# Two-Stage Regularization-Based Structured Pruning for LLMs

Anonymous ACL submission

## Abstract

The deployment of large language models (LLMs) is largely hindered by their large number of parameters. Structural pruning has emerged as a promising solution. Prior structured pruning methods directly remove unimportant parameters based on certain metrics, which often causes knowledge loss and necessitates extensive retraining. To overcome this, we introduce a novel pruning method **TRSP**: **T**wo-Stage **R**egularization-Based **S**tructured **P**runing for LLMs. Specifically, we multiply the output of each transformer layer by an initial learnable weight and iteratively learn these weights by adding their $\ell_1$-norm as a regularization term to the loss function, serving as the first-stage regularization. Subsequently, we apply additional regularization to the difference between the output and input of layers with smaller weights, encouraging the shift of knowledge to the preserved layers. This serves as the second-stage regularization. TRSP retains more knowledge and better preserves model performance than direct parameter elimination. Through extensive experimentation we show that TRSP outperforms strong layer-wise structured pruning methods without requiring retraining. As a layer-wise pruning method, it delivers notable end-to-end acceleration, making it a promising solution for efficient LLM deployment.

## 1 Introduction

Large language models (LLMs) have made remarkable progress in natural language processing (Yang et al., 2024a; Wu et al., 2024; Guo et al., 2025). However, their large scale makes real-world deployment challenging. There is an urgent need for techniques that can enhance the compactness and computational efficiency of LLMs while preserving their language modeling capabilities.

Structured pruning is a method used to simplify neural networks by removing unnecessary or redundant parameters (Xia et al., 2024; An et al., 2024; Feng et al., 2025). Structured pruning is categorized into channel-wise pruning (Ma et al., 2023; Ashkboos et al., 2024) and layer-wise pruning (Song et al., 2024). Channel-wise pruning operates at the row or column level of parameter matrices. Layer-wise pruning operates at the level of entire transformer layers thereby offering a more simple approach compared to channel-wise pruning (Chen et al., 2024; Men et al., 2024).

However, existing layer-wise pruning methods have a certain limitation. They consistently first compute the importance of each transformer layer using a designed criteria, prune unimportant layers, and then fine-tune the pruned model to compensate for performance degradation caused by pruning. However, even unimportant layers can hold valuable knowledge (Dettmers et al., 2022; Yin et al., 2024; An et al., 2025). This sequential process of selecting and then directly pruning layers does not handle the important knowledge contained in the layers that are to be pruned, resulting in its direct loss. The performance drop requires substantial retraining for recovery, leading to considerable computational overhead (Ma et al., 2023).

To address this, we present TRSP that first apply two-stage regularization and then prune. The first regularization process iteratively learns layer weights. The second regularization process dynamically transfers valuable knowledge from the layers to be pruned to the remaining layers in advance, greatly reducing the knowledge loss caused by pruning. The comparison of TRSP with existing layer-wise pruning approaches is shown in Figure 1. *First*, we sample a small portion of data from standard benchmark datasets randomly. Given the limited scale of the selected data, the computational overhead incurred during the two-stage regularization is significantly reduced. *Second*, we multiply the output of each transformer layer by an initial learnable weight and iteratively learn these weights by incorporating their $\ell_1$-norm as a regularization

term in the loss function. This serves as the first stage regularization. *Third*, we apply regularization ($\ell_1$-norm or $\ell_2$-norm) to the difference between the outputs and inputs of the layers with smaller weights, forcing important knowledge to be transferred to the remaining layers which significantly reduces the performance decline caused by parameter removal. Thus the model can maintain good language modeling capability. This serves as the second stage regularization. *Finally*, we prune the layers with smaller weights. Comprehensive experiments demonstrate that TRSP substantially outperforms strong layer-wise pruning methods in generation tasks and zero-shot tasks across different pruning ratios, while also significantly improving end-to-end acceleration. The main contributions of TRSP are summarized as follows:

- **Retention of Knowledge:** TRSP reduces knowledge loss by progressively applying two-stage regularization and performing pruning. This approach helps preserve model performance without requiring for retraining.
- **Effectiveness:** TRSP outperforms strong layer-wise pruning methods in generation and zero-shot tasks. The pruned model demonstrates a considerable acceleration.
- **Minimal Cost:** The data required for two-stage regularization is minimal and TRSP is retraining free after pruning.

## 2 Related Works

### 2.1 Model Pruning

Model Pruning aims to improve model efficiency by sparsification or parameter removal (LeCun et al., 1989; Hassibi et al., 1993; Han et al., 2015; Liu et al., 2017). Several studies employ unstructured (Kurtic et al., 2022; Zhang et al., 2024; Xu et al., 2024) and structured pruning (Xia et al., 2024; Yang et al., 2024b; Gao et al., 2024b).

Unstructured pruning zeros individual neurons according to their importance such as SparseGPT (Frantar and Alistarh, 2023), SpQR (Dettmers et al., 2024), Pruner-Zero (Dong et al., 2024), and Wanda (Sun et al., 2024) . Their main advantage is flexibility. However, they need dedicated hardware to accelerate (Xia et al., 2023), and are not able to retrain on downstream tasks.

Structured pruning methods can be categorized by granularity into channel-wise pruning (Ashkboos et al., 2024) and layer-wise pruning (Kim et al., 2024a). Channel-wise pruning methods create a metric to assess the significance of channels in the parameter matrices of LLMs, and then remove the less significant ones. Layer-wise pruning methods treat entire layers as the basic units for pruning. For instance, SLEB (Song et al., 2024) iteratively remove entire transformer layers by evaluating their impact on the model's final loss, ShortGPT (Men et al., 2024) thinks high similarity between layers means redundancy, LaCo (Yang et al., 2024b) uses layer collapse to prune, Shortened LLaMA (Kim et al., 2024b) prune layers in one-shot based on their importance. In this paper, we focus on layer-wise pruning. Prior layer-wise pruning approaches suffer from the drawback that the less important layers may still carry critical knowledge, and pruning them often causes knowledge loss. Discovering a way to reshape knowledge distribution prior to pruning could potentially alleviate knowledge loss.

### 2.2 Regularization

In machine learning, regularization plays a vital role in controlling overfitting (Santos and Papa, 2022) and identifying informative features (Tibshirani, 1996), and has been extensively studied (Hoerl and Kennard, 1970; Poggio et al., 1987; Balestriero et al., 2022). The $\ell_1$-norm tends to enforce compact representations by eliminating certain parameters, whereas the $\ell_2$-norm favors stability and continuity (Boyd and Vandenberghe, 2004). Both can both alter the underlying structure and representation of the data (Han et al., 2015; Tao et al., 2023). Inspired by this insight, regularization can be leveraged to migrate critical knowledge from pruned layers to preserved layers, thereby enhancing model performance.

## 3 Methodology

The complete TRSP procedure is outlined in Algorithm 1 and Figure 1. The TRSP framework involves four key stages: **(1) Prepare data:** Select a small amount of data for the following two-stage regularization. **(2) Learn layer weights:** Iteratively learn the weights of each layer by incorporating their $\ell_1$-norm as a regularization term in the loss function. **(3) The second stage regularization:** Apply regularization to the difference between the output and input of layers with smaller weights, facilitating knowledge transfer. **(4) Pruning:** Removes the layers that were regularized in

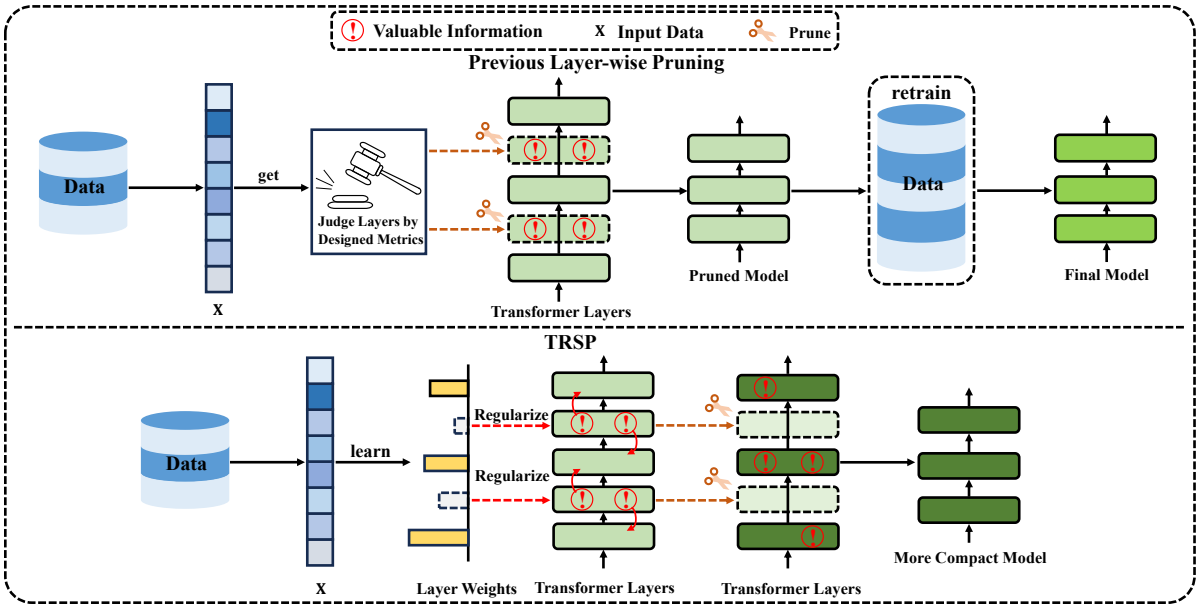


Figure 1: A comparison between existing layer-wise structured pruning methods and TRSP . Deeper blue layer represents greater performance impact, the taller cylinder represents larger data volume.

the previous step.

**Algorithm 1** TRSP algorithm.

**Input:** selected data: $\mathbf{X}$, number of layers: $l$, initial model $\mathbf{W}$, number of layers to prune: $n$, layer weights: $S$, set of pruned layers: $P$, norm type: flag.
Initialize each layer weight in $S$ to 1.
$P \leftarrow \emptyset$, $MinS \leftarrow 1e9$
**for** $i = 0$ **to** $n - 1$ **do**
  $S \leftarrow learnWeights(\mathbf{W}, S, \mathbf{X})$
  **for** $j = 0$ **to** $l - i - 1$ **do**
    **if** $S[j] < MinS$ **then**
      $MinS \leftarrow S[j]$
      $MinS\_id \leftarrow j$
    **end if**
  **end for**
  $\mathbf{W} \leftarrow mask(\mathbf{W}, MinS\_id)$
  $P \leftarrow P \bigcup \{MinS\_id\}$
**end for**
$\mathcal{L}_{\text{sum}} \leftarrow \mathcal{L}(\mathbf{W}, \mathbf{X})$
**for** $i = 0$ **to** sizeof$(P)$ **do**
  $\mathcal{L}_{\text{sum}} \leftarrow \mathcal{L}_{\text{sum}} + regularized(\mathbf{W}[P[i]])$
**end for**
update $\mathbf{W}$ using backpropagation algorithm
$\mathbf{W} \leftarrow Prune(\mathbf{W}, P)$

### 3.1 Prepare Data

The data is drawn from standard benchmarks, including Alpaca (Taori et al., 2023), WikiText-2 (Merity et al., 2016), PTB (Marcus et al., 1993), and C4 (Raffel et al., 2020). For example, 128 instances are randomly drawn from the WikiText-2 training set to for layer weight learning via regularization and the second stage regularization.

### 3.2 Learn Layer Weights

This is the first stage regularization. To better understand our paper, we first define some notations. $\mathbf{W}$ represents the initial model. Let $p$ be the pruning ratio, indicating that $p\%$ of model layers will be pruned. The number of layers in the initial model is $l$. The model hidden size is $d$. Let $\mathbf{X} \in \mathbb{R}^{b \times n \times d}$ represents the data embedding, where $b$ is the batch size and $n$ is the number of tokens. The input to the $i$th layer is denoted as $\mathbf{X}^{\mathbf{i}}_{\text{in}} \in \mathbb{R}^{b \times n \times d}$, and the output from the $i$th layer is denoted as $\mathbf{X}^{\mathbf{i}}_{\text{out}} \in \mathbb{R}^{b \times n \times d}$. A learnable weight $S[i]$ is assigned to the $i$th transformer layer, and the set of all layer weights is denoted as $S$.

According to Algorithm 1, we initialize the weight of each layer to 1. As shown in Figure 2, the output of each layer is scaled by its associated weight before being passed as input to the next layer. To learn the weight of each layer, we employ the input data embedding $\mathbf{X}$. When pruning $n$ layers, $\mathbf{X}$ is repeatedly used as input in each iteration. The objective function in Equation 1 comprises two parts: the language modeling loss $\mathcal{L}(\mathbf{W}, \mathbf{X})$, and the sum of the $\ell_1$-norm of all layer weights, $\lambda_1$ balances the two components, $l_1$ is the number of

layers in the current model that are not masked.

$$\mathcal{L}_{\text{learn}} = \mathcal{L}(\mathbf{W}, \mathbf{X}) + \lambda_1 \sum_{i=0}^{l_1-1} \|S[i]\|_1 \quad (1)$$

Forward and backward propagation is then performed to learn the set of layer weights $S$. Subsequently, the layer with the smallest weight is identified and masked out in the next iteration, and its index is added to the pruning set $P$. This process is iteratively performed $n$ times if there are $n$ layers need to be pruned which follows a greedy strategy. We also explored a one-shot pruning approach, in which a single forward and backward propagation is used to identify the $n$ layers with the lowest weights. However, as shown in Section 4.8, this often results in the removal of consecutive transformer layers, which leads to a substantial degradation in model performance.

The process of minimizing the function in Equation 1 is treated as an optimization task. Since $\ell_1$-norm is not differentiable, backpropagation (BP) can't be used directly, we need to transform the problem using Proposition 3.1.

**Proposition 3.1** (If the objective function contains an $\ell_1$ regularization term, it can still be optimized using BP. Proof in Appendix A). *The following unconstrained optimization problem is equivalent to the constrained optimization problem, where* $\|\cdot\|_1$ *denotes the* $\ell_1$*-norm.*

$$\begin{aligned} \min \quad ||x||_1 \quad \Longleftrightarrow \quad \min_{x,y} \quad & \mathbf{1}^T y \\ \textit{s.t.} \quad & -y \leq x \leq y, \\ & y \geq 0. \end{aligned} \quad (2)$$

The objective function in Equation 1 can be reformulated as an equivalent constrained problem in Equation 3. After transformation, the objective function is differentiable, and can be solved by BP.

$$\begin{aligned} \min_{W,y} \quad & \mathcal{L}(\mathbf{W}, \mathbf{X}) + \lambda_1 \mathbf{1}^T y \\ \text{s.t.} \quad & -y \leq S \leq y, \\ & y \geq 0. \end{aligned} \quad (3)$$

### 3.3 The Second Stage Regularization

After learning the layer weighs, we further lighten the impact of the pruned layers on the final model output by applying regularization to them. The approach is simple and straightforward: we use the data embedding $\mathbf{X}$ as input and apply a one-shot regularization on the difference between the output and input of each layer in the pruning set $P$. For example, in Figure 2, the $i$th layer in $P$ and should be regularized, we just add the $\ell_1$-norm or $\ell_2$-norm of $\mathbf{X^i_{out}} - \mathbf{X^i_{in}}$ to the loss function. This encourages the knowledge to be redistributed from the pruned layers to the remaining ones, thereby reducing the amount of knowledge retained in the pruned layers and significantly minimizing the performance degradation caused by subsequent pruning.

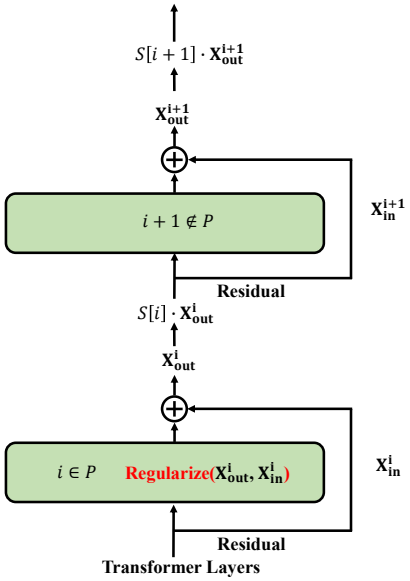


Figure 2: Details of layer weight learning and regularization. $P$ is the set of prunable layers, identified iteratively using the method in Section 3.2.

The objective function in Equation 4 comprises two parts: the language modeling loss $\mathcal{L}(\mathbf{W}, \mathbf{X})$ and the regularization loss, $\lambda_2$ balances the two components. When the $\ell_1$-norm is used, the equivalent constrained optimization problem transformed using Proposition 3.1 is shown in Equation 5. This formulation can be directly solved using BP.

$$\mathcal{L}_{\text{sum}} = \mathcal{L}(\mathbf{W}, \mathbf{X}) + \lambda_2 \sum_{i=0}^{|P|-1} \|\mathbf{X^i_{out}} - \mathbf{X^i_{in}}\| \quad (4)$$

$$\begin{aligned} \min_{W,Y_i} \quad & \mathcal{L}(\mathbf{W}, \mathbf{X}) + \lambda_2 \sum_{i=0}^{|P|-1} \mathbf{1}^T \mathbf{Y_i} \mathbf{1} \\ \text{s.t.} \quad & -\mathbf{Y_i} \leq \mathbf{X^i_{out}} - \mathbf{X^i_{in}} \leq \mathbf{Y_i}, \\ & \mathbf{Y_i} \geq 0. \end{aligned} \quad (5)$$

### 3.4 Pruning

After applying the regularization, we directly remove the transformer layers in the set $P$.

## 4 Experiments

This section introduces experimental setup (4.1) and analyzes the effectiveness of TRSP from the following aspects: performance comparison (4.2), acceleration 4.3, robustness under different pruning ratios (4.4), dependency on different datasets (4.5), low overhead (4.6), and ablation study (4.7), choice of learning layer weights (4.8), impact of regularization (4.9).

### 4.1 Experimental Setup

**Datasets:** We evaluated on generation and zero-shot tasks. For generation task, following prior work(Ashkboos et al., 2024), we evaluate the model's perplexity on WikiText-2 (Merity et al., 2016) test set. For zero-shot task, we evaluate on 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 Section 4.5 we randomly selected data from Alpaca (Taori et al., 2023), WikiText-2 (Merity et al., 2016), PTB (Marcus et al., 1993), and C4 (Raffel et al., 2020).

**Implementation:** All methods are developed in PyTorch (Paszke et al., 2019), leveraging the Hugging Face Transformers library (Wolf, 2019). Experimental evaluations are carried out on 80GB NVIDIA A100 GPUs. Pruned models are evaluated with llm-eval-harness (Gao et al., 2024a). More details are in Appendix B.

**Evaluation Metrics:** The ability on generation task is evaluated by *perplexity*, a well-established and robust metric (Yao et al., 2022). Zero-shot tasks ability is evaluated using *accuracy* (Dong et al., 2024). Acceleration is measured by *throughput* and *latency* (Song et al., 2024).

**Models:** The models include the Phi-2 (Javaheripi et al., 2023), OPT models (OPT-2.7B, OPT-13B) (Zhang et al., 2022), and LLaMA models (LLaMA2-7B, LLaMA2-13B, LLaMA3-8B) (Touvron et al., 2023; Grattafiori et al., 2024).

**Baselines:** We compare TRSP with strong layer-wise structured pruning methods: SLEB (Song et al., 2024), ShortGPT (Men et al., 2024), LaCo (Yang et al., 2024b), Shortened LLaMA (PPL version) (Kim et al., 2024b).

### 4.2 Performance Comparison

To ensure fairness, 128 sequences of length 2048 were randomly drawn from the WikiText-2 training set for TRSP's two-stage regularization and the calibration process of the baselines. Following (Ashkboos et al., 2024), we selected another 1,000 samples from WikiText-2 training dataset to retrain leveraging LoRA (Hu et al., 2022) on the baselines after pruning. The aforementioned 128 samples and the 1,000 samples have no overlap. Because TRSP is retraining-free, we did not retrain TRSP. The pruning ratio was 25% which means 25% of the transformer layers in a model will be removed. As shown in Table 1, TRSP achieves the lowest perplexity and the highest average accuracy across all models, demonstrating its superior performance on both *generation* and *zero-shot* tasks. Notably, on OPT-13B, TRSP only drop 1% in average accuracy compared to the dense model. On LLaMA2-7B, its perplexity is 20% lower than the second-best method, ShortGPT. This further demonstrates the effectiveness of TRSP. The minimal performance difference between $\ell_2$-norm and $\ell_1$-norm suggests that TRSP is not sensitive to the choice of regularization norm. In the following sections, we refer to using the $\ell_2$-norm in the stage two regularization.

### 4.3 Acceleration

LLMs language processing involves two key phases with different bottlenecks: compute-bound prompt processing and memory-bound token generation. We measured the speedup for each stage individually. Table 2 presents the latency and throughput results for OPT-13B and LLaMA2-13B running on a single 80GB NVIDIA A100 GPU. Following prior method (Song et al., 2024), the token generation was tested by producing 128-token sentences with a batch size of 64, and prompt processing latency was assessed with a 2048-token input.

Pruning OPT-13B by 50% with TRSP yields a 75% increase in throughput and a 46% reduction in latency compared to the dense model. For LLaMA2-13B, it delivers a 71% improvement in throughput and a 45% decrease in latency. These results underscore the end-to-end acceleration achieved by TRSP.

### 4.4 Robustness to Different Pruning Ratios

Using the same settings as Section 4.2, we vary the pruning ratio from 20% to 60%. As shown in Figure 3, TRSP consistently achieves lower perplexity than other methods. At 20% pruning, it matches the dense model, and even at 60%, where SLEB fails, it maintains low perplexity. This demonstrates TRSP's robustness and effectiveness in structured pruning for model acceleration. Additional results are provided in Appendix D.

Table 1: Performance comparison of TRSP and baselines. 'PR' is the pruning ratio. 'PPL' is the perplexity on WikiText-2. The accuracy is evaluated on five zero-shot benchmarks. TRSP -$\ell_2$ means using the $\ell_2$-norm in the second stage regularization, TRSP -$\ell_1$ is using the $\ell_1$-norm. The best result is in **bold**, the second-best is underlined.

| 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 |
| | SLEB | 25% | 7.65 | 68.85 | 63.63 | 50.96 | 49.38 | 30.79 | 52.72 |
| | ShortGPT | 25% | 7.15 | 69.67 | 65.19 | 51.26 | **52.46** | 33.89 | 54.49 |
| | LaCo | 25% | 7.38 | 68.53 | 63.76 | 50.39 | 51.28 | 33.45 | 53.48 |
| | Shortened LLaMA | 25% | 7.74 | 66.88 | 62.19 | 51.45 | 51.47 | 32.66 | 52.93 |
| | TRSP -$\ell_2$ | 25% | **6.53** | 71.35 | **67.62** | 55.84 | 52.26 | 35.75 | **56.56** |
| | TRSP -$\ell_1$ | 25% | 6.58 | **71.42** | 67.13 | **56.05** | 52.18 | **35.79** | 56.51 |
| **OPT-2.7B** | Dense | 0% | 12.46 | 74.81 | 61.01 | 60.58 | 54.42 | 31.14 | 56.39 |
| | SLEB | 25% | 15.71 | 65.19 | 56.26 | 44.54 | 46.28 | 25.14 | 47.48 |
| | ShortGPT | 25% | 14.96 | 67.37 | 57.65 | **46.82** | 49.43 | 26.54 | 49.56 |
| | LaCo | 25% | 15.38 | 66.54 | 59.45 | 43.68 | 48.74 | 24.96 | 48.67 |
| | Shortened LLaMA | 25% | 15.89 | 63.14 | 57.36 | 43.57 | 47.62 | 25.88 | 47.51 |
| | TRSP -$\ell_2$ | 25% | 13.18 | 70.54 | **60.27** | 46.35 | 51.59 | 27.36 | 51.22 |
| | TRSP -$\ell_1$ | 25% | **13.12** | **70.65** | 60.13 | 46.24 | **51.76** | **27.55** | **51.27** |
| **LLaMA2-7B** | Dense | 0% | 5.47 | 79.11 | 69.06 | 75.99 | 74.58 | 46.25 | 69.00 |
| | SLEB | 25% | 9.63 | 65.22 | 63.38 | 55.51 | 56.39 | 33.46 | 54.79 |
| | ShortGPT | 25% | 8.89 | 66.75 | 66.26 | 57.14 | 58.93 | 36.42 | 57.10 |
| | LaCo | 25% | 9.14 | 69.45 | 65.31 | 52.67 | 55.73 | 34.89 | 55.61 |
| | Shortened LLaMA | 25% | 9.47 | 65.58 | 64.72 | 58.36 | 54.19 | 32.96 | 55.16 |
| | TRSP -$\ell_2$ | 25% | **7.08** | **72.48** | 67.52 | **60.45** | **62.69** | 39.73 | **60.57** |
| | TRSP -$\ell_1$ | 25% | 7.17 | 72.08 | **67.93** | 60.42 | 62.38 | **39.89** | 60.54 |
| **LLaMA3-8B** | Dense | 0% | 5.76 | 85.56 | 77.94 | 79.27 | 78.84 | 56.49 | 75.62 |
| | SLEB | 25% | 10.38 | 72.74 | 64.12 | 67.74 | 65.84 | 45.16 | 63.12 |
| | ShortGPT | 25% | 9.26 | 75.38 | 69.25 | 70.12 | 68.54 | 47.57 | 66.17 |
| | LaCo | 25% | 10.14 | 74.69 | 67.52 | 66.36 | 69.43 | 46.31 | 64.86 |
| | Shortened LLaMA | 25% | 9.84 | 73.62 | 68.73 | 68.55 | 66.16 | 43.39 | 64.09 |
| | TRSP -$\ell_2$ | 25% | 7.84 | 77.25 | **71.63** | 72.26 | **71.49** | 49.58 | **68.44** |
| | TRSP -$\ell_1$ | 25% | **7.68** | **77.36** | 71.33 | **72.82** | 70.75 | **49.66** | 68.38 |
| **OPT-13B** | Dense | 0% | 10.12 | 76.82 | 64.80 | 69.81 | 61.87 | 35.67 | 61.79 |
| | SLEB | 25% | 11.96 | 72.83 | 64.06 | 63.32 | 59.98 | 34.65 | 58.97 |
| | ShortGPT | 25% | 11.38 | 73.59 | **64.52** | 65.68 | 60.41 | 34.99 | 59.84 |
| | LaCo | 25% | 11.79 | 73.96 | 63.24 | 62.17 | 61.46 | 33.28 | 58.82 |
| | Shortened LLaMA | 25% | 11.62 | 71.26 | 63.57 | 66.29 | 58.47 | 33.83 | 58.68 |
| | TRSP -$\ell_2$ | 25% | 10.45 | 74.15 | 64.47 | **68.82** | **61.55** | **35.23** | **60.84** |
| | TRSP -$\ell_1$ | 25% | **10.32** | **74.58** | 64.37 | 68.45 | 61.29 | 34.87 | 60.71 |
| **LLaMA2-13B** | Dense | 0% | 4.88 | 80.47 | 72.22 | 79.39 | 77.48 | 49.23 | 71.76 |
| | SLEB | 25% | 7.08 | 68.31 | 66.86 | 57.12 | 62.19 | 38.45 | 58.59 |
| | ShortGPT | 25% | 6.79 | 73.26 | 68.37 | 62.25 | 67.54 | 43.38 | 62.96 |
| | LaCo | 25% | 7.14 | 72.35 | 65.78 | 59.43 | 65.36 | 42.73 | 61.13 |
| | Shortened LLaMA | 25% | 6.92 | 69.45 | 66.38 | 59.86 | 65.25 | 39.12 | 60.01 |
| | TRSP -$\ell_2$ | 25% | **5.82** | 74.56 | **69.34** | 64.79 | 71.25 | **45.63** | 65.11 |
| | TRSP -$\ell_1$ | 25% | 5.89 | **75.06** | 69.18 | **64.96** | **71.43** | 45.26 | **65.18** |

### 4.5 Dependency on Datasets

Since TRSP relies on data-driven regularization, we investigate its dataset dependency. Keeping all other settings consistent with Section 4.2, we only change the source of the 128 samples: they are drawn respectively from Alpaca, WikiText-2, PTB, and C4. We evaluated perplexity of five methods on WikiText-2. As shown in Figure 4, TRSP consistently outperforms the other methods across datasets, demonstrating its robustness.

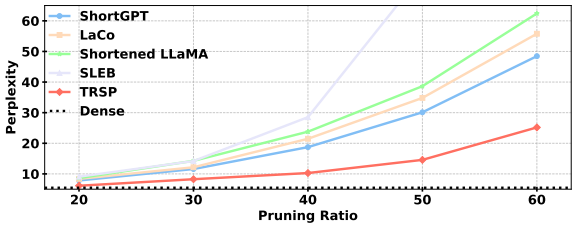


Figure 3: Perplexity of LLaMA2-7B pruned by five methods under different pruning ratios on WikiText-2.

### 4.6 Minimal Cost

We only vary the retraining data size from 1,000 to 8,000 and use the same settings as in Section 4.2. We evaluated the perplexity of baselines on LLaMA2-7B. TRSP is retraining-free. From Figure 5, we observe that TRSP outperforms the other methods with 4,000 retraining data, significantly reducing the cost. We speculate that TRSP iteratively learns layer weights and then applies reg-

Table 2: Throughput (tokens/s) and latency (ms) on OPT-13B and LLaMA2-13B. 'PPL' is the perplexity on Wikitext2. 'PR' is pruning ratio. 'TI' is throughput increase.

| Model | Method | PR | PPL(↓) | Avg_Acc(%) | Tokens/s(↑) | TI(↑) | Latency(↓) | Speedup(↑) |
|---|---|---|---|---|---|---|---|---|
| **OPT-13B** | Dense | 0% | 10.12 | 61.79 | 1029 | 1.00× | 386.5 | 1.00× |
| | TRSP | 25% | 10.45 | 60.84 | 1348 | 1.31× | 286.3 | 1.35× |
| | TRSP | 50% | 15.38 | 50.83 | 1801 | 1.75× | 208.9 | 1.85× |
| **LLaMA2-13B** | Dense | 0% | 4.88 | 71.76 | 1066 | 1.00× | 396.9 | 1.00× |
| | TRSP | 25% | 5.82 | 65.11 | 1386 | 1.30× | 298.4 | 1.33× |
| | TRSP | 50% | 11.28 | 57.57 | 1823 | 1.71× | 216.9 | 1.83× |

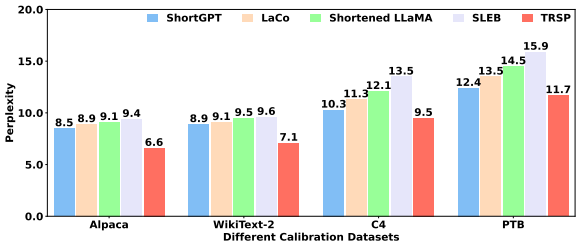


Figure 4: Comparison of perplexity on different calibration datasets at a pruning ratio of 25% on LLaMA2-7B.

Table 3: Ablation results on LLaMA2-7B and LLaMA2-13B. 'w/o W' denotes learning layer weights in one-shot, 'w/o R' means no regularization. The pruning ratio is 25%.

| Model | Setting | PPL(↓) | Δ | AVG_ACC | Δ(↓) |
|---|---|---|---|---|---|
| **LLaMA2-7B** | TRSP | 7.08 | 0.00 | 60.57 | 0.00 |
| | w/o W | 9.26 | +2.18 | 56.19 | -4.38 |
| | w/o R | 10.15 | +3.07 | 54.36 | -6.21 |
| **LLaMA2-13B** | TRSP | 5.82 | 0.00 | 65.11 | 0.00 |
| | w/o W | 8.35 | +2.53 | 59.36 | -5.75 |
| | w/o R | 9.47 | +3.65 | 56.25 | -8.86 |

ularization, allowing it to identify layers to prune more accurately, and then transfer the knowledge from those layers to the remaining layers of the model through regularization. This process reduces knowledge loss, thus preserving model performance and lowering the retraining cost.

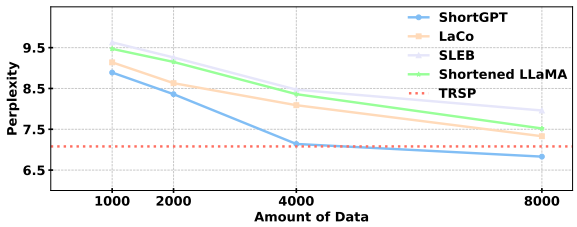


Figure 5: The perplexity on WikiText-2 using TRSP and baselines under different amounts of retraining data.

### 4.7 Ablation Study

**Effect of Learning Layer Weights Iteratively** TRSP learns layer weights in a greedy and iterative manner. As shown in Table 3 (Row 3 and Row 6), replacing iterative layer weight learning with a one-shot approach leads to increased model perplexity and decreased accuracy, highlighting the importance of learning layer weights iteratively, which will be discussed in detail in Section 4.8.

**Effect of Applying Regularization** As shown in Table 3 (Row 4 and Row 7), removing the regularization process results in increased model perplexity and decreased accuracy, demonstrating the effectiveness of applying regularization, which will be discussed in detail in Section 4.9.

With all other settings the same as Section 4.2, we evaluate the model's performance by varying $\lambda_1 \in [10^{-5}, 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}]$ and $\lambda_2 \in [10^{-5}, 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}]$. We perform a grid search over $\lambda_1$ and $\lambda_2$, with details provided in Appendix C. The optimal combination yielding the lowest perplexity on LLaMA2-7B is $\lambda_1 = 5 \times 10^{-3}$ and $\lambda_2 = 10^{-3}$.

### 4.8 Choice of Learning Layer Weights

In this section, we explore the effectiveness of (1) iteratively learning layer weights using a greedy strategy compared to (2) acquiring all layer weights at one-shot. We set the pruning ratio to 25% and conduct experiments on LLaMA2-7B (32 layers), using the same settings as in Section 4.2.

The perplexity and average accuracy of the pruned models are shown in Table 4. It can be observed that, compared to using a greedy strategy to iteratively learn layer weights, learning all layer weights in one-shot exhibits significant degradation in model performance. This behavior can be explained by the observation that the importance of a block changes as other blocks are removed. The results of selecting the eight least important layers using both methods are shown in Figure 6.

It can be seen that learning layer weights in one-shot tends to select consecutive layers. While these blocks may individually have limited impact on LLM inference performance, removing a continuous sequence of blocks can significantly degrade the overall inference results.

Table 4: The pruned LLaMA2-7B performance under different learning layer weights methods.

| Cases | PPL(↓) | Avg_Acc(%) |
|---|---|---|
| **(1)** | 7.08 | 60.57 |
| **(2)** | 9.26 | 56.19 |

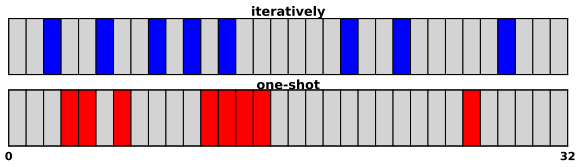


Figure 6: Results of selecting the eight lowest-weight layers using iterative and one-shot layer weight learning.

### 4.9 Impact of Stage Two Regularization

We keep other settings consistent with Section 4.2. After learning the layer weights iteratively, we consider two scenarios: (1) without regularization and (2) with regularization, then prune. The perplexity and average accuracy under (1) and (2) on LLaMA2-7B are in Table 5. The model exhibits lower perplexity and higher average accuracy with regularization. Since perplexity is the exponential form of cross-entropy loss, lower perplexity corresponds to a lower cross-entropy loss. A smaller loss function indicates better model performance indicating that the regularization process mitigates the impact of pruning on the overall model.

According to previous work (Liu et al., 2023), the output and input of each layer in LLMs exhibit high similarity. This inherent similarity enables us to apply regularization on the difference between the input and output of certain layers using only a small portion of data. We compute the cosine similarity according to Equation 9 on the LLaMA2-7B model between the input representations $\mathbf{X}_{\text{in}}^{\mathbf{i}} \in \mathbb{R}^{b\times n\times d}$ and the output representations $\mathbf{X}_{\text{out}}^{\mathbf{i}} \in \mathbb{R}^{b\times n\times d}$, where $b$ denotes the batch size, $n$ the number of tokens, and $d$ is the hidden size of the model. The computation is performed on layers [3, 6, 9, 11, 13, 20, 23, 29], which are selected using the iterative method described in Section 4.8, both before and after regularization.

As shown in Figure 7, the input-output similarity of these layers is already high before regularization and increases even further after regularization. The increased similarity in the layers with regularization indicates that the input undergoes less change after passing through these layers, suggesting that less knowledge is retained in them. In the Appendix E, we illustrate the changes in similarity between the input and output of the layers that were not regularized, before and after regularization. The results in Figure 8 of Appendix E show that the similarity in these layers decreases, meaning that, after regularization, the input undergoes greater changes when passing through these layers than before. This demonstrates that the regularization process weakens the influence of the regularized layers and enhances the influence of the unregularized parts, suggesting that regularization process may facilitate the transfer of knowledge from the regularized layers to the rest of the model.

Table 5: The performance differences of (1) and (2).

| Cases | PPL(↓) | Avg_Acc(%) |
|---|---|---|
| **(1)** | 7.08 | 60.57 |
| **(2)** | 10.15 | 54.36 |

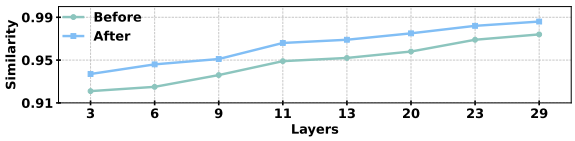


Figure 7: The input-output similarity of the regularized layers.

## 5 Conclusion

We propose a novel structured pruning method, TRSP. By performing two-stage regularization, TRSP retains more knowledge and better preserves model performance compared to direct parameter elimination. TRSP surpasses existing layer-wise pruning methods in generation and zero-shot tasks. For example, it reduces perplexity by 20% compared to ShotGPT on LLaMA2-7B, under 25% sparsity, the average accuracy decreases by just 1%, while delivering a 1.35× acceleration over the dense model. TRSP is retraining-free, significantly lowering computational overhead and dependence on retraining. The novel structured pruning method offers potential guidance for pruning strategies in LLMs.

## Limitations

TRSP has primarily been evaluated on autoregressive language models and its applicability to other architectures or tasks remains unexplored. In future work, we plan to explore the application of TRSP to other model architectures, such as convolutional neural networks (CNNs).

## References


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.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2025. Systematic outliers in large language models. In *The Thirteenth International Conference on Learning Representations*.

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*.

Randall Balestriero, Leon Bottou, and Yann LeCun. 2022. The effects of regularization and data augmentation are class dependent. *Advances in Neural Information Processing Systems*, 35:37878–37891.

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.

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. 2024. Streamlining redundant layers to compress large language models. *arXiv preprint arXiv:2403.19135*.

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.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2024. Spqr: A sparse-quantized representation for near-lossless llm weight compression. In *ICLR*.

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.

Mingkuan Feng, Jinyang Wu, Shuai Zhang, Pengpeng Shao, Ruihan Jin, Zhengqi Wen, Jianhua Tao, and Feihu Che. 2025. Dress: Data-driven regularized structured streamlining for large language models. *arXiv preprint arXiv:2501.17905*.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, 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*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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.

Arthur E Hoerl and Robert W Kennard. 1970. Ridge regression: applications to nonorthogonal problems. *Technometrics*, 12(1):69–82.

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.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. 2024a. Shortened llama: A simple depth pruning for large language models. *ICLR Workshop on Mathematical and Empirical Understanding of Foundation Models (ME-FoMo)*.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. 2024b. Shortened LLaMA: A simple depth pruning for large language models. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*.

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.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, and 1 others. 2023. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, 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.

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.

Tomaso Poggio, Vincent Torre, and Christof Koch. 1987. Computational vision and regularization theory. *Readings in computer vision*, pages 638–643.

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.

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.

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*.

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. *arXiv 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*.

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*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024a. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024b. Laco: Large language model pruning via layer collapse. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6401–6417.

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*.

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

## A Proof of Proposition 3.1

**Step 1: Expressing $\ell_1$-norm Using Elements.** The objective function in the unconstrained problem is the $\ell_1$-norm of the vector $x$, which is defined as:

$$||x||_1 = \sum_{i=1}^{n} |x_i| \tag{6}$$

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 $i$:

$$x_i \geq -y_i \quad \text{and} \quad x_i \leq y_i \tag{7}$$

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:

$$\mathbf{1}^T y \tag{8}$$

Thus, minimizing $\mathbf{1}^T y$ is equivalent to minimizing the sum of the absolute values of $x$, which is the $\ell_1$-norm of $x$.

This transformation allows the optimization problem to be solved without directly involving the absolute value function, resulting in an equivalent constrained optimization problem that can be addressed via backpropagation. Thus, the proof of the Proposition 3.1 is complete.

## B Detailed Implementation

In this part, we first introduce several hyperparameter settings, with the detailed results shown in Table 6. In our experiments, we employ FP16 precision for all evaluated models, including Phi-2, OPT-2.7B, LLaMA3-8B, OPT-13B, LLaMA2-7B, and LLaMA2-13B. For all retraining configurations, we set the LoRA rank $r$ to 32, the scaling

factor $\alpha$ 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 TRSP and other methods, we maintain consistency in the data used across all approaches. Specifically, the data used by TRSP for learning layer weights and performing the second stage regularization is identical to the calibration data used by the baseline methods. Furthermore, we ensure that the data employed during the retraining process is consistent across all baseline methods. Following previous works (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 TRSP for learning weights and regularization.

# C Optimal $\lambda_1$ and $\lambda_2$ for LLaMA2-7B

Keeping all other settings consistent with Section 4.2, we evaluate the model's perplexity on WikiText-2 by varying $\lambda_1 \in [10^{-5}, 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}]$ and $\lambda_2 \in [10^{-5}, 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}]$, resulting in a total of 49 combinations.

It can be observed from Table 7 that the optimal combination yielding the lowest perplexity on LLaMA2-7B is $\lambda_1 = 5 \times 10^{-3}$ and $\lambda_2 = 10^{-3}$.

When fixing $\lambda_2 = 10^{-3}$, we gradually decrease $\lambda_1$ from $5 \times 10^{-3}$ to $10^{-5}$, during which the model's perplexity increases steadily. This suggests that when the $\ell_1$-norm loss constitutes a relatively small portion of the total loss during the iterative learning of layer weights, it fails to effectively constrain the layer weights, resulting in larger deviations. Conversely, when $\lambda_1$ is gradually increased from $5 \times 10^{-3}$ to $10^{-1}$, the model's perplexity also increases, indicating that a dominant $\ell_1$-norm loss in the total objective function can hinder the optimization of the language modeling capability.

As shown in Table 7, when $\lambda_1 = 5 \times 10^{-3}$ is fixed, setting $\lambda_2 = 10^{-5}$ results in a regularization loss that is too weak to effectively redistribute important information. This leads to a substantial increase in perplexity after pruning. On the other hand, when $\lambda_2 = 10^{-1}$, the overly strong regularization impairs the model's language modeling capability, also resulting in a noticeable drop in performance after pruning. The best performance is observed at $\lambda_2 = 10^{-3}$, highlighting the critical need to balance the language modeling loss and regularization loss.

# D Performance of TRSP under Different Pruning Ratios and Datasets

## D.1 The Perplexity of TRSP under Different Pruning Ratios and Datasets

To systematically evaluate the performance of TRSP on zero-shot tasks across different pruning rates, we adopt the experimental setup outlined in Section 4.2, in which 128 samples are randomly sampled from the WikiText-2 training set to guide both the iterative learning of layer weights and the regularization process. Subsequently, we evaluate multiple large language models (LLMs) by measuring changes in perplexity across various generative task datasets, including WikiText-2, Alpaca, PTB, and C4, under pruning rates of 10%, 20%, 30%, 40%, 50%, and 60%. The detailed results, presented in Table 8, indicate that TRSP exhibits greater robustness as model scale increases, suggesting that the proposed method effectively mitigates performance degradation in larger architectures. This highlights the scalability of TRSP and its potential to maintain model efficiency under varying levels of sparsity.

## D.2 The Accuracy of TRSP under Different Pruning Ratios on Zero-shot Tasks

To systematically evaluate the performance of TRSP on zero-shot tasks across different pruning rates, we adopt the experimental setup outlined in Section 4.2, in which 128 samples are randomly sampled from the WikiText-2 training set to guide both the iterative learning of layer weights and the regularization process. We assess the accuracy of different model configurations at pruning rates of 10%, 20%, 30%, 40%, 50%, and 60% across a diverse set of benchmark datasets, including PIQA, WinoGrande, HellaSwag, ARC-e, and ARC-c. The results, summarized in Table 9, provide insights into the impact of sparsity on zero-shot generalization. Notably, the analysis reveals that TRSP maintains competitive performance even at higher pruning rates, demonstrating its effectiveness in preserving reasoning and commonsense understanding across different tasks.

Table 6: Implementation Details

| Precision | LoRA Rank | Scaling Factor | Max Sequence Length | Batch Size | Learning Rate | Early Stop Threshold | Min Delta |
|---|---|---|---|---|---|---|---|
| FP16 | 32 | 10 | 2048 | 64 | 2e-5 | 5 | 1e-4 |

Table 7: The optimal $\lambda_1$ and $\lambda_2$ for LLaMA2-7B.

| $\lambda_1$ / $\lambda_2$ | $10^{-5}$ | $10^{-4}$ | $10^{-3}$ | $5 \times 10^{-3}$ | $10^{-2}$ | $5 \times 10^{-2}$ | $10^{-1}$ |
|---|---|---|---|---|---|---|---|
| $10^{-5}$ | 12.28 | 11.91 | 8.94 | 10.48 | 13.79 | 14.73 | 15.87 |
| $10^{-4}$ | 11.45 | 10.36 | 8.26 | 9.82 | 11.87 | 12.56 | 13.62 |
| $10^{-3}$ | 10.82 | 9.14 | 7.92 | 8.53 | 9.75 | 10.82 | 12.98 |
| $5 \times 10^{-3}$ | 10.09 | 8.35 | **7.08** | 7.46 | 8.52 | 10.17 | 12.25 |
| $10^{-2}$ | 11.26 | 9.97 | 8.13 | 8.92 | 10.46 | 12.88 | 13.43 |
| $5 \times 10^{-2}$ | 12.89 | 11.25 | 10.38 | 11.76 | 12.85 | 14,75 | 15.37 |
| $10^{-1}$ | 14.55 | 13.58 | 12.25 | 14.41 | 15.72 | 16.91 | 17.93 |

Table 8: Perplexity comparison of TRSP 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 TRSP, we use the $\ell_2$-norm.

| Model | Pruning Ratio | WikiText-2(↓) | Alpaca(↓) | PTB(↓) | C4(↓) |
|---|---|---|---|---|---|
| **OPT-2.7B** | Dense | 12.46 | 11.64 | 17.97 | 14.32 |
| | 10% | 12.78 | 11.89 | 18.65 | 15.02 |
| | 20% | 12.96 | 12.15 | 19.38 | 16.36 |
| | 30% | 15.52 | 13.47 | 24.28 | 20.75 |
| | 40% | 19.67 | 15.82 | 32.25 | 25.41 |
| | 50% | 25.62 | 21.89 | 47.67 | 33.73 |
| | 60% | 35.62 | 32.19 | 59.87 | 47.38 |
| **OPT-6.7B** | Dense | 10.85 | 10.27 | 15.77 | 12.71 |
| | 10% | 11.03 | 10.95 | 16.78 | 13.46 |
| | 20% | 11.48 | 11.53 | 17.96 | 15.19 |
| | 30% | 12.87 | 12.75 | 20.86 | 17.52 |
| | 40% | 14.87 | 13.39 | 26.12 | 21.63 |
| | 50% | 21.43 | 16.51 | 35.36 | 28.74 |
| | 60% | 29.63 | 22.16 | 48.04 | 40.25 |
| **OPT-13B** | Dense | 10.12 | 9.46 | 14.52 | 12.06 |
| | 10% | 10.28 | 9.89 | 15.12 | 12.73 |
| | 20% | 10.39 | 10.37 | 16.51 | 13.45 |
| | 30% | 11.38 | 11.12 | 19.42 | 16.14 |
| | 40% | 12.67 | 12.33 | 24.62 | 20.79 |
| | 50% | 15.38 | 14.76 | 30.88 | 27.65 |
| | 60% | 28.23 | 20.98 | 39.38 | 36.14 |
| **LLaMA2-7B** | Dense | 5.47 | 5.25 | 7.92 | 7.26 |
| | 10% | 5.58 | 5.31 | 8.12 | 7.47 |
| | 20% | 6.13 | 5.87 | 8.78 | 7.92 |
| | 30% | 8.26 | 7.64 | 9.58 | 8.85 |
| | 40% | 10.28 | 9.79 | 12.87 | 11.42 |
| | 50% | 14.58 | 13.14 | 19.52 | 15.96 |
| | 60% | 25.18 | 21.46 | 30.14 | 28.32 |
| **LLaMA2-13B** | Dense | 4.88 | 4.63 | 7.16 | 6.73 |
| | 10% | 4.99 | 4.91 | 7.48 | 7.93 |
| | 20% | 5.34 | 5.26 | 8.25 | 8.75 |
| | 30% | 6.87 | 6.35 | 8.97 | 9.68 |
| | 40% | 8.95 | 7.92 | 10.08 | 11.26 |
| | 50% | 11.23 | 9.73 | 12.63 | 14.73 |
| | 60% | 15.74 | 12.52 | 17.82 | 19.45 |

Table 9: Accuracy comparison of TRSP 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 TRSP, we use the $\ell_2$-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% | 72.35 | 60.75 | 51.53 | 52.73 | 29.15 | 53.30 |
| | 20% | 71.47 | 60.46 | 48.72 | 51.95 | 28.04 | 52.13 |
| | 30% | 66.73 | 58.53 | 44.36 | 50.02 | 26.58 | 49.24 |
| | 40% | 63.57 | 55.32 | 41.63 | 47.79 | 25.24 | 46.71 |
| | 50% | 58.48 | 52.29 | 40.52 | 46.26 | 21.32 | 43.77 |
| | 60% | 52.85 | 49.59 | 38.46 | 43.24 | 16.07 | 40.04 |
| **OPT-6.7B** | Dense | 76.39 | 65.19 | 67.16 | 60.14 | 34.64 | 60.70 |
| | 10% | 75.42 | 63.38 | 65.16 | 57.23 | 32.89 | 58.82 |
| | 20% | 74.58 | 62.25 | 61.46 | 55.98 | 31.75 | 57.20 |
| | 30% | 71.88 | 60.56 | 59.15 | 53.62 | 28.54 | 54.75 |
| | 40% | 66.98 | 57.25 | 53.66 | 49.42 | 25.73 | 50.61 |
| | 50% | 62.78 | 54.56 | 45.63 | 46.38 | 21.75 | 46.22 |
| | 60% | 54.35 | 50.26 | 41.32 | 42.69 | 18.23 | 41.37 |
| **OPT-13B** | Dense | 76.82 | 64.80 | 69.81 | 61.87 | 35.67 | 61.79 |
| | 10% | 75.49 | 64.67 | 69.26 | 61.73 | 35.54 | 61.34 |
| | 20% | 74.89 | 64.59 | 68.95 | 61.62 | 35.41 | 61.09 |
| | 30% | 71.46 | 62.67 | 66.53 | 59.39 | 33.61 | 58.73 |
| | 40% | 68.52 | 60.39 | 63.57 | 56.12 | 28.83 | 55.49 |
| | 50% | 63.35 | 57.62 | 57.43 | 51.65 | 24.09 | 50.83 |
| | 60% | 56.28 | 53.06 | 51.69 | 47.26 | 21.87 | 46.03 |
| **LLaMA2-7B** | Dense | 79.11 | 69.06 | 75.99 | 74.58 | 46.25 | 69.00 |
| | 10% | 77.62 | 68.45 | 70.25 | 69.85 | 43.42 | 65.92 |
| | 20% | 75.38 | 67.96 | 65.26 | 65.83 | 41.57 | 63.20 |
| | 30% | 71.29 | 64.77 | 57.68 | 60.45 | 38.65 | 58.57 |
| | 40% | 67.61 | 60.38 | 54.12 | 57.53 | 35.45 | 55.02 |
| | 50% | 61.87 | 56.42 | 51.62 | 53.45 | 30.28 | 50.73 |
| | 60% | 55.39 | 51.54 | 48.73 | 47.42 | 27.08 | 46.03 |
| **LLaMA2-13B** | Dense | 80.47 | 72.22 | 79.39 | 77.48 | 49.23 | 71.76 |
| | 10% | 78.45 | 71.87 | 74.32 | 76.26 | 48.39 | 69.86 |
| | 20% | 76.12 | 70.91 | 70.06 | 74.68 | 46.51 | 67.66 |
| | 30% | 72.54 | 68.15 | 63.51 | 70.88 | 43.21 | 63.66 |
| | 40% | 71.45 | 65.52 | 62.28 | 69.13 | 41.79 | 62.03 |
| | 50% | 67.36 | 58.52 | 60.65 | 63.45 | 37.88 | 57.57 |
| | 60% | 61.89 | 55.26 | 58.06 | 57.62 | 31.74 | 52.91 |

## E Similarity Changes in Unregularized Layers

We plot the input-output similarity of the unregularized layers in LLaMA2-7B, as described in Section 4.9, in Figure 8. According to the results shown below, the similarity of all layers except for the first one is already high before regularization is applied. After applying regularization, the similarity in these unregularized layers decreases, indicating that the transformations undergone by the inputs in these layers become more substantial. This suggests that more information is being captured in these layers compared to before, implying that the regularization process may cause information to shift from the regularized layers to the unregularized ones.

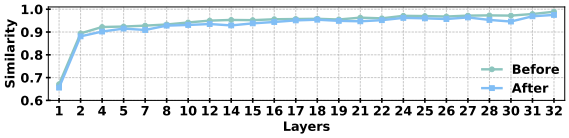


Figure 8: A comparison between existing layer-wise structured pruning methods and TRSP. Deeper blue layer represents greater performance impact, the taller cylinder represents larger data volume.

## F Cosine Similarity

$$\text{CosSim}(\mathbf{X}^{\mathbf{i}}_{\text{in}}, \mathbf{X}^{\mathbf{i}}_{\text{out}}) = \frac{1}{bn}\sum_{k=1}^{b}\sum_{j=1}^{n}\frac{\left\langle \mathbf{X}^{\mathbf{i}}_{\text{in}}[k,j,:],\ \mathbf{X}^{\mathbf{i}}_{\text{out}}[k,j,:]\right\rangle}{\left\|\mathbf{X}^{\mathbf{i}}_{\text{in}}[k,j,:]\right\|_2 \cdot \left\|\mathbf{X}^{\mathbf{i}}_{\text{out}}[k,j,:]\right\|_2} \quad (9)$$