Logits-Based Block Pruning with Affine Transformations for Large Language Models

Zekun Hu[1,2], Yichu Xu[1,2], De-Chuan Zhan[1,2]*
[1]School of Artificial Intelligence, Nanjing University
[2]National Key Laboratory for Novel Software Technology, Nanjing University
{huzk,xuyc,zhandc}@lamda.nju.edu.cn

Abstract

As the scale of Large Language Models (LLMs) continues to grow rapidly, the cost of training and inference has significantly increased, limiting their application in resource-constrained scenarios. To address this challenge, model pruning has been widely used to reduce computational complexity. Among various pruning strategies, block-wise pruning has gained popularity due to its ability to accelerate computation by removing entire blocks of parameters. However, existing methods often rely on hard labels from calibration datasets and neglect the cumulative effects of pruning on subsequent blocks. To address this, we propose two complementary techniques: the Logit Disruption Score (LDS), a novel block importance criterion that measures the impact of pruning by comparing the cosine similarity between the logits of the original and pruned models, focusing on the most informative logit dimensions to better preserve the model's core capabilities; and Activation Statistics Correction (ASC), an affine transformation mechanism that aligns the mean and variance of activations in the pruned model with those of the original model, effectively mitigating the distribution shift caused by block removal and improving the information flow in subsequent blocks. Experiments across multiple datasets show that our approach reduces reliance on calibration data and improves generalization, achieving competitive results with existing methods.

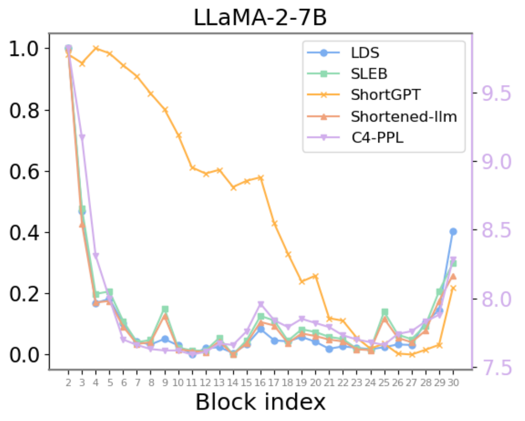

Figure 1: Comparison of normalized block importance calculated by other methods and our metric (LDS), along with model performance after pruning the corresponding blocks. The normalized importance scores correspond to the left y-axis, while perplexity(PPL) corresponds to the right y-axis. All evaluation metrics are computed on the Wikitext2 dataset.

1 Introduction

In recent years, the scale of large language models has grown rapidly (Brown et al., 2020; Touvron et al., 2023; DeepSeek-AI et al., 2025), leading to significant improvements in a wide range of natural language processing tasks (Wei et al., 2022). Despite these impressive achievements, the substantial computational resources required for both training and inference phases of LLMs present considerable obstacles to their deployment, particularly in resource-constrained environments. To address the trade-off between model performance and computational efficiency, model pruning—a well-established model compression technique—has garnered increasing attention from the research community (Han et al., 2015; Hassibi et al., 1993; LeCun et al., 1989). By targeting and removing redundant parameters or structures, pruning reduces computational and storage costs, enabling wider adoption of LLMs in practice.

Pruning techniques are generally divided into unstructured and structured approaches. Unstructured pruning (Chen et al., 2020; Frankle and Carbin, 2018; Lee et al., 2019; Li et al., 2020; Sanh et al., 2020; Sun et al., 2024) removes individual parameters to achieve high sparsity, but often fails to deliver real-world acceleration due to hardware inefficiencies (Shi et al., 2020; Wang, 2020). In contrast,

*Corresponding author, email: zhandc@lamda.nju.edu.cn.

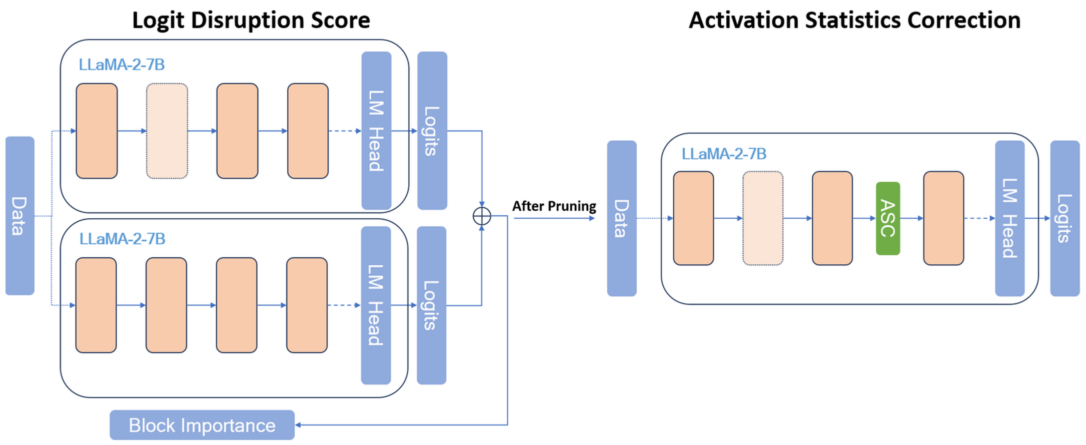

Figure 2: The structural diagrams of our proposed method where ⊕ represents the computation of cosine similarity. Our process starts with pruning the least important block, with the impact measured by the cosine similarity between the original and pruned models' logits. An affine transformation is then applied to each subsequent block to align its activation statistics (mean and variance) with those of the original model.

structured pruning eliminates entire components such as blocks or attention heads, offering better hardware compatibility and practical speedup (Gromov et al., 2024; Song et al., 2024; Yang et al., 2024b). Recently, block-wise structured pruning which removes whole transformer blocks has become a central focus for compressing large language models, as it can substantially reduce computational overhead and facilitate efficient inference (Gromov et al., 2024; Kim et al., 2024; Men et al., 2024; Song et al., 2024; Yang et al., 2024b).

Existing block-wise pruning methods often overlook critical inter-block dependencies and the broader impact of pruning decisions on model performance. For example, some approaches assess redundancy by measuring the similarity between block inputs and outputs (Men et al., 2024) or by evaluating the change in loss after removing a block (Song et al., 2024). As a result, there remains a need for pruning strategies that better preserve the model's inherent capabilities while mitigating the adverse effects on downstream blocks.

As illustrated in Figure 1, we observe that existing block-wise pruning criteria may not accurately reflect the true importance of each block in LLMs. For example, SLEB identifies block 9 and block 25 as relatively important, but perplexity, a widely recognized metric for assessing LLM capabilities, suggests that these blocks are not particularly critical. This misalignment shows that current methods, which often rely on hard labels from calibration datasets and focus on single-token loss, may overlook broader distributional information and limit the generalization ability of pruned models (Hinton et al., 2015).

Motivated by these observations, we propose two complementary techniques to address these limitations. First, we introduce the Logit Disruption Score (LDS), a novel block importance criterion that measures the similarity between model logits before and after block removal, focusing on the most informative logit dimensions to better preserve the model's core capabilities. Second, we present Activation Statistics Correction (ASC), an affine transformation mechanism that normalizes the output distribution of the subsequent blocks after pruning, ensuring it remains closely aligned with the original statistics and mitigating the adverse effects of pruning on subsequent blocks. The structure of our approach is depicted in Figure 2.

Our main contributions are summarized as follows:

- We propose a novel block importance evaluation criterion, Logit Disruption Score (LDS), which measures the impact of pruning by comparing the cosine similarity between the logits of the original and pruned models. By focusing on the most informative logit dimensions, LDS better preserves the model's core capabilities and reduces overfitting to the calibration dataset.

- We introduce Activation Statistics Correction (ASC), an affine transformation technique that aligns the mean and variance of activations in the pruned model with those of the original model, effectively mitigating the distribution shift caused by block removal and improving the information flow in subsequent blocks. ASC is also portable and can be applied to other block pruning methods to boost their performance. A key advantage is that ASC requires no training and each module adding only two scalar parameters.

2 Related Work

2.1 Block-wise Structure Pruning

Structured pruning is a common model compression technique that removes redundant neural network components for hardware efficiency. Traditional methods target fine-grained elements like neurons, channels, or attention heads, but these often fail to yield real-world speedups since their sparsity patterns do not align well with hardware operations (Ashkboos et al., 2024; Frantar and Alistarh, 2023; Ma et al., 2023). Block-wise structure pruning overcomes this by pruning entire computational blocks, such as transformer layers (Kim et al., 2024; Men et al., 2024; Yang et al., 2024b). This reduces model depth and directly improves computational efficiency and inference speed, while maintaining compatibility with standard hardware (Song et al., 2024).

Recent research has demonstrated the effectiveness of block-wise structure pruning for large language models. For example, ShortGPT (Men et al., 2024) identifies redundant blocks by measuring the similarity between each block's input and output representations, and prunes those with high similarity. Laco (Yang et al., 2024b) merges transformer blocks that produce similar outputs across a dataset, resulting in a more compact model. LLM-Streamline (Chen et al., 2025) replaces consecutive layers in a Large Language Model with a lightweight transformer layer, but this replacement requires fine-tuning. Gromov (Gromov et al., 2024) prunes consecutive blocks by evaluating the similarity of their outputs, removing those that contribute least to model diversity. Shortened LLaMA (Kim et al., 2024) uses one-shot pruning metrics, assessing the importance of each block by measuring the change in perplexity and loss on a validation set after its removal. SLEB (Song et al., 2024) introduces a metric that evaluates block importance by measuring the change in token prediction loss on a labeled calibration dataset after block removal. However, this metric's strong reliance on such hard labels which has limited information may limit its generalization and robustness (Hinton et al., 2015). In contrast, our method advances block-wise pruning by directly quantifying the similarity of logits between the original and pruned models, making pruning decisions that better preserve the model's inherent capabilities and reducing dependence on the calibration dataset.

2.2 Activation Statistics

Since the advent of deep learning, the statistics of internal activations across layers have been recognized as a fundamental factor in neural network design and optimization. Early studies showed that inappropriate activation scaling can lead to vanishing or exploding signals, making deep networks difficult to train. Accordingly, careful parameter initialization schemes were developed to preserve stable activation statistics during both forward and backward propagation, thereby enabling effective training of deep architectures (Glorot and Bengio, 2010; He et al., 2015). In parallel, normalization methods such as Batch Normalization (Ioffe and Szegedy, 2015), Layer Normalization (Ba et al., 2016), and their variants have been widely adopted to explicitly stabilize activation distributions across training steps, accelerating convergence and improving generalization. These methods work by re-centering and re-scaling activations based on statistical estimates. Moreover, techniques such as dropout were introduced, which randomly deactivate a subset of neurons during forward passes and rescale the remaining activations to preserve the variance of the overall distribution (Srivastava et al., 2014). Altogether, these techniques contribute to more stable propagation of information across layers in deep neural networks, which is critical for effective learning and gradient flow.

More recently, the importance of activation distributions has also been highlighted in the field of model fusion. REPAIR (Jordan et al., 2023) demonstrates that independently trained neural networks can exhibit significantly divergent activation statistics, leading to suboptimal performance when combined via parameter interpolation. To address this, REPAIR (Jordan et al., 2023) proposes aligning activation distributions between models, significantly improving the effectiveness of model merging. Inspired by this line of research, we extend the investigation to the context of structured pruning, where entire blocks or layers are removed from the network. Even in architectures with residual connections, we observe that pruning induces non-negligible shifts in the activation distributions of subsequent blocks. This motivates our method: a post-pruning activation correction mechanism that adjusts the activation statistics of pruned models to better match the original unpruned distribution. Importantly, this correction can be applied without any retraining, leading to consistent performance improvements.

3 Method

In this section, we present our proposed method with a focus on two key aspects. First, we introduce a novel criterion for evaluating block importance. Second, we propose an activation statistics correction mechanism to mitigate the impact of pruning on subsequent blocks.

3.1 Logit Disruption Score

Following prior work (Ashkboos et al., 2024; Kim et al., 2024; Men et al., 2024; Song et al., 2024) which focuses on structured pruning in Transformer-based language models, we also base our study on the standard decoder-only Transformer architecture exemplified by models like GPT. Specifically, these models consist of a stack of repeated Transformer blocks, followed by an additional output layer used for final prediction. We denote our model structure $\mathcal{M}$ as a standard decoder-only Transformer consisting of L stacked Transformer blocks, indexed by the set $\mathcal{I} = \{1, 2, \ldots, L\}$. The indices in $\mathcal{I}$ are ordered, and the model processes the blocks sequentially in this order. Given an input sequence $w_1, w_2, \ldots, w_N$, the model predicts the probability of the n-th token w_n conditioned on the previous tokens $w_1, w_2, \ldots, w_{n-1}$. The computation at each block is as follows:

$$h^{(i)} = f_i\left(h^{(\text{prev}(i))}\right) + h^{(\text{prev}(i))}, \quad \forall i \in \mathcal{I},$$
$$p_{\mathcal{M}_{\mathcal{I}}}(\cdot \mid w_{<n}) = f_o\left(h^{(\max(\mathcal{I}))}\right), \quad (1)$$

where $h^{(i)}$ denotes the hidden state at the i-th block, $h^{(0)}$ is the input embedding, $f_i(\cdot)$ denotes the transformation performed by the i-th block, and $\text{prev}(i)$ denotes the previous index in the ordered set $\mathcal{I}$. After passing through all blocks in $\mathcal{I}$, the model $\mathcal{M}_{\mathcal{I}}$ produces the output logits for the next-token prediction using an output head $f_o(\cdot)$.

As mentioned in Section 2, previous approaches often use hard labels from the calibration dataset as targets for block importance calculation. This practice may cause the pruned model to overfit the calibration dataset, thereby reducing its generalization ability. To address these limitations, we propose a new block importance evaluation criterion based on logit similarity that preserves the model's original outputs, termed the **L**ogit **D**isruption **S**core (LDS). This method assesses the importance of a block by computing the cosine similarity between the logits of the model after removing a block and those of the original model. The importance of the i-th block by LDS is defined as:

$$\text{LDS}(i) = -\frac{1}{N}\sum_{n=1}^{N} \text{sim}\left(p_{\mathcal{M}_{\mathcal{I}\setminus\{i\}}}, p_{\mathcal{M}_{\mathcal{I}}}\right), \quad (2)$$

where $\text{sim}(\cdot,\cdot)$ denotes the cosine similarity, $p_{\mathcal{M}_{\mathcal{I}}}$ and $p_{\mathcal{M}_{\mathcal{I}\setminus\{i\}}}$ denote the logits output by the original and pruned models respectively, given the same context $w_{<n}$, where we have omitted the arguments in the probability expression for simplification. The LDS value is computed as the average across all token positions in the calibration sequence. A larger LDS value indicates that removing the i-th block causes a greater disruption to the model's logits, meaning this block is more important to the model's performance.

However, not all dimensions in the logits are useful, and negative values in the logits can introduce noisy information (Tang et al., 2024). We recognize that after model pruning, not all logit dimensions can or should remain identical to those of the original model. This stems from two considerations: first, the original model's logit outputs for low-probability tokens may not represent reliable or useful knowledge; second, the pruning process itself can introduce noise affecting all dimensions. Therefore, we posit that the evaluation metric should primarily focus on the subset

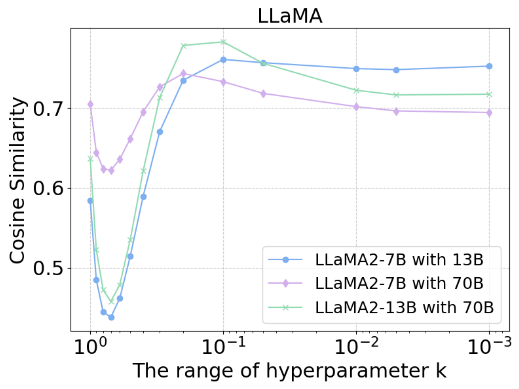

Figure 3: Cosine similarity between logits across LLaMA2 models when truncating logits with the same k on Wikitext2 dataset.

Algorithm 1 Our Proposed Method

```
Initialize: Original model $\mathcal{M}$, pruned model $\hat{\mathcal{M}}$, keep block set $\hat{\mathcal{I}} \leftarrow \mathcal{I}$, block size $L$, number of blocks to prune $n$, calibration dataset $\mathcal{D}$, hyperparameter $k$, $s_{\min} \leftarrow \infty$
for $i = 1$ to $n$ do
    for $j = 1$ to $L$ do
        $s \leftarrow \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \mathbf{LDS}_k(j; x)$
        if $s < s_{\min}$ then
            $s_{\min} \leftarrow s, idx \leftarrow j$
        end if
    end for
    $\hat{\mathcal{I}} \leftarrow \hat{\mathcal{I}} \setminus \{idx\}, L \leftarrow L - 1$
    for $l = idx + 1$ to $L$ do
        $\hat{\mathcal{M}} \leftarrow \mathbf{ASC}(\hat{\mathcal{M}}, l, \mathcal{D}, \mathcal{M})$
    end for
end for
```

Algorithm 2 Activation Statistics Correction

```
Require: Pruned model $\hat{\mathcal{M}}$, index $i$, calibration dataset $\mathcal{D}$, original model $\mathcal{M}$
    Compute original output statistics from block $i$ in $\mathcal{M}$ on $\mathcal{D}$: $\mu_i$, $\sigma_i$
    Compute output statistics from block $i$ in $\hat{\mathcal{M}}$ on $\mathcal{D}$: $\hat{\mu}_i$, $\hat{\sigma}_i$
    Apply affine transformation $\mathcal{A}(\mathbf{x}) = \frac{\sigma_i}{\hat{\sigma}_i} \cdot (\mathbf{x} - \hat{\mu}_i) + \mu_i$
    Replace block $i$ with $\mathcal{A} \circ h^{(i)}$ in $\hat{\mathcal{M}}$
    return $\hat{\mathcal{M}}$
```

of the vocabulary corresponding to higher prediction probabilities. Observations from Figure 3 lend support to this idea, showing higher logit similarity in the high-value regions across different models. This observation suggests that low-probability dimensions primarily contribute noise, whereas the similarity between pruned and original models is better captured by their most confident predictions. Accordingly, we introduce a hyperparameter k to restrict the comparison to the top $k\%$ dimensions of the logit vector. The resulting Logit Disruption Score is defined as

$$\mathrm{LDS}_k(i) = -\frac{1}{N} \sum_{n=1}^{N} \mathrm{sim}\Big(K(p_{\mathcal{M}_{\mathcal{I}}}),\, K\Big(p_{\mathcal{M}_{\mathcal{I} \setminus \{i\}}}\Big)\Big), \tag{3}$$

where $K(\cdot)$ retains only the top $k\%$ largest logit values and sets all remaining dimensions to zero. We discuss the comparison between softmax and TopK in Appendix A.4. Since the importance of blocks is dynamic, we adopt an iterative pruning method as SLEB (Song et al., 2024). That is, after each pruning step, we recalculate the importance of the blocks and perform another pruning, continuing until the desired pruning rate is reached. Note that during the iteration process, our method always computes the importance based on the original model's logits. A comparison between iterative pruning and single pruning is provided in Appendix A.1.

3.2 Activation Statistics Correction

Inspired by recent work on activation statistics, we investigate how pruning blocks from a Transformer affects the distribution of activations in subsequent blocks. Similar to prior studies (Jordan et al., 2023), we assume that the activations follow a Gaussian distribution. As shown in the left part of Figure 4, when more blocks are removed, the mean and variance of the output activations from the final block increasingly deviate from those of the original model. This observation suggests that the removal of blocks leads to a growing mismatch in the activation statistics, which may hinder the information flow in the remaining network.

To alleviate the impact of block removal on the distribution of activations in subsequent blocks, we propose an **A**ctivation **S**tatistics **C**orrection (ASC) method. This method is used to adjust the mean and variance of activations after pruning, so that the pruned model's activations better match those of the original model, helping to reduce performance loss caused by distribution shifts. Specifically, for the i-th block, we denote the mean and standard deviation of its output in the original model (computed on the calibration dataset) as μ and σ, and those in the pruned model as $\hat{\mu}$ and $\hat{\sigma}$, respectively. We define an affine transformation layer $\mathcal{A}$ that adjusts the activation statistics as follows:

$$\mathcal{A}(\mathbf{x}) = \frac{\sigma}{\hat{\sigma}} \cdot (\mathbf{x} - \hat{\mu}) + \mu \tag{4}$$

where μ, σ, $\hat{\mu}$, and $\hat{\sigma}$ are scalars. Then, we compose this affine layer with the block hidden states $h^{(i)}$ that $\tilde{h}^{(i)} = \mathcal{A} \circ h^{(i)}$, which is applied to the blocks subsequent to the pruned one.

By applying this correction, we ensure that after removing a block, the distribution of neuron activations in the output becomes closer to that of the original model, thereby allowing subsequent blocks to effectively transmit information. As shown in the right subfigure of Figure 4, with the increase in the number of pruned blocks, our method consistently leads to improvements in PPL. The detailed procedure of the ASC algorithm is presented in Algorithm 2.

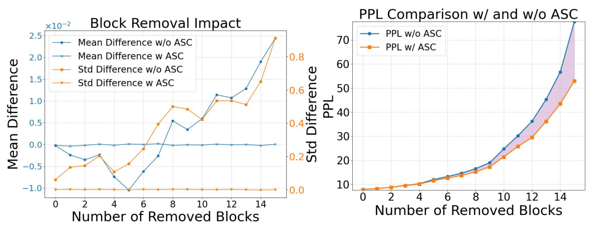

Figure 4: Changes in the mean and variance of the last block output as the number of pruned blocks increases using SLEB on LLaMA-2-7B, and the effect of ASC on these statistics; PPL with and without ASC on LLaMA-2-7B pruned with LDS, evaluated on a subset of the Wikitext2 dataset.

4 Experiments

Setup: Our method is implemented using PyTorch (Paszke et al., 2019) and Hugging Face Transformers (Wolf et al., 2020). All experiments are conducted on two NVIDIA RTX 6000 Ada Generation GPUs with 48GB of memory. For calibration dataset, we follow SLEB (Song et al., 2024) and use 128 randomly sampled examples from the traning set of WikiText2 (Merity et al., 2016). We use another 256 samples from the same training set as the validation set. The range of hyperparameter k is set to $[0.001, 0.01, 0.1]$. We selected k based on the PPL of the validation set. The impact of the hyperparameter k on the algorithm is discussed in Appendix A.1.

Models: To evaluate the effectiveness of our approach, we conducted experiments on the Qwen2.5 family (Yang et al., 2024a) and the LLaMA family (Touvron et al., 2023). The Qwen2.5 family includes models with 1.5B, 3B, 7B and 32B parameters, while the LLaMA family includes models with LLaMA-2-7B, LLaMA-2-13B, LLaMA-2-70B, LLaMA-3.1-8B and LLaMA-3.2-3B.

Benchmark: Following SLEB (Song et al., 2024), we evaluate model performance using both perplexity and zero-shot accuracy. For perplexity evaluation, we use the C4 dataset (Raffel et al., 2019). For zero-shot evaluation, we adopt a suite of benchmarks: PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2021), HellaSwag (Zellers et al., 2019), ARC-easy, and ARC-challenge (Clark et al., 2018). All evaluations are carried out using the LM Evaluation Harness (Gao et al., 2024) with default settings.

Baseline: We compare our method with the previous channel-wise pruning method SliceGPT (Ashkboos et al., 2024) and the block-wise pruning without training method ShortGPT (Men et al., 2024), Shortened LLaMA (Kim et al., 2024) and SLEB (Song et al., 2024). Shortened LLaMA proposes two methods, and we chose the one with better performance as the baseline. This method measures the importance of a block based on the PPL after pruning.

4.1 Generation Performance

We evaluate generation performance by measuring perplexity at varying sparsity levels on models from the Qwen2.5 and LLaMA series. Following SLEB (Song et al., 2024), when the product of target sparsity and the number of transformer blocks is non-integer, we round up to determine the number of blocks to prune.

Table 1 presents perplexity results on the C4 dataset, which is chosen to assess generalization since it differs from the calibration data. The results on the Wikitext2 dataset are presented in Table 13. Notably, the official implementation of SliceGPT lacks compatibility with the Qwen2.5 and LLaMA 3.x series, underscoring its limited applicability. In contrast, block-wise pruning methods are architecture-agnostic and better suited for transformer models. SliceGPT also suffers from significant PPL degradation at high sparsities, likely due to overfitting from its fine-grained channel-level pruning.

Compared to other block-wise approaches, our method shows robust performance across models and sparsity levels, especially at 50% sparsity. Prior methods are often limited by their importance metrics: ShortGPT relies on local input-output similarity without considering the global impact of pruning on the entire model, while Shortened LLaMA uses one-shot pruning based on perplexity, ignoring dynamic block interactions. SLEB improves this with iterative pruning but computes importance only from calibration loss, using limited hard label information.

Conversely, our approach evaluates block importance utilizing logits as soft labels, thereby capturing richer information that focuses on maintaining predictive performance rather than fitting to the calibration data alone. Additionally, we introduce a method to adjust subsequent blocks' activation statistics to mitigate cumulative degradation effects from pruning, a critical factor often ignored in prior

Table 1: Perplexity results on C4 dataset. The data in bold represents the best performance at the same sparsity level. The comparison is conducted within the scope of methods where the pruning unit is T.Block (T.Block: Transformer Block. Note: For brevity, "Shortened LLaMA" is abbreviated as "Shortened" throughout all tables).

Method	**Pruning**	**Sparsity**	**Qwen2.5**				**LLaMA-2**			**LLaMA-3**		**Avg.**
	Unit		**1.5B**	**3B**	**7B**	**32B**	**7B**	**13B**	**70B**	**3B**	**8B**	
Dense	-	0%	15.12	13.36	11.88	10.17	7.26	6.73	5.71	11.33	9.54	10.12
SliceGPT	Channel	20%	-	-	-	-	26.06	22.90	15.84	-	-	-
SliceGPT	Channel	25%	-	-	-	-	32.74	29.86	20.03	-	-	-
SliceGPT	Channel	30%	-	-	-	-	41.69	38.43	25.79	-	-	-
ShortGPT	T. Block	10%	22.49	17.12	13.65	10.86	10.31	7.98	6.52	18.94	15.84	13.75
Shortened	T. Block	10%	19.75	16.62	13.71	10.85	9.36	7.93	6.40	17.34	15.02	13.00
SLEB	T. Block	10%	17.70	**15.79**	13.66	10.88	9.34	7.78	6.43	16.07	13.61	12.36
Ours	T. Block	10%	**17.53**	16.19	**13.64**	**10.84**	**9.09**	**7.77**	**6.36**	**15.56**	**13.44**	**12.27**
ShortGPT	T. Block	30%	50.02	45.54	28.98	19.83	27.93	15.55	10.08	213.72	124.49	59.57
Shortened	T. Block	30%	35.97	25.50	23.85	19.56	32.51	14.91	9.41	69.10	33.40	29.36
SLEB	T. Block	30%	30.04	24.79	23.38	14.65	17.42	11.84	8.95	38.76	25.35	21.69
Ours	T. Block	30%	**29.95**	**24.36**	**21.99**	**14.10**	**15.29**	**11.10**	**8.82**	**36.07**	**23.65**	**20.59**
ShortGPT	T. Block	50%	799.88	452.12	36742.46	104.84	1325.62	199.98	34.24	1445.57	2086.18	4798.99
Shortened	T. Block	50%	98.20	301.89	181.14	854.99	156.25	61.98	16.51	447.70	251.33	263.33
SLEB	T. Block	50%	85.74	85.21	70.32	33.47	85.96	31.48	18.57	158.63	136.29	78.41
Ours	T. Block	50%	**80.69**	**76.60**	**60.46**	**25.93**	**41.21**	**21.08**	**14.65**	**116.97**	**80.50**	**57.57**

methods. Consequently, our approach achieves superior average perplexity results across all sparsity levels, demonstrating enhanced generalization capabilities.

4.2 Zero-shot Task Evaluation

To further validate the generalization capabilities of pruned models, we conduct a comprehensive evaluation on various zero-shot tasks. Table 2 summarizes the average accuracy achieved by various pruning techniques, along with the inference speedup ratios, which were empirically measured using the LLaMA-2-7B model.

Our evaluation spans multiple model scales within the Qwen2.5 and LLaMA families, providing a detailed comparison of both channel-wise and block-wise pruning methodologies. Consistent with prior findings (Song et al., 2024), block-wise pruning methods exhibit a distinct advantage over channel-wise approaches in terms of computational acceleration, achieving notably higher inference speedup at equivalent sparsity levels. When matched for similar inference acceleration, block-wise methods consistently yield superior zero-shot task accuracy, reinforcing the practical efficiency and effectiveness of block-level structural pruning. Importantly, among the block-wise pruning methods evaluated, our proposed approach consistently delivers superior average accuracy across all sparsity configurations. These robust performance underscores our method's potential as optimized pruning strategies for LLMs, effectively preserving the generalization ability of pruned models. The indices of the pruned blocks for different methods and models are listed in Table 12.

4.3 Ablation Study

We conduct ablation experiments to evaluate the contribution of each component in our pruning method. Table 3 compares the perplexity of SLEB and our approach, with and without ASC, across different sparsity levels on the Wikitext2 and C4 datasets using LLaMA-2 models.

"Ours w/o ASC" shows the effect of the LDS metric, which consistently outperforms SLEB, confirming that our logit-based measure better identifies the importance of blocks with minimal impact on model performance and generalization. With ASC enabled ("Ours w/ ASC"), perplexity further decreases, demonstrating that adjusting activation statistics effectively mitigates the cumulative degradation from block pruning. This supports our view that correcting activation shifts is critical for maintaining stable information flow in pruned transformers. The experimental results of applying ASC to other methods are provided in Appendix A.2.

Additionally, following the methodology outlined in previous work (Song et al., 2024), we perform fine-tuning experiments on the LLaMA-2 models pruned at a sparsity level of 50%. Specifically, fine-tuning is carried out using Low-Rank Adaptation (LoRA) (Hu et al., 2021) with rank 8,

Table 2: Mean accuracy (%) on zero-shot tasks. The data in bold represents the best performance at the same sparsity level. The comparison is conducted within the scope of methods where the pruning unit is T.Block.

Method	Pruning Unit	Sparsity	Speedup	Qwen2.5				LLaMA-2			LLaMA-3		Avg.
				1.5B	3B	7B	32B	7B	13B	70B	3B	8B	
Dense	-	0%	1.00x	64.87	68.17	72.12	75.56	69.00	71.76	76.57	67.80	73.66	71.06
SliceGPT	Channel	20%	1.05x	-	-	-	-	58.17	63.45	72.34	-	-	-
SliceGPT	Channel	25%	1.11x	-	-	-	-	55.49	58.90	69.75	-	-	-
SliceGPT	Channel	30%	1.15x	-	-	-	-	51.50	55.16	66.11	-	-	-
ShortGPT	T. Block	10%	1.13x	56.97	62.52	66.37	75.41	**64.15**	69.68	70.11	57.66	68.45	65.70
Shortened	T. Block	10%	1.13x	58.80	60.43	66.96	**76.27**	61.56	**69.69**	73.57	54.55	60.62	64.72
SLEB	T. Block	10%	1.13x	59.76	61.66	67.31	75.19	62.13	66.72	73.44	57.21	**68.05**	65.72
Ours	T. Block	10%	1.13x	**59.80**	**61.91**	**67.38**	76.06	61.99	66.73	**74.04**	**57.76**	67.16	**65.87**
ShortGPT	T. Block	30%	1.42x	46.77	49.47	51.48	65.70	51.29	59.34	66.05	45.04	50.48	53.96
Shortened	T. Block	30%	1.42x	48.97	52.30	52.16	61.09	47.63	60.53	66.05	44.81	50.75	53.81
SLEB	T. Block	30%	1.42x	**49.49**	**53.17**	51.63	65.98	51.93	58.93	66.55	46.22	**51.35**	55.03
Ours	T. Block	30%	1.42x	48.91	52.63	**52.22**	**67.26**	**52.58**	**60.57**	**67.34**	**46.33**	51.10	**55.44**
ShortGPT	T. Block	50%	1.61x	39.25	38.93	36.68	42.89	39.11	41.14	50.09	38.31	40.09	40.72
Shortened	T. Block	50%	1.61x	40.62	40.26	39.68	36.73	38.26	44.87	54.64	39.06	39.66	41.53
SLEB	T. Block	50%	1.61x	41.08	**42.06**	41.82	50.17	40.68	46.46	55.57	39.14	39.23	44.02
Ours	T. Block	50%	1.61x	**41.14**	41.91	**42.16**	**55.47**	**43.91**	**47.64**	**56.71**	**40.71**	**42.57**	**45.80**

Table 3: Comparison of SLEB and our method (w/ and w/o ASC) under different sparsity levels on Wikitext2 and C4 datasets for LLaMA-2 models((–): without fine-tuning; (✓): with fine-tuning).

Method	Sparsity &FT	LLaMA-2-7B		LLaMA-2-13B	
		Wikitext2	C4	Wikitext2	C4
SLEB	10%(-)	6.95	9.34	5.63	7.80
Ours w/o ASC	10%(-)	6.79	9.13	5.63	7.79
Ours w/ ASC	10%(-)	**6.77**	**9.09**	**5.58**	**7.77**
SLEB	30%(-)	13.82	17.42	8.85	11.84
Ours w/o ASC	30%(-)	13.94	16.12	8.73	11.75
Ours w/ ASC	30%(-)	**12.68**	**15.29**	**8.26**	**11.10**
SLEB	50%(-)	106.2	85.96	27.83	31.48
Ours w/o ASC	50%(-)	72.65	59.51	28.10	28.51
Ours w/ ASC	50%(-)	**41.77**	**41.21**	**18.60**	**21.08**
SLEB	50%(✓)	18.14	20.20	12.75	15.16
Ours w/o ASC	50%(✓)	16.86	19.26	**12.21**	15.13
Ours w/ ASC	50%(✓)	**16.38**	**18.75**	12.27	**15.07**

trained for a single epoch on the Alpaca dataset. The experimental results after fine-tuning, shown in Table 3, indicate that models pruned using our method consistently outperform those pruned by SLEB. This further underscores the robustness and effectiveness of our pruning methodology, highlighting its ability to retain and even enhance the fine-tuned model's performance.

4.4 Impact of Calibration Dataset

To systematically explore how the choice of calibration data influences the effectiveness of pruning methods, we conduct an extensive set of experiments using two distinct calibration datasets: WikiText2 and C4. Specifically, we evaluate pruning performance at a fixed sparsity level of 30%, applying both our proposed method and the SLEB baseline to two representative model architectures, LLaMA-2-7B and Qwen2.5-7B. The results of these experiments are summarized and illustrated in Figure 5. The experimental results reveal that our method exhibits substantially smaller performance fluctuations when changing the calibration dataset, compared to SLEB. Specifically, while the SLEB method demonstrates considerable sensitivity to the choice of calibration data, our method remains consistently robust, displaying minimal variation in performance. This reduced sensitivity implies that our approach effectively mitigates the risk of calibration-data overfitting and is capable of delivering stable pruning outcomes across varied datasets, further underscoring its practical robustness and reliability.

Moreover, we investigate the impact of the size of the calibration dataset on pruning effectiveness. To this end, we systematically vary the number of samples included in the calibration dataset, maintaining a fixed sparsity ratio of 30% on the LLaMA-2-7B model. To isolate the effect of calibration dataset size, we directly utilize the calibration set as the validation set for hyperparameter k tuning, ensuring no additional data is introduced into the process. Results from this analysis are depicted in Figure 6. Our observations confirm that, as expected,

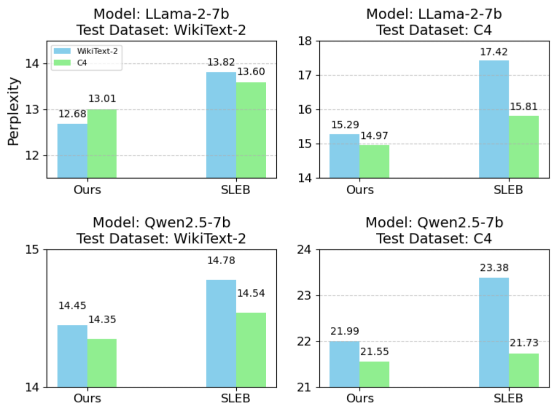

Figure 5: Perplexity results with different calibration datasets.

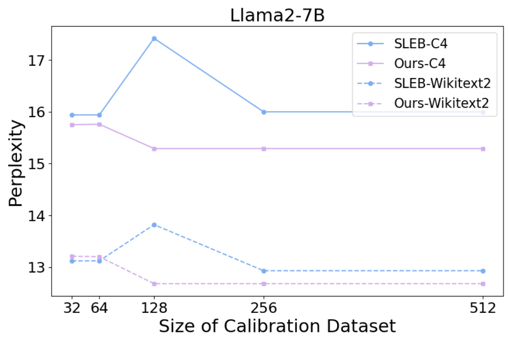

Figure 6: The perplexity results of pruning on Llama2-7B using different calibration dataset sizes.

both our method and SLEB exhibit improved pruning performance as the calibration dataset size increases, demonstrating that a larger calibration set generally provides richer information to better guide pruning decisions. However, notably, our method consistently outperforms SLEB across almost all calibration sizes and, importantly, achieves superior results even with fewer calibration examples. This clearly highlights that our method is not only more effective but also more data-efficient, reducing reliance on extensive calibration datasets to achieve robust pruning results.

5 Conclusion

In this paper, we have introduced the Logit Disruption Score (LDS) and Activation Statistics Correction (ASC), two complementary techniques aimed at enhancing block-wise pruning for large language models (LLMs). LDS provides a novel criterion for block importance evaluation by focusing on logit similarity rather than hard labels, effectively capturing the core predictive capabilities of LLMs and reducing overfitting to calibration datasets. ASC addresses distributional shifts caused by block removal through affine transformations, ensuring stability and information flow in subsequent blocks without retraining. Our extensive experiments across multiple datasets and model architectures, including the Qwen2.5 and LLaMA families, demonstrate that our proposed methods significantly outperform existing pruning strategies. LDS effectively identifies redundant blocks, and ASC consistently mitigates pruning-induced performance degradation without training, thus maintaining superior generalization capabilities even at higher sparsity levels. Furthermore, our approach shows robustness across various calibration datasets and dataset sizes, highlighting its practical applicability. Overall, our findings underline the effectiveness of logits-based pruning strategies and distributional corrections for achieving computationally efficient and high-performing LLMs suitable for deployment in resource-constrained scenarios. Future research can further explore novel performance recovery techniques to fully maximize the potential of pruned models.

Limitations

Our work has several limitations. First, the LDS component introduces a hyper-parameter k, which requires tuning on a small calibration dataset. This process marginally increases the pruning time. While the overhead is negligible for small-scale models, it becomes more pronounced for large-scale models (e.g., 32B or 70B parameters). Second, the proposed ASC module, despite having only two scaling parameters, necessitates alterations to the original model architecture. A promising future direction would be to integrate its functionality into existing model components. Finally, while our method is training-free and yields significant perplexity (PPL) improvements, its gains on zero-shot tasks are not as substantial. We note that this evaluation follows the SLEB benchmark setup, which utilizes a limited number of zero-shot tasks, potentially affecting the comprehensiveness of our task performance assessment.

References

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. *Preprint*, arXiv:2401.15024.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *Preprint*, arXiv:1607.06450.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about Physical Commonsense in Natural Language. *AAAI*, 34(05):7432–7439.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language Models are Few-Shot Learners. In *NeurIPS*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The Lottery Ticket Hypothesis for Pre-trained BERT Networks. In *NeurIPS*, volume 33, pages 15834–15846. Curran Associates, Inc.

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. 2025. Streamlining Redundant Layers to Compress Large Language Models. *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. *Preprint*, arXiv:1803.05457.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. DeepSeek-V3 Technical Report. *Preprint*, arXiv:2412.19437.

Jonathan Frankle and Michael Carbin. 2018. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *ICLR*.

Elias Frantar and Dan Alistarh. 2023. SparseGPT: Massive Language Models Can be Accurately Pruned in One-Shot.

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. 2024. The language model evaluation harness.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. 2024. The Unreasonable Ineffectiveness of the Deeper Layers. *Preprint*, arXiv:2403.17887.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both Weights and Connections for Efficient Neural Network. In *NeurIPS*, volume 28. Curran Associates, Inc.

B. Hassibi, D.G. Stork, and G.J. Wolff. 1993. Optimal Brain Surgeon and general network pruning. In *IJCNN*, pages 293–299 vol.1.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *Preprint*, arXiv:1502.01852.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. *Preprint*, arXiv:1503.02531.

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.

Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Preprint*, arXiv:1502.03167.

Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. 2023. REPAIR: REnormalizing Permuted Activations for Interpolation Repair.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. 2024. Shortened LLaMA: Depth Pruning for Large Language Models with Comparison of Retraining Methods. *Preprint*, arXiv:2402.02834.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal Brain Damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. 2019. SNIP: Single-shot Network Pruning based on Connection Sensitivity. *Preprint*, arXiv:1810.02340.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. 2020. Train Big, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers. In *ICML*, pages 5958–5968. PMLR.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. LLM-Pruner: On the Structural Pruning of Large Language Models. *Preprint*, arXiv:2305.11627.

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.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer Sentinel Mixture Models. *Preprint*, arXiv:1609.07843.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, volume 32. Curran Associates, Inc.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.

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.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement Pruning: Adaptive Sparsity by Fine-Tuning. In *NeurIPS*, volume 33, pages 20378–20389. Curran Associates, Inc.

Shaohuai Shi, Qiang Wang, and Xiaowen Chu. 2020. Efficient Sparse-Dense Matrix-Matrix Multiplication on GPUs Using the Customized Sparse Storage Format. *Preprint*, arXiv:2005.14469.

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.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024. A Simple and Effective Pruning Approach for Large Language Models. *Preprint*, arXiv:2306.11695.

Chenxia Tang, Jianchun Liu, Hongli Xu, and Liusheng Huang. 2024. Top-$nσ$: Not All Logits Are You Need. *Preprint*, arXiv:2411.07641.

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. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *Preprint*, arXiv:2307.09288.

Ziheng Wang. 2020. SparseRT: Accelerating Unstructured Sparsity on GPUs for Deep Learning Inference. *Preprint*, arXiv:2008.11849.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent Abilities of Large Language Models. *Preprint*, arXiv:2206.07682.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *Preprint*, arXiv:1910.03771.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024a. Qwen2 Technical Report. *Preprint*, arXiv:2407.10671.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024b. LaCo: Large Language Model Pruning via Layer Collapse. In *EMNLP*, pages 6401–6417, Miami, Florida, USA. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? *Preprint*, arXiv:1905.07830.

A Appendix

A.1 Hyperparameter and Pruning Method Analysis

To comprehensively examine our pruning method, we perform two supplementary analyses on the LLaMA-2-7B model to maintain consistency in experimental conditions.

We first investigate the sensitivity of our method to the hyperparameter k, which controls the proportion of logit dimensions retained when calculating LDS. The left subfigure of Figure 7 illustrates perplexity scores on the validation set at a sparsity level of 30%, as well as the WikiText2 and C4 test sets, across various values of k. The results clearly indicate a strong alignment between validation and test performances, demonstrating that the optimal k identified on the validation set reliably generalizes to the test scenarios. Notably, employing only the

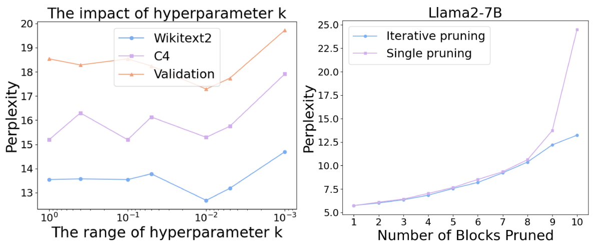

Figure 7: The impact of hyperparameter k on our method; Comparison of perplexity between iterative and single pruning on LLaMA-2-7B

top 1% of logit dimensions ($k = 0.01$) yields superior performance compared to utilizing the entire logit vector ($k = 1$). This suggests that essential predictive information is highly concentrated within a small subset of dimensions exhibiting the highest logit values, while lower-value dimensions predominantly contain less relevant or noisy signals. Nevertheless, an excessively restrictive choice of k leads to performance deterioration, underscoring the necessity of balancing the informativeness of retained logit dimensions with noise exclusion.

We further explore the impact of iterative pruning compared to single-step pruning by evaluating their respective perplexities on the WikiText2 dataset without employing ASC. The comparative results, depicted in the right subfigure of Figure 7, show that at lower pruning ratios, iterative and single-step pruning perform similarly. However, as the sparsity level increases, iterative pruning significantly outperforms single-step pruning, achieving notably lower perplexity. This observation highlights iterative pruning's ability to dynamically reassess and mitigate cumulative performance losses arising from successive block removals. Despite iterative pruning requiring higher computational overhead due to repeated block importance recalculations, it provides substantial performance benefits at high sparsity levels.

A.2 Other Methods with ASC

To evaluate the flexibility of our method, we applied ASC to several baseline methods and conducted experiments on models including the Qwen2.5 and LLaMA families. As summarized in Table 4, ASC leads to systematic improvements across all tested baselines, consistently enhancing their performance. These results underscore the broad applicability and effectiveness of our proposed ASC approach.

Table 4: Comparison of the baseline method with and without ASC at 50% sparsity: reporting PPL on C4 and mean accuracy (as in Table 2). ((–): without ASC; (✓): with ASC).

Method	**Qwen2.5-7B**		**LLaMA-2-7B**		**LLaMA-3.1-8B**	
	PPL↓	**Acc.**	**PPL↓**	**Acc.**	**PPL↓**	**Acc.**
ShortGPT(-)	36742.46	36.68	1325.62	39.11	2086.18	40.09
ShortGPT(✓)	**7199.78**	**36.80**	**129.75**	**40.80**	**522.95**	**41.58**
Shortened(-)	181.14	39.68	156.25	38.26	251.33	39.66
Shortened(✓)	**147.40**	**40.03**	**143.62**	**39.34**	**157.88**	**40.43**
SLEB(-)	70.32	41.82	85.96	40.68	136.29	39.23
SLEB(✓)	**58.40**	**43.08**	**56.49**	**41.70**	**87.59**	**40.92**

A.3 Analysis of Logits vs. Last-Block Outputs

To further substantiate our methodological choice of utilizing model logits rather than solely relying on the final block's intermediate representations for block importance estimation, we introduce an additional evaluation metric, termed the **O**utput **D**isruption **S**core (ODS). Formally, ODS measures the disruption caused by pruning a given transformer block through the cosine similarity of representations produced by the model's final transformer block before and after pruning, defined as:

$$\mathrm{ODS}(i) = 1 - \frac{1}{N}\sum_{n=1}^{N} \mathrm{sim}\left(h_{\mathcal{I}\backslash\{i\}}^{(\max(\mathcal{I}))}, h_{\mathcal{I}}^{(\max(\mathcal{I}))}\right), \quad (5)$$

While this approach may intuitively capture local block-level effects, relying solely on the last-block output representations has several critical limitations compared to using logits. Firstly, logits possess a substantially higher dimensionality relative to intermediate block outputs, inherently encompassing richer semantic and predictive information. Secondly, each logit dimension directly corresponds to a specific vocabulary token, enabling straightforward identification and selective retention of highly informative logits, while simultaneously providing a convenient mechanism to discard less informative (low-value) dimensions, thus effectively mitigating noise. Empirical results presented in Table 5 further support these theoretical considerations. The performance evaluation consistently indicates that logit-based block pruning achieves superior results compared to the alternative method utilizing only the final transformer block outputs. This empirical evidence highlights that logits provide a more comprehensive representation of the model's overall predictive capabilities, thereby allowing pruning decisions to more effec-

Table 5: Comparison of ODS and LDS for block importance on C4 PPL and zero-shot accuracies for LLaMA-2 models at a sparsity level of 30%.

Method	LLaMA-2-7B		LLaMA-2-13B		LLaMA-2-70B	
	PPL↓	Acc.	PPL↓	Acc.	PPL↓	Acc.
ODS	16.49	52.55	11.95	58.57	9.50	66.18
LDS	**15.50**	**52.73**	**11.70**	**58.59**	**8.90**	**66.56**

Table 6: Comparison of Softmax and TopK strategies in LDS at 50% sparsity: reporting PPL on C4 and mean accuracy (as in Table 2).

Method	Qwen2.5-7B		LLaMA-2-7B		LLaMA-3.1-8B	
	PPL↓	Acc.	PPL↓	Acc.	PPL↓	Acc.
Ours(Softmax)	62.27	42.12	51.31	40.62	**80.47**	41.08
Ours(TopK)	**60.46**	**42.16**	**41.21**	**43.91**	80.50	**42.57**

tively preserve the inherent functionality of the original, unpruned model.

A.4 Analysis of Softmax vs. TopK

We evaluated the effectiveness of using Softmax for noise suppression in LDS and compared it with the Top-k approach. As shown in Table 6, Softmax generally underperforms compared to Top-k. This may be attributed to the tendency of Softmax to transform the original multi-modal distribution of logits into a unimodal one, leading to some information loss and thus suboptimal performance under high sparsity.

A.5 Downstream Task Gains Corresponding to Table 3

In addition to perplexity results reported in Table 3, we further evaluate the downstream task performance of pruned models under the same settings. This analysis provides a more comprehensive assessment of model quality beyond language modeling metrics.

Table 7 reports the mean zero-shot accuracy on downstream benchmarks for models pruned at 50% sparsity, both before and after fine-tuning. The results demonstrate that Activation Statistics Correction consistently improves downstream performance not only immediately after pruning, but also after subsequent fine-tuning. This indicates that ASC effectively preserves the model's trainability and latent knowledge, enabling pruned models to better recover and utilize their original capabilities during adaptation.

Table 7: Downstream zero-shot accuracy (%) corresponding to Table 3 at 50% sparsity for LLaMA-2 models ((–): without fine-tuning; (✓): with fine-tuning).

Method	Setting	LLaMA-2-7B	LLaMA-2-13B
SLEB	50%(-)	40.68	46.46
Ours w/o ASC	50%(-)	42.32	47.54
Ours w/ ASC	50%(-)	**43.91**	**47.64**
SLEB	50%(✓)	45.94	52.47
Ours w/o ASC	50%(✓)	47.41	52.54
Ours w/ ASC	50%(✓)	**49.11**	**53.94**

Table 8: Vocabulary sizes of the evaluated models.

Model	Vocab Size
Qwen2.5-0.5B	151,936
Qwen2.5-3B	151,936
Qwen2.5-7B	152,064
Qwen2.5-32B	152,064
LLaMA-2	32,000
LLaMA-3	128,256

A.6 Vocabulary Size and Generation Throughput

Table 8 lists the vocabulary sizes of the language models evaluated in this work.

In addition to prompt processing efficiency reported in the main paper, we further evaluate token generation throughput to better reflect practical deployment scenarios. We report throughput in tokens per second (tokens/s) for LLaMA-2-7B on NVIDIA RTX 6000 Ada Generation GPUs.

The evaluation setup consists of autoregressively generating sequences of 128 tokens with a batch size of 64. Results are summarized in Table 9. Block-wise pruning achieves substantial throughput improvements at higher sparsity levels, significantly outperforming channel-wise pruning under the same hardware configuration.

A.7 Evaluation on Conversational Reasoning and Instruction Following

To further assess the instruction-following and conversational reasoning capabilities of pruned models, we conduct additional evaluations on the MuTual dataset. MuTual is a multi-turn dialogue reasoning benchmark that requires models to select the most appropriate response given a conversational context.

We evaluate models from different model families under various sparsity levels and report both Recall@2 (R@2) and Mean Reciprocal Rank (MRR), which are standard metrics for MuTual.

Table 9: Generation throughput (tokens/s) on LLaMA-2-7B with different pruning methods.

Method	**Sparsity**	**Tokens/s**
Dense	–	1254
SliceGPT	20%	1332
SliceGPT	25%	1404
SliceGPT	30%	1442
Block-wise pruning	10%	1386
Block-wise pruning	30%	1623
Block-wise pruning	50%	2349

Table 10: Performance on the MuTual dataset for evaluating multi-turn conversational reasoning. R@2 and MRR are reported for different models and sparsity levels.

Method	**Sparsity**	**LLaMA-2-7B**		**Qwen2.5-7B**		**LLaMA-3-8B**	
		R@2	**MRR**	**R@2**	**MRR**	**R@2**	**MRR**
SLEB	10%	43.79	0.6741	42.89	**0.6942**	41.31	**0.6804**
Ours	10%	**43.91**	**0.6889**	**43.91**	0.6910	**42.33**	0.6789
SLEB	30%	43.45	**0.6423**	43.57	0.6193	**43.68**	0.6284
Ours	30%	**45.32**	0.6356	**44.47**	**0.6317**	43.68	**0.6321**
SLEB	50%	43.45	0.5905	43.34	0.6082	43.68	0.5923
Ours	50%	**44.36**	**0.6052**	**43.52**	**0.6092**	**44.58**	**0.6045**

The results are summarized in Table 10. Overall, our method achieves better performance than the SLEB baseline in most settings, particularly at higher sparsity levels, indicating improved preservation of conversational reasoning ability after pruning.

A.8 Evaluation on Reasoning-Heavy Tasks

We further extend our evaluation to reasoning-heavy benchmarks to examine whether the proposed pruning method preserves complex reasoning capabilities. We focus on two representative reasoning-intensive benchmarks: GSM8K with chain-of-thought prompting (gsm8k_cot_zeroshot) and Big-Bench Hard (bhh_zeroshot). Both benchmarks are widely used to assess multi-step numerical and logical reasoning. The results are summarized in Table 11.

Overall, our method consistently outperforms the SLEB baseline on reasoning-heavy tasks, particularly at higher sparsity levels. These results indicate that our approach is effective at preserving and even enhancing reasoning-related capabilities in heavily pruned models.

Table 11: Performance on reasoning-heavy benchmarks under different sparsity levels. Results are reported for GSM8K (chain-of-thought, zero-shot) and Big-Bench Hard (zero-shot).

Method	**Sparsity**	**LLaMA-2-7B**		**LLaMA-2-13B**	
		gsm8k_cot	**bhh**	**gsm8k_cot**	**bhh**
SLEB	10%	**1.59**	20.26	5.40	30.76
Ours	10%	**1.59**	**21.87**	**6.09**	**31.23**
SLEB	30%	1.21	15.14	**1.67**	22.73
Ours	30%	**1.33**	**18.00**	**1.67**	**23.50**
SLEB	50%	0.45	2.80	**0.91**	8.97
Ours	50%	**0.83**	**6.79**	0.53	**15.34**

Table 12: The indices of the first nine blocks pruned by different methods on various models at 50% sparsity.

Method	Model	Block Index
ShortGPT	Qwen2.5-1.5B	16, 15, 26, 14, 12, 24, 13, 17, 11
Shortened	Qwen2.5-1.5B	14, 16, 12, 15, 10, 9, 18, 11, 17
SLEB	Qwen2.5-1.5B	14, 15, 16, 13, 12, 11, 10, 9, 17
Ours	Qwen2.5-1.5B	14, 13, 15, 16, 12, 11, 10, 9, 17
ShortGPT	Qwen2.5-3B	21, 2, 22, 19, 20, 17, 34, 18, 29
Shortened	Qwen2.5-3B	17, 19, 22, 21, 18, 20, 24, 3, 15
SLEB	Qwen2.5-3B	17, 18, 19, 20, 21, 22, 23, 16, 3
Ours	Qwen2.5-3B	3, 21, 22, 23, 20, 19, 18, 17, 16
ShortGPT	Qwen2.5-7B	16, 17, 15, 14, 12, 25, 11, 13, 24
Shortened	Qwen2.5-7B	14, 16, 17, 12, 13, 15, 18, 11, 8
SLEB	Qwen2.5-7B	14, 15, 16, 17, 13, 12, 18, 11, 9
Ours	Qwen2.5-7B	16, 17, 14, 15, 13, 12, 11, 18, 26
ShortGPT	Qwen2.5-32B	25, 20, 21, 22, 23, 24, 26, 19, 27
Shortened	Qwen2.5-32B	19, 28, 25, 18, 26, 24, 22, 23, 7
SLEB	Qwen2.5-32B	19, 25, 22, 23, 26, 41, 24, 18, 2
Ours	Qwen2.5-32B	20, 21, 24, 25, 23, 28, 26, 22, 2
ShortGPT	LLaMA-2-7B	27, 26, 24, 23, 22, 21, 19, 18, 17
Shortened	LLaMA-2-7B	14, 12, 11, 24, 23, 10, 7, 18, 15
SLEB	LLaMA-2-7B	14, 23, 11, 24, 10, 27, 15, 21, 25
Ours	LLaMA-2-7B	11, 12, 27, 13, 24, 29, 14, 10, 23
ShortGPT	LLaMA-2-13B	33, 31, 30, 29, 28, 27, 26, 25, 24
Shortened	LLaMA-2-13B	31, 28, 29, 27, 30, 33, 32, 25, 26
SLEB	LLaMA-2-13B	31, 29, 10, 28, 11, 33, 19, 12, 35
Ours	LLaMA-2-13B	29, 32, 27, 10, 31, 35, 11, 12, 26
ShortGPT	LLaMA-2-70B	65, 60, 61, 59, 57, 54, 53, 56, 55
Shortened	LLaMA-2-70B	33, 28, 30, 29, 35, 61, 57, 26, 38
SLEB	LLaMA-2-70B	33, 61, 29, 50, 32, 59, 28, 27, 21
Ours	LLaMA-2-70B	61, 65, 63, 60, 28, 30, 27, 29, 26
ShortGPT	LLaMA-3.2-3B	22, 21, 20, 18, 17, 16, 25, 15, 12
Shortened	LLaMA-3.2-3B	8, 10, 9, 23, 12, 22, 17, 11, 13
SLEB	LLaMA-3.2-3B	8, 23, 9, 22, 10, 11, 7, 18, 17
Ours	LLaMA-3.2-3B	8, 10, 17, 22, 9, 21, 11, 7, 18
ShortGPT	LLaMA-3.1-8B	25, 24, 23, 22, 20, 19, 18, 28, 17
Shortened	LLaMA-3.1-8B	14, 16, 17, 12, 13, 15, 18, 11, 8
SLEB	LLaMA-3.1-8B	10, 25, 11, 26, 12, 9, 19, 8, 23
Ours	LLaMA-3.1-8B	16, 17, 14, 15, 13, 12, 11, 18, 26

Table 13: Perplexity results on Wikitext2 dataset. The data in bold represents the best performance at the same sparsity level.

Method	Pruning	Sparsity	Qwen2.5				LLaMA-2			LLaMA-3		Avg.
	Unit		**1.5B**	**3B**	**7B**	**32B**	**7B**	**13B**	**70B**	**3B**	**8B**	
Dense	-	0%	9.26	8.03	6.85	5.02	5.47	4.88	3.37	7.81	6.24	6.33
ShortGPT	T. Block	10%	15.37	10.34	**8.14**	5.72	7.57	5.70	4.22	16.48	10.60	9.35
Shortened	T. Block	10%	12.44	10.32	8.31	**5.68**	7.01	5.67	4.08	11.75	9.82	8.34
SLEB	T. Block	10%	11.06	**9.75**	8.16	5.73	6.95	5.66	4.09	11.28	8.93	**7.96**
Ours	T. Block	10%	**11.05**	10.89	**8.14**	5.70	**6.77**	**5.58**	**4.06**	**11.11**	**8.82**	8.01
ShortGPT	T. Block	30%	41.41	36.61	23.62	12.31	19.47	11.81	7.39	300.78	264.61	79.78
Shortened	T. Block	30%	25.95	17.45	15.96	12.31	25.54	10.51	6.51	61.4	25.34	22.33
SLEB	T. Block	30%	20.95	17.13	14.78	8.61	13.82	8.85	6.15	32.64	21.06	16.00
Ours	T. Block	30%	**20.89**	**16.48**	**14.45**	**8.41**	**13.60**	**8.26**	**6.03**	**29.72**	**18.00**	**15.09**
ShortGPT	T. Block	50%	897.64	494.98	11368.62	83.21	1360.42	142.56	31.15	1646.06	5785.72	2423.37
Shortened	T. Block	50%	88.56	201.34	274.02	554.94	171.42	50.36	13.87	1005.96	853.14	357.07
SLEB	T. Block	50%	72.34	72.24	51.90	23.49	106.20	27.83	11.58	177.32	159.29	78.02
Ours	T. Block	50%	**68.06**	**68.97**	**46.72**	**17.64**	**41.77**	**18.60**	**8.61**	**121.56**	**92.23**	**53.80**