoRA, despite the latter’s recovery stage alone consuming several hours. This speedup is critical for practical iteration cycles.

Table 12: Perplexity on different validation datasets and accuracy (%) on seven zero-shot tasks with various layer pruning methods combined with RA. The best performance result is indicated in bold.

Model	Method	Perplexity↓			Accuracy(%)↑							
		WikiText	PTB	C4	WinoGrande	HellaSwag	BoolQ	PIQA	OBQA	ARC-e	ARC-c	Mean
LLaMA2-7B	Linear-patch +RA	9.92	38.12	13.62	64.01	64.49	63.82	69.70	38.00	46.09	34.98	54.44
		10.15	40.50	13.94	64.56	63.14	71.74	70.18	37.40	44.32	35.84	55.31
	ReplaceMe +RA	13.09	40.80	15.34	62.43	59.54	68.20	68.44	37.00	45.29	33.79	53.53
		13.72	43.00	15.56	63.85	58.58	71.16	66.87	37.20	44.07	33.96	53.67
LLaMA2-7B	LLM-Streamline +RA	13.20	38.99	13.23	65.04	61.10	67.06	70.18	38.40	44.99	34.13	54.42
		11.12	35.57	12.56	65.75	60.64	69.05	70.02	36.80	45.29	34.98	54.65
	ReplaceMe +RA	18.67	26.63	28.30	68.35	60.33	69.17	69.15	32.20	58.50	36.86	56.37
LLaMA3-8B	Linear-patch +RA	16.86	21.64	24.16	70.72	64.09	75.81	69.80	32.00	63.17	39.68	59.32
		15.44	30.21	26.70	71.98	70.24	65.57	73.01	37.40	62.67	42.75	60.52
		14.63	23.50	24.97	71.90	69.01	78.84	71.98	35.80	61.62	42.83	61.71
LLaMA3-8B	LLM-Streamline +RA	13.47	15.94	19.05	71.35	66.68	76.94	71.49	37.40	65.49	43.43	61.83
		13.43	15.92	19.06	71.98	67.52	75.05	72.47	37.60	66.58	43.94	62.16
	ReplaceMe +RA	18.67	26.63	28.30	68.35	60.33	69.17	69.15	32.20	58.50	36.86	56.37

Table 13: Accuracy (%) on seven zero-shot tasks using RA or BI metrics with SPSR module. L_p denotes the number of pruned layers and L_t denotes the total number of layers of the model. The best performance is set in bold.

Model	L_p/L_t	Metric	WinoGrande	HellaSwag	BoolQ	PIQA	OBQA	ARC-e	ARC-c	Mean
LLaMA2-7B	8/32	BI	62.75	60.80	66.67	70.18	37.00	44.23	33.11	53.53
	8/32	RA	63.54	59.71	72.57	69.15	36.20	43.73	34.22	54.16
	6/32	BI	64.64	65.16	63.85	73.56	37.20	48.02	35.07	55.36
	6/32	RA	64.56	65.03	68.01	72.47	37.00	46.89	34.73	55.53
LLaMA3-8B	8/32	BI	72.30	65.78	59.02	72.03	35.00	63.51	40.36	58.29
	8/32	RA	71.67	64.56	78.23	70.73	35.20	64.02	41.47	60.84
	6/32	CL	73.01	70.03	72.84	75.73	38.60	70.41	44.45	63.58
	6/32	RA	73.01	70.03	72.84	75.73	38.60	70.41	44.45	63.58
Qwen2.5-7B	8/28	BI	57.38	60.31	53.39	73.83	38.40	58.16	37.20	54.10
	8/28	RA	62.59	57.55	74.34	67.74	34.00	61.87	36.26	56.34
	6/28	CL	56.99	65.94	64.04	75.30	39.60	63.43	41.38	58.10
	6/28	RA	64.40	62.72	77.13	70.24	36.80	67.55	43.34	60.31
Qwen3-8B	8/36	BI	62.83	55.91	77.61	66.81	32.60	54.76	37.46	55.42
	8/36	RA	62.83	55.91	77.61	66.81	32.60	54.76	37.46	55.42
	6/36	BI	64.96	61.56	81.59	70.08	33.60	63.13	42.15	59.58
	6/36	RA	68.43	61.60	86.02	71.60	35.80	64.35	43.09	61.56

Notably, for the LLaMA family, the SPSR+LoRA pipeline recovers zero-shot accuracy to levels comparable with the dense baseline at a sparsity of approximately 25%, highlighting its effectiveness in preserving model quality under a tight compute budget.

E Performance on generative and reasoning benchmarks

To assess whether SPSR affects capabilities beyond knowledge-intensive tasks, we evaluated the pruned Qwen2.5-7B and Qwen3-8B models on GSM8K, a multi-step mathematical reasoning benchmark. We further applied LoRA fine-tuning

(2 epochs on alpaca-cleaned) to examine whether SPSR provides a better initialization for downstream adaptation.

Note that alpaca-cleaned consists of general instruction-following samples and does not contain explicit reasoning chains. Consequently, any improvement on GSM8K after fine-tuning should be interpreted as recovery of the model’s underlying reasoning abilities that were disrupted by layer removal, rather than as newly learned mathematical skills. The fine-tuning primarily restores the instruction-following format and repairs the activation distribution, which in turn allows the existing reasoning capabilities to re-emerge.

Table 14: Accuracy (%) on seven zero-shot tasks using LoRA fine-tuning (2 epochs on alpaca-cleaned) with 8 layers pruned. The best performance is set in bold.

Model	Method	WinoGrande	HellaSwag	BoolQ	PIQA	OBQA	ARC-e	ARC-c	Mean
LLaMA2-7B	Dense	67.17	72.96	70.15	78.29	40.80	53.62	29.71	58.96
	LLM-StreamLine + LoRA	63.61	66.48	70.00	73.88	38.80	50.13	37.46	57.19
	SPSR + LoRA	64.88	65.80	75.81	71.93	37.00	49.54	37.63	57.51
LLaMA3-8B	Dense	72.61	79.16	81.59	79.65	45.00	77.69	53.41	69.87
	LLM-StreamLine + LoRA	72.06	71.94	81.41	75.35	39.40	72.90	47.78	65.83
	SPSR + LoRA	72.30	71.54	81.28	74.76	38.80	72.77	46.16	65.37
Qwen2.5-7B	Dense	73.01	78.95	85.11	78.78	47.20	77.40	50.94	70.20
	LLM-StreamLine + LoRA	57.14	65.70	62.84	76.66	41.20	65.95	41.98	58.78
	SPSR + LoRA	62.83	63.82	73.52	71.60	35.40	67.09	43.94	59.74
Qwen3-8B	Dense	67.64	74.96	86.64	76.88	41.40	80.93	56.48	69.28
	LLM-StreamLine + LoRA	65.67	63.74	84.89	71.82	35.60	64.90	44.80	61.63
	SPSR + LoRA	63.38	62.87	85.32	70.89	38.60	64.52	45.39	61.57

As shown in Table 15, layer pruning substantially degrades few-shot (8-shot) GSM8K accuracy. While SPSR alone already yields some recovery, SPSR + LoRA further reduces the gap toward the dense baseline and consistently outperforms other pruning methods under the same fine-tuning budget. These results suggest that SPSR creates a more favorable starting point for subsequent fine-tuning, even when the fine-tuning data does not directly target the evaluated reasoning skill.

Table 15: GSM8K accuracy(%) on Qwen models using various layer pruning methods. The best performance result is indicated in bold.

Model	Method	GSM8K	
		zero-shot	8-shot
Qwen2.5-7B	Dense	35.48	87.49
	ReplaceMe	1.90	2.43
	Linear-Patch	4.70	4.40
	LLM-Streamline	2.12	2.50
	SPSR	15.77	15.62
	SPSR + LoRA	31.99	38.44
Qwen3-8B	Dense	51.40	91.43
	ReplaceMe	2.05	2.35
	Linear-Patch	5.91	11.07
	LLM-Streamline	3.34	4.62
	SPSR	6.90	26.08
	SPSR + LoRA	44.81	62.47

We further extend our evaluation to multi-step commonsense reasoning (StrategyQA), abstractive summarization (XSum), and code generation (HumanEval). Table 16 presents the results on

Table 16: Performance on generative and reasoning benchmarks for Qwen2.5-7B with 8 layers pruned. Best results among pruning methods are in **bold**.

Method	StrategyQA	XSum	HumanEval
Dense	80.87	14.17	57.32
ReplaceMe	48.17	11.66	8.54
Linear-Patch	55.68	10.89	10.98
LLM-Streamline	53.49	12.22	5.49
SPSR	57.99	12.48	10.37

Qwen2.5-7B with 8 layers pruned. SPSR consistently outperforms ReplaceMe and achieves competitive or superior performance compared to heavier baselines across all three diverse capability dimensions, demonstrating that our lightweight recovery module effectively preserves complex generative and reasoning abilities.

F Training Settings of SPSR

In this section, we further discuss the training procedure and dataset settings of SPSR. All experiments are conducted on LLaMA3-8B with 8 layers removed.

F.1 Training Settings

Figure 5 illustrates how the number of training epochs and the learning rate affect SPSR. As the number of training epochs increases, the performance of the pruned model improves as expected. When the number of epochs exceeds 5, the final performance becomes nearly stable. Therefore, we adopt 10 training epochs for SPSR, which is sufficient.

Regarding the learning rate, it must be sufficiently large to ensure convergence. SPSR employs a simple replacement component with approximately $2d$ trainable parameters, where d denotes the hidden size of the LLM. Consequently, the dimensionality of the optimization problem is substantially lower than that of a linear layer or a transformer layer. The resulting loss landscape is relatively simple and smooth, with few local optima, and the basin around the optimum is likely to be wide. In such a low-dimensional and well-behaved space, larger update steps are less likely to overshoot the optimum. Instead, they can traverse flat regions more efficiently and move rapidly toward the optimal solution, thereby significantly accelerating convergence. Even if an update occasionally overshoots, the parameters can be easily corrected in subsequent optimization steps. Therefore, we adopt a learning rate of $3e-2$ for SPSR.

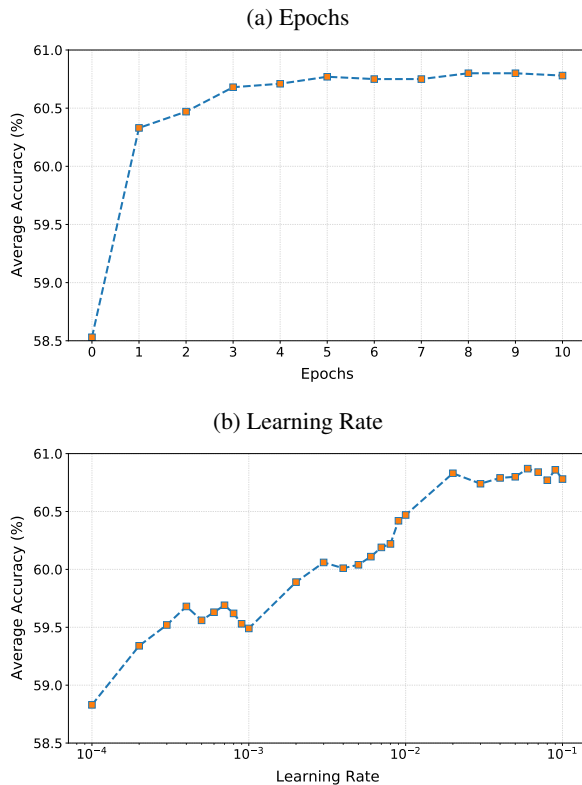


Figure 5: Mean accuracy across seven zero-shot tasks under different training settings after pruning eight layers of LLaMA3-8B using SPSR.

F.2 Dataset Settings

Figure 6 illustrates how the number of training samples and the sequence length of each sample affect

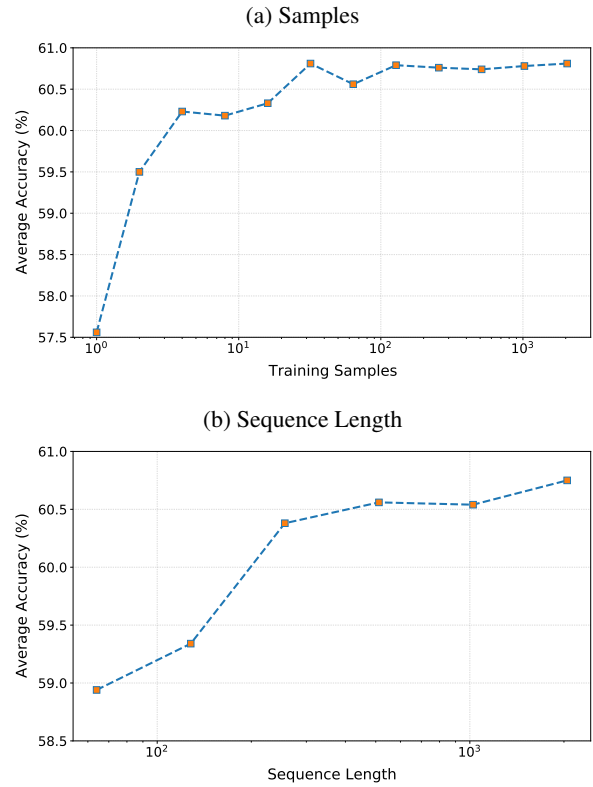


Figure 6: Mean accuracy across seven zero-shot tasks under different dataset settings after pruning eight layers of LLaMA3-8B using SPSR.

SPSR. For sequence length, we evaluate values of $[64, 128, 256, 512, 1024, 2048]$. When the sequence length exceeds 512, the performance gains from increasing the length diminish, while memory consumption increases substantially. Therefore, we adopt a sequence length of 2048 for SPSR.

For the number of training samples, we evaluate values of $[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048]$. Since SPSR contains at least $2d$ trainable parameters, it theoretically requires approximately $2d$ tokens to achieve convergence. Accordingly, the minimum number of training samples is $2d/d_s$, where d_s denotes the sequence length. For LLaMA3-8B, with $d = 4096$ and $d_s = 2048$, this theoretical minimum is 4 samples. However, empirical results show that SPSR converges only when the number of samples exceeds 32. This discrepancy arises from the greater complexity of Next-Token Prediction, which involves near-lossless compression of valid information, in contrast to the MSE loss used in local fine-tuning. To ensure robust performance, we therefore adopt

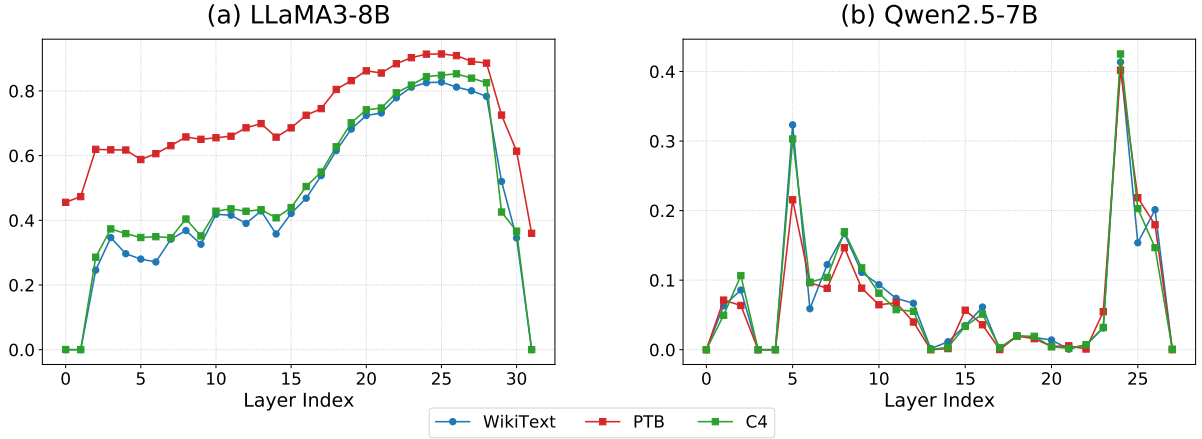


Figure 7: RA metrics under different datasets.

128 training samples for SPSR.

F.3 Initialization Settings

For both Linear-Patch and LLM-Streamline, the initialization of the replacement component plays a critical role. LLM-Streamline initializes the replacement component using the first layer among the removed layers, as this choice yields the lowest validation loss. Linear-Patch initializes a diagonal matrix by computing the $L1$ norm ratio and then combines it with Hadamard matrix. For SPSR, we investigate four initialization strategies: optimal, simple, magnitude and random initialization. Optimal initialization sets the parameters according to Eq. (10). Simple initialization assigns w with 1 and b with 0. Magnitude initialization sets w to the ratio of $L1$ norms and $b = 0$. Random initialization assigns both parameters with randomly sampled values.

We conduct experiments on LLaMA3-8B by removing eight layers to evaluate their impact. As shown in Table 17, the performance variance among optimal, simple and magnitude initializations are minor. These results indicate that SPSR is largely robust to initialization choices, with the exception of random initialization. Since an appropriate w can accelerate convergence and setting $b = 0$ helps mitigate overfitting, we adopt magnitude initialization for the SPSR components.

G Recovery Ability Metric across Various Dataset

In this section, we compare the RA metrics computed on different datasets, including WikiText,

PTB, and C4. As shown in Figure 7, the RA values across these datasets differ slightly while exhibiting the same overall trend. Since RA measures recovery complexity, which depends on both the layer parameters and the data distribution, such minor differences are expected. However, when more than eight layers are removed, RA may select different layers depending on the calibration dataset. Therefore, the training and calibration data should follow the same distribution. SPSR adopts the C4 dataset for both calibration and training.

H Performance on More LLMs

We further validate our method on both larger and smaller models, including LLaMA2-13B and LLaMA3.2-3B, as reported in Table 18. LLM-Streamline consistently achieves the best performance, owing to its large number of trainable parameters and substantial training data. Nevertheless, the performance gap between SPSR and LLM-Streamline is considerably smaller than the disparity in training resources. Given that SPSR is nearly 100× faster than LLM-Streamline, enabling minute-level recovery, the associated performance loss is acceptable, especially when subsequent fine-tuning can bridge that gap much more effectively. Moreover, SPSR achieves comparable or even superior performance to Linear-Patch and ReplaceMe while requiring significantly less training time and memory, which further demonstrates the effectiveness of SPSR.

Table 17: Accuracy (%) on seven zero-shot tasks of LLaMA3-8B using SPSR with several initialization settings.

Initialization	WinoGrande	HellaSwag	BoolQ	PIQA	OBQA	ARC-e	ARC-c	Mean
optimal	71.74	64.83	78.44	70.95	35.00	63.85	41.21	60.86
simple	71.90	64.42	77.86	70.73	35.40	63.93	41.38	60.80
magnitude	71.67	64.56	78.23	70.73	35.20	64.02	41.47	60.84
random	71.43	63.70	75.38	70.40	34.00	62.42	39.59	59.56

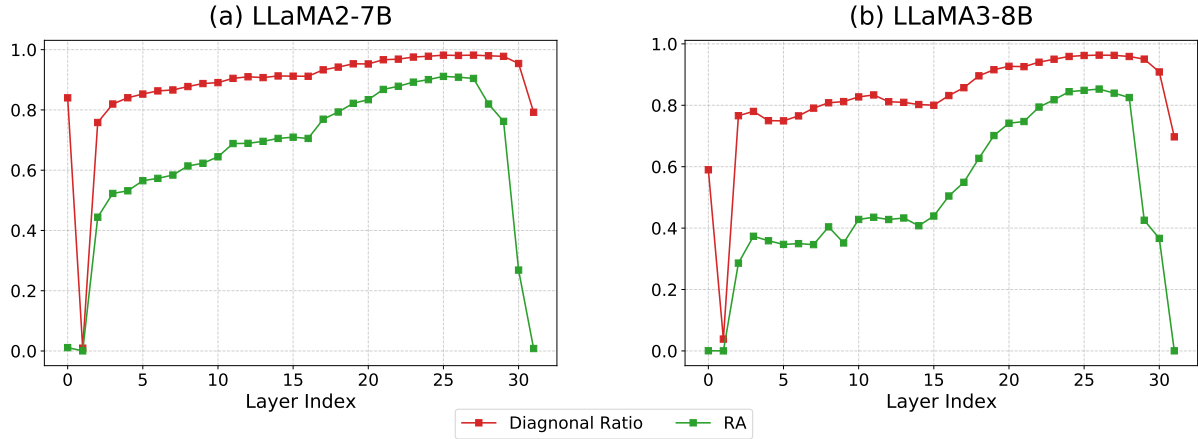


Figure 8: Diagonal ratio of each Transformer layer when approximated by the optimal weights that minimize MSE.

I Removing Layers Indices Comparison

To further illustrate the difference between CL (consecutive layer cosine similarity) and RA, Table 19 reports the layer indices selected for removal by each metric. On LLaMA2-7B, LLaMA3-8B, and Qwen3-8B, the two metrics identify several overlapping layers, indicating consistent redundancy in these layers. In contrast, RA selects substantially different layers on Qwen2.5-7B and achieves better performance, demonstrating its superior ability to measure the recovery potential of layers.

J Diagonal Ratio in Individual Transformer Layers

In Section 4.1, we demonstrated that multiple consecutive layers can be approximated by a single vector due to the high diagonal ratio of the replacing linear layer. To further establish the general applicability of SPSR to non-consecutive pruning, we measure the diagonal ratio of each Transformer layer when it is approximated by the optimal solution of Eq. (6). As shown in Figure 8, individual layers exhibit a high diagonal ratio, particularly in deeper layers. Although the diagonal ratio curve follows a trend similar to that of RA, the two measures do not fully align.

K Detailed Training Configurations and Fairness Analysis

SPSR has been shown to achieve performance competitive with other state-of-the-art methods, such as LLM-Streamline, while substantially improving recovery efficiency. However, there is no evidence that these heavier methods cannot achieve such performance in a low-data regime. Therefore, we evaluate LLM-Streamline on LLaMA2-7B using the same dataset setting as SPSR, denoted as LLM-Streamline(C4), with 128 samples from C4, to assess whether these methods truly require large datasets. As shown in Table 21, under the same dataset setting, LLM-Streamline(C4) performs far worse than SPSR and LLM-Streamline, indicating underfitting.

Therefore, we evaluated all baselines using their original, well-tuned configurations as reported in their respective papers. Table 20 provides a comprehensive breakdown of training budgets, hyperparameters, and resource consumption. The differing sample sizes across methods are not an artifact of unfair comparison but reflect intrinsic parameter efficiency: methods with larger trainable modules (e.g., LLM-Streamline) require more calibration data to avoid underfitting, whereas SPSR’s mini-

Table 18: Perplexity on different validation datasets and accuracy (%) on seven zero-shot tasks with various layer pruning methods at 25% sparsity. The best performance is set in bold, and the second best is set in underline.

Model	Method	Perplexity↓			Accuracy(%)↑							
		WikiText	PTB	C4	WinoGrande	HellaSwag	BoolQ	PIQA	OBQA	ARC-e	ARC-c	Mean
LLaMA2-13B	Dense	4.88	34.41	6.99	69.69	76.59	67.25	78.67	42.00	57.87	44.20	62.32
	ShortGPT	20.03	56.10	21.39	62.98	65.41	62.26	71.33	37.40	46.04	37.71	54.73
	CL	9.77	57.17	13.20	64.88	66.24	62.11	70.73	40.00	48.32	37.46	55.68
	ReplaceMe	8.25	53.67	11.29	68.51	66.87	62.42	72.14	37.80	51.22	38.65	56.80
	Linear-patch	7.35	<u>40.20</u>	<u>10.62</u>	67.88	69.93	62.14	73.78	39.00	46.89	38.40	<u>56.86</u>
	LLM-Streamline	<u>7.63</u>	38.84	10.23	69.53	68.49	62.63	73.39	39.60	51.39	37.88	57.56
	SPSR	7.78	44.60	10.79	66.85	67.72	62.23	71.87	39.80	48.78	37.46	56.39
LLaMA3.2-3B	Dense	7.81	11.79	11.70	69.77	73.61	73.91	76.61	43.00	71.68	45.90	64.93
	ShortGPT	539.07	4.10e7	100.44	61.64	49.01	45.05	64.20	32.40	49.12	34.13	47.94
	CL	95.09	135.74	88.18	62.35	48.41	68.72	64.80	32.40	49.20	35.15	51.58
	ReplaceMe	<u>22.77</u>	36.41	32.64	65.98	55.37	71.83	66.76	30.60	52.27	34.56	53.91
	Linear-patch	20.20	<u>35.72</u>	32.31	65.35	60.02	70.18	66.97	32.20	50.88	34.81	54.35
	LLM-Streamline	24.36	27.33	25.67	67.32	58.07	70.37	67.57	34.80	59.18	36.60	56.27
	SPSR	23.37	42.02	<u>27.83</u>	67.48	57.81	71.93	67.03	31.20	55.51	34.39	<u>55.05</u>

Table 19: Layer indices selected for removal by CL and RA are shown, with indexing starting from 0. L_p denotes the number of pruned layers and L_t denotes the total number of layers of the model.

Model	L_p/L_t	Metric	Removing Layer Indices
LLaMA2-7B	8/32	CL	21, 22, 23, 24, 25, 26, 27, 28
	8/32	RA	20, 21, 22, 23, 24, 25, 26, 27
	6/32	CL	23, 24, 25, 26, 27, 28
	6/32	RA	22, 23, 24, 25, 26, 27
LLaMA3-8B	8/32	CL	22, 23, 24, 25, 26, 27, 28, 29
	8/32	RA	21, 22, 23, 24, 25, 26, 27, 28
	6/32	CL	22, 23, 24, 25, 26, 27
	6/32	RA	23, 24, 25, 26, 27, 28
Qwen2.5-7B	8/28	CL	11, 12, 13, 14, 15, 16, 17, 18
	8/28	RA	2, 5, 6, 7, 8, 9, 24, 25
	6/28	CL	12, 13, 14, 15, 16, 17
	6/28	RA	2, 5, 8, 9, 24, 25
Qwen3-8B	8/36	CL	25, 26, 27, 28, 29, 30, 31, 32
	8/36	RA	26, 27, 28, 29, 30, 31, 32, 33
	6/36	CL	27, 28, 29, 30, 31, 32
	6/36	RA	27, 29, 30, 31, 32, 33

mal scale+bias parameterization enables effective recovery with only 128 samples.

L Various Validation Datasets and Zero-Shot Tasks Performance

We further evaluate our method on WikiText (Merity et al., 2017), PTB (Marcus et al., 1994), and C4 (Raffel et al., 2020). We conduct extensive experiments on LLaMA2, LLaMA3, Qwen2.5, and Qwen3. As shown in Table 24, SPSR achieves comparable or even lower perplexity than other methods while requiring only minute-level recovery

time. These results demonstrate the robustness of our method across diverse validation datasets.

For zero-shot evaluation, we measure accuracy on seven commonsense benchmarks from EleutherAI LM Harness (Gao et al., 2023), including BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), ARC-Easy (Boratto et al., 2018), ARC-Challenge (Boratto et al., 2018) and OpenbookQA (Mihaylov et al., 2018). Per task accuracies are shown in Table 24. Overall, SPSR achieves comparable or even higher average accuracy than other methods while requiring only minute-level recovery time.

M End-to-End Inference Latency

A natural concern for non-consecutive layer pruning is whether the resulting fragmented architecture introduces deployment overhead. We measure end-to-end latency and throughput on both CPU (DeepSparse) and GPU (vLLM) engines. As shown in Table 22 and 23, incorporating SPSR’s scale-and-bias vectors introduces negligible overhead compared to the pruned model without replacement. On GPU with vLLM, SPSR achieves a 1.19× throughput improvement over the dense baseline while maintaining near-identical Time-To-First-Token (TTFT) and Time-Per-Output-Token (TPOT) as the purely pruned model, confirming that SPSR is deployment-friendly even with non-consecutive layer removal.

Table 20: Detailed training configurations and resource consumption for all compared methods on LLaMA3-8B (8 layers pruned).

Method	Samples (Dataset)	SeqLen	Optimizer	LR	Epochs	Batch Size	Weight Decay	Scheduler	Loss	Trainable Module	Runtime	Memory (GPU/CPU)
SPSR	128 (C4)	2048	AdamW	3e-2	10	1	1e-4	Linear warmup (~201 steps)	NTP loss	Scale/Bias	3 min	17.0G / 2.2G
ReplaceMe	8000 (SlimOrca)	1024	Adam	1e-4	10	1024	0	None	Cosine loss	Linear	38 min	15.6G / 55.9G
LLM-Streamline	3000 (SlimPajama)	2048	AdamW	2e-4→5e-5	20	128 (8×16)	1e-3	Cosine + warmup	MSE	Transformer layer	227 min	29.5G / 106.7G
Linear-Patch	2048 (WikiText)	2048	AdamW	1e-4→5e-6	1	1	1e-4	CosineAnnealingLR	KL divergence	Linear	42 min	17.4G / 6.9G

Table 21: Perplexity on different validation datasets and accuracy (%) on seven zero-shot tasks with various layer pruning methods on LLaMA2-7B at 25% sparsity. The best performance is set in bold, and the second best is set in underline.

Method	Perplexity↓			Accuracy(%)↑							
	WikiText	PTB	C4	WinoGrande	HellaSwag	BoolQ	PIQA	OBQA	ARC-e	ARC-c	Mean
Dense	5.47	24.09	7.53	67.17	72.96	70.15	78.29	40.80	53.62	29.71	58.96
LLM-Streamline	13.20	38.99	13.23	65.04	61.10	67.06	70.18	38.40	44.99	34.13	54.42
LLM-Streamline(C4)	24.70	55.18	20.74	63.54	56.13	65.50	67.57	36.20	41.79	33.79	52.08
SPSR	11.16	38.64	13.73	63.54	59.71	72.57	69.15	36.20	43.73	34.22	54.16

Table 22: End-to-end inference latency and throughput on CPU using the DeepSparse engine (Qwen3-8B, sequence length 1024).

Method	Latency (ms) ↓	Throughput (tok/s) ↑
Dense	703.93	1.42
−8 Layers (no replacement)	534.50	1.87
+ SPSR	542.85	1.84

Table 23: End-to-end inference performance on GPU using vLLM (LLaMA3-8B, RTX 3090, input len=32, output len=4).

Method	Throughput (tok/s) ↑	TTFT (ms) ↓	TPOT (ms) ↓
Dense	1353.82	58.04	22.47
−8 Layers (no replacement)	1637.38	47.18	18.35
+ SPSR	1608.64	47.32	18.89

N Large Language Models Usage in Our Paper

We gratefully acknowledge the use of Large Language Models linguistic editing to enhance the manuscript’s read ability. The AI-generated edits were critically evaluated and modified by the author, ensuring the fidelity of the content.

Table 24: Perplexity on different validation datasets and accuracy (%) on seven zero-shot tasks with various layer pruning methods. L_p denotes the number of pruned layers and L_t denotes the total number of layers of the model. The best performance is set in bold, and the second best is set in underline.

Model	L_p/L_t	Method	Perplexity↓			Accuracy(%)↑							
			WikiText	PTB	C4	WinoGrande	HellaSwag	BoolQ	PIQA	OBQA	ARC-e	ARC-c	Mean
LLaMA2-7B	0/32	Dense	5.47	24.09	7.53	67.17	72.96	70.15	78.29	40.80	53.62	29.71	58.96
	8/32	ShortGPT	25.41	79.80	29.94	61.01	57.75	62.66	66.97	37.20	41.16	33.11	51.41
	8/32	CL	18.19	57.36	21.99	59.67	58.17	67.61	66.65	37.60	40.87	32.08	51.81
	8/32	ReplaceMe	13.09	40.80	15.34	62.43	59.54	68.20	68.44	37.00	45.29	33.79	53.53
	8/32	Linear-Patch	9.92	38.12	<u>13.62</u>	64.01	64.49	63.82	69.70	38.00	46.09	34.98	54.44
	8/32	LLM-Streamline	13.20	38.99	13.23	65.04	61.10	67.06	70.18	38.40	44.99	34.13	<u>54.42</u>
	8/32	SPSR(CL)	11.41	<u>38.20</u>	13.63	63.30	61.10	65.78	69.80	38.00	45.83	33.45	53.90
	8/32	SPSR	<u>11.16</u>	38.64	13.73	63.54	59.71	72.57	69.15	36.20	43.73	34.22	54.16
	6/32	ShortGPT/CL	12.18	43.25	15.54	62.35	63.33	65.66	70.40	39.00	46.00	32.59	54.19
	6/32	ReplaceMe	9.34	34.02	11.70	65.11	64.61	72.42	72.63	37.00	49.58	34.64	56.57
	6/32	Linear-Patch	7.85	31.49	11.13	65.82	68.36	62.87	73.45	37.00	48.57	36.52	<u>56.09</u>
	6/32	LLM-Streamline	10.25	35.24	11.38	64.01	65.48	63.88	72.36	37.80	48.27	36.69	55.50
	6/32	SPSR(CL)	9.10	33.52	11.34	64.80	65.50	64.95	73.39	36.80	48.65	35.07	55.60
	6/32	SPSR	<u>8.90</u>	<u>32.92</u>	<u>11.25</u>	64.56	65.03	68.01	72.47	37.00	46.89	34.73	55.53
LLaMA3-8B	0/32	Dense	6.14	9.91	9.86	72.61	79.16	81.59	79.65	45.00	77.69	53.41	69.87
	8/32	ShortGPT/CL	3945.75	3954.53	4893.89	54.06	31.50	37.61	58.71	28.20	38.13	31.23	39.92
	8/32	ReplaceMe	18.67	26.63	28.30	68.35	60.33	69.17	69.15	32.20	58.50	36.86	56.37
	8/32	Linear-Patch	15.44	30.21	26.70	71.98	70.24	65.57	73.01	37.40	62.67	42.75	60.52
	8/32	LLM-Streamline	13.47	15.94	19.05	71.35	66.68	76.94	71.49	37.40	65.49	43.43	61.83
	8/32	SPSR(CL)	16.28	27.08	21.33	72.30	65.78	59.02	72.03	35.00	63.51	40.36	58.29
	8/32	SPSR	<u>14.16</u>	<u>18.88</u>	<u>20.23</u>	71.67	64.56	78.23	70.73	35.20	64.02	41.47	<u>60.84</u>
	6/32	ShortGPT	41.94	40.84	43.51	70.48	68.14	72.05	72.42	37.20	60.31	43.52	60.59
	6/32	CL	29.17	59.47	38.13	67.64	67.70	71.35	70.24	37.40	61.53	44.11	60.00
	6/32	ReplaceMe	12.74	17.09	20.30	71.74	66.70	77.95	72.03	33.40	64.52	39.76	60.87
	6/32	Linear-Patch	10.81	16.78	18.07	73.32	72.71	68.75	76.28	40.40	67.85	45.65	63.57
	6/32	LLM-Streamline	14.02	18.75	17.07	72.61	71.88	73.27	75.52	41.80	72.22	46.33	64.80
	6/32	SPSR(CL)	<u>11.13</u>	14.70	16.63	73.01	70.20	70.15	74.43	39.20	70.08	43.94	63.00
	6/32	SPSR	11.28	<u>15.59</u>	16.63	73.01	70.03	72.84	75.73	38.60	70.41	44.45	<u>63.58</u>
Qwen2.5-7B	0/28	Dense	6.85	11.36	12.16	73.01	78.95	85.11	78.78	47.20	77.40	50.94	70.20
	8/28	ShortGPT	13.92	22.69	21.51	55.25	56.41	49.39	71.71	35.60	59.34	33.36	51.58
	8/28	CL	12.63	20.65	20.07	52.64	56.35	51.44	73.12	36.60	59.85	34.22	52.03
	8/28	ReplaceMe	10.93	18.37	18.17	55.64	58.50	55.72	75.14	42.00	60.94	35.32	54.75
	8/28	Linear-Patch	9.38	17.20	<u>16.90</u>	55.09	62.19	49.72	75.03	41.00	63.80	38.14	55.00
	8/28	LLM-Streamline	<u>9.84</u>	<u>17.35</u>	16.26	56.43	60.01	62.35	74.32	39.20	58.84	34.90	55.15
	8/28	SPSR(CL)	10.38	17.36	17.26	54.85	61.01	61.41	75.24	41.00	61.57	38.48	<u>56.22</u>
	8/28	SPSR	15.90	25.02	23.18	62.59	57.55	74.34	67.74	34.00	61.87	36.26	56.34
	6/28	ShortGPT	12.12	20.35	18.84	56.91	63.07	63.52	73.39	39.40	60.69	38.99	56.57
	6/28	CL	10.25	17.26	16.82	54.22	62.88	58.01	75.63	39.40	63.93	37.54	55.94
	6/28	ReplaceMe	9.48	15.99	16.06	57.77	63.66	64.62	76.33	40.80	67.34	40.70	58.75
	6/28	Linear-Patch	8.43	<u>15.18</u>	<u>15.15</u>	61.40	68.73	49.76	77.31	42.20	66.96	41.30	58.24
	6/28	LLM-Streamline	<u>8.58</u>	14.47	14.49	63.38	66.54	64.10	76.77	42.20	64.90	40.02	<u>59.70</u>
	6/28	SPSR(CL)	9.18	15.32	15.49	57.62	66.40	59.27	76.39	40.40	66.71	41.13	58.27
	6/28	SPSR	13.80	21.24	20.64	64.40	62.72	77.13	70.24	36.80	67.55	43.34	60.31
Qwen3-8B	0/36	Dense	9.72	15.44	15.82	67.64	74.96	86.64	76.88	41.40	80.93	56.48	69.28
	8/36	ShortGPT	594.75	1037.98	229.10	55.96	52.09	64.98	62.08	30.20	44.15	32.17	48.80
	8/36	CL	112.06	142.66	97.12	56.04	46.57	72.75	62.73	29.40	43.35	32.42	49.04
	8/36	ReplaceMe	45.32	62.81	52.21	60.93	48.81	77.89	62.51	26.80	45.16	32.42	50.65
	8/36	Linear-Patch	22.49	47.83	<u>37.56</u>	64.25	59.91	68.47	67.79	31.20	53.54	38.48	<u>54.80</u>
	8/36	LLM-Streamline	65.30	<u>43.01</u>	38.24	62.12	49.93	63.49	64.53	31.80	49.58	32.00	50.49
	8/36	SPSR(CL)	31.80	45.80	39.48	61.72	52.99	83.58	65.61	29.80	50.34	35.58	54.23
	8/36	SPSR	<u>30.39</u>	39.72	34.84	62.83	55.91	77.61	66.81	32.60	54.76	37.46	55.42
	6/36	ShortGPT	78.74	97.78	51.28	62.12	56.85	68.41	65.83	33.80	57.49	39.51	54.86
	6/36	CL	37.59	59.52	44.16	62.59	55.48	77.65	66.49	30.00	55.18	38.23	55.09
	6/36	ReplaceMe	28.70	34.04	34.39	63.30	55.87	85.32	66.49	31.40	55.72	37.88	56.57
	6/36	Linear-Patch	14.87	24.93	25.31	67.17	63.71	84.22	70.67	33.60	63.80	44.03	<u>61.03</u>
	6/36	LLM-Streamline	30.10	33.89	29.57	67.17	57.97	85.84	68.28	33.60	63.13	41.47	59.64
	6/36	SPSR(CL)	<u>18.17</u>	<u>24.96</u>	<u>25.39</u>	67.25	61.42	81.38	71.22	34.20	64.52	43.26	60.46
	6/36	SPSR	19.79	27.86	26.40	68.43	61.60	86.02	71.60	35.80	64.35	43.09	61.56