LOSS-ALIGNED STRUCTURED PRUNING FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in large language models (LLMs) have achieved remarkable performance across diverse tasks, yet their increasing size poses significant storage and computational challenges. Model compression, particularly pruning, has emerged as a crucial strategy to reduce memory footprint and computation while preserving predictive performance. In this work, we present LASP, a Loss-Aligned Structured Pruning method that evaluates the contribution of individual model units, such as neurons and attention heads, to the overall performance, subsequently removing those deemed to be of low importance. By combining the activation magnitudes of model units with their gradients with respect to the loss, LASP defines an importance metric that is directly aligned with the model's objective, thereby ensuring the preservation of performance. To mitigate uncertainty caused by the limited calibration dataset used for importance estimation, LASP incorporates the Upper Confidence Bound (UCB) strategy, refining the selection of low-importance units. In implementation, LASP leverages a moving average to maintain running statistics and reduce storage overhead. Empirical results across diverse LLMs and benchmarks demonstrate that LASP outperforms state-of-the-art baselines, effectively balancing efficiency and performance, thus enabling the practical deployment of LLMs.

1 INTRODUCTION

Recent advances in large language models (LLMs) (Touvron et al., 2023a;b; OpenAI, 2023) have demonstrated remarkable capabilities in language understanding, reasoning, and problem-solving. With increasing model size, their performance continues to improve (Kaplan et al., 2020) highlighting the benefits of scaling. However, this rapid growth in parameter count also leads to substantial demands on storage and computational resources. As a result, reducing the memory footprint and computational cost of LLMs has become a central research focus. A variety of approaches have been explored to compress LLMs, such as pruning (Cheng et al., 2023), quantization (Gholami et al., 2021) and knowledge distillation (Goulami et al., 2021).

In this work, we focus on pruning, a widely used model compression technique that removes internal redundancies in neural networks while preserving predictive performance, ideally without requiring costly recovery fine-tuning. Pruning reduces the number of parameters and computational operations, which can significantly lower memory footprint and inference latency, making it particularly important for deploying LLMs in resource-constrained environments.

Existing pruning approaches for LLMs can be broadly categorized into two types: unstructured and structured. Unstructured methods (Sun et al., 2024), such as the one illustrated in Fig. 1 (left), sparsify weight matrices by zeroing individual elements, producing generally sparse matrices; variants like 2:4 or 4:8 sparsity patterns are also employed to leverage GPU sparse acceleration units for faster inference. Structured methods (Ma et al., 2023; Ashkboos et al., 2024), shown in Fig. 1 (middle and right), remove entire rows, columns, or higher-level structures such as attention heads. By pruning these contiguous blocks, structured methods yield models that are naturally compatible with hardware accelerators, enabling reductions in both memory and computation while maintaining alignment with the network architecture.

However, existing pruning techniques face several limitations. Unstructured pruning can yield highly sparse weight matrices, yet it fails to deliver actual storage reduction because the zeroed-out

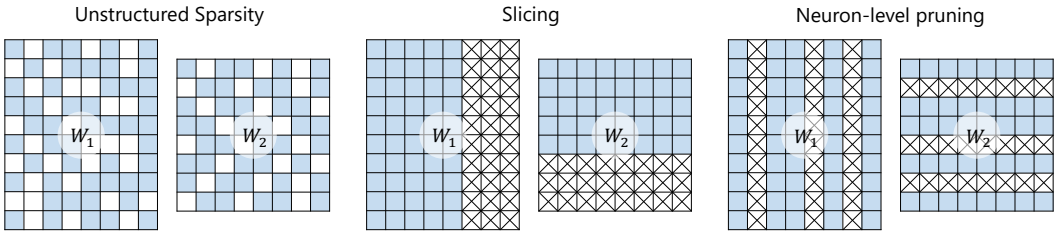

Figure 1: Comparison of representative pruning methods applied to two consecutive weight matrices, W_1 and W_2, within a neural network. White grids indicate pruned parameters. **Unstructured sparsity** removes individual weights, producing irregular sparsity patterns, but does not reduce storage requirements. **Slicing** removes contiguous blocks of weights, partially reducing the matrix dimensions. **Neuron-level pruning** removes entire units, corresponding to full columns in W_1 and full rows in W_2. Both slicing and neuron-level pruning effectively reduce storage requirements.

parameters must still be stored within the original dense tensor format. Given the critical demand for low memory footprint in practical deployment, we steer clear of this approach, and instead focus on structured pruning, which is inherently more hardware-friendly. Nevertheless, current structured pruning methods have their own drawbacks. Many gradient-based approaches, though hardware-compatible, often rely on computationally expensive operations such as estimating second-order Hessian information (Ma et al., 2023). Meanwhile, other prominent methods prune based on weight matrices, typically seeking to reconstruct them via low-rank decomposition (Ashkboos et al., 2024). While effective in practice, these methods primarily target the reconstruction of intermediate weight matrices, lacking an explicit alignment with the model's optimization objective, making it difficult to guarantee performance preservation.

Addressing the aforementioned challenges, we introduce LASP, a first-order loss-aligned structured pruning method. Unlike existing approaches that primarily assess the importance of individual parameters, LASP evaluates the significance of model units, such as neurons and attention heads, by combining their activation magnitudes with gradients with respect to the model loss, thus defining a unit-level pruning metric directly aligned with the optimization objective. Units deemed unimportant are pruned together with all associated parameters, enabling the effective removal of redundancy while preserving model performance. To mitigate uncertainty arising from the limited calibration dataset used to estimate activations and gradients, LASP incorporates the Upper Confidence Bound (UCB) strategy (Auer et al., 2002), ensuring that low-importance units are pruned with high confidence. Moreover, in practical implementation, LASP leverages a simple moving average to maintain running statistics, thereby reducing storage overhead during pruning.

Extensive experiments across diverse LLMs and benchmarks demonstrate the effectiveness of our approach. At a 20% pruning ratio, the pruned models achieve lower loss on the calibration dataset than the original models, indicating that LASP effectively aligns pruning decisions with the optimization objective. Across multiple benchmarks, the pruned models maintain strong performance, preserving 93.5% of the original capability at a 25% pruning ratio and 90% at a 30% pruning ratio, while outperforming the state-of-the-art baseline.

In summary, our contributions are threefold:

- **Loss-Aligned Importance Metric**: LASP introduces an innovative metric that evaluates the importance of model units, allowing for the pruning of parameters associated with low-importance units. It ensures that pruning decisions are aligned with the model's objective.
- **Uncertainty-Aware Pruning**: LASP employs the UCB strategy to handle uncertainty in the limited calibration dataset, ensuring high-confidence selection of low-importance units.
- **Comprehensive Experimental Validation**: We conduct extensive experiments across diverse LLMs, i.e., the Llama (Touvron et al., 2023a), Llama2 (Touvron et al., 2023b), and Vicuna-v1.5 (Zheng et al., 2023) model series, to validate LASP. Our results show that LASP can efficiently compress LLMs while effectively preserving their performance.

2 Related Work

Pruning with Limited Data. A recent line of research, closely related to our approach, focuses on the challenging task of pruning with limited data (Hubara et al., 2021; Frantar et al., 2022; Frantar & Alistarh, 2022; Kwon et al., 2022). These methods are highly desirable because they don't require any changes to the original training process and eliminate the need for computationally expensive retraining on the full dataset. To achieve this, their primary aim is to preserve model performance by leveraging a small amount of data, often called calibration data. They do this by solving a layer-wise reconstruction problem (Hubara et al., 2021) to mitigate the inevitable accuracy drop, which aims to minimize the change in a layer's output with respect to the calibration data. However, a key limitation of existing solvers is their reliance on the computationally heavy calculation of second-order Hessian inverses (Singh & Alistarh, 2020; Frantar et al., 2022), which makes them impractical and difficult to scale to the large hidden state sizes of modern Large Language Models (LLMs). The SparseGPT method (Frantar & Alistarh, 2023) offers a solution by developing a more efficient weight update procedure for LLMs, which uses synchronized second-order Hessian updates to circumvent this computational bottleneck.

Structured Pruning Structured pruning approaches aim to identify and remove less important neurons or components while maintaining model performance. SliceGPT (Ashkboos et al., 2024) leverages PCA on weight matrices to selects the most informative subspaces for pruning and minimize reconstruction error. LLM-Pruner (Ma et al., 2023), on the other hand, evaluates the importance of each parameter using gradient and second-order information, aggregates these scores at the channel or neuron level, and prunes accordingly to enable efficient deployment of large models.

3 Methodology

In this section, we first formulate the problem, then introduce the loss-aligned importance metric and the uncertainty-aware unit selection strategy. Afterwards, we outline the implementation of LASP.

3.1 Problem Formulation

Given a pre-trained LLM f, a target pruning ratio α, and a calibration dataset $\mathcal{D}_{\text{cal}}$ containing sequences x, our goal is to conduct structured pruning by identifying a subset of units $\mathcal{U}_{\text{prune}} \subset \mathcal{U}$ whose removal minimizes the degradation in model performance. Model performance is measured by a negative log-likelihood loss function $\mathcal{L}$, and the pruning objective is formally expressed as:

$$\mathcal{U}^*_{\text{prune}} = \arg\min_{\mathcal{U}_{\text{prune}} \subset \mathcal{U}} \frac{1}{|\mathcal{D}_{\text{cal}}|} \sum_{x \in \mathcal{D}_{\text{cal}}} \Big(\mathcal{L}\big(f_{\text{pruned}}(x)\big) - \mathcal{L}\big(f(x)\big) \Big);$$

where f_{pruned} denotes the model after removing the units in $\mathcal{U}_{\text{prune}}$.

Model pruning can be performed either globally or layer-wise. Global pruning ranks all units across the model and removes the least important ones. Although theoretically effective, it often leads to uneven pruning, with some layers excessively pruned while others remain largely intact. Such imbalance can degrade model performance and complicate hardware deployment, making global pruning less common in practice, especially for large models with highly varying layer sensitivities.

Consequently, in this work we adopt a *layer-wise pruning* strategy, commonly used in prior work (Sun et al., 2024; Ma et al., 2023; Ashkboos et al., 2024), and begin by examining how units within each individual layer contribute to the loss function.

3.2 Loss-Aligned Importance Metric

Consider a pretrained model f with L layers, represented as the composition of layer-wise functions $f_1, f_2, \ldots, f_L$. Once the model parameters are fixed, each f_l becomes a deterministic nonlinear mapping. Consequently, the entire network can be viewed as a fixed composition of such mappings.

Focusing on a particular layer l, the prediction y_{pred} for an input x can be expressed as:

$$y_{\text{pred}} = f(x) = \underbrace{(f_L \circ \cdots \circ f_{l+1})}_{\text{Subsequent sub-network } g_l} \circ \underbrace{(f_l \circ \cdots \circ f_1)(x)}_{\text{Output of layer } l \text{ as } \boldsymbol{h}_l} = g_l(\boldsymbol{h}_l); \tag{1}$$

where $\boldsymbol{h}_l \in \mathbb{R}^d$ is the activation vector of layer l, and g_l encapsulates the computation from layer $l+1$ to the output layer L. Let $\mathcal{U}_l$ denote the set of units in layer l, such as neurons or attention heads, and let $\boldsymbol{z}_u$ represent the contribution of unit $u \in \mathcal{U}_l$ to the layer's activation vector, then the layer activation can be expressed as $\boldsymbol{h}_l = \sum_{u \in \mathcal{U}_l} \boldsymbol{z}_u$.

For a single input x, the training loss can be expressed as a function of $\boldsymbol{h}_l$ as Eq. 2.

$$\mathcal{L}(y_{\text{pred}}) = \mathcal{L}(g_l(\boldsymbol{h}_l)) \equiv \mathcal{L}(\boldsymbol{h}_l) \tag{2}$$

Removing a subset of units $\mathcal{U}_{\text{prune}} \subseteq \mathcal{U}_l$ from layer l is equivalent to perturbing its activation vector by $\Delta \boldsymbol{h}_l = -\sum_{u \in \mathcal{U}_{\text{prune}}} \boldsymbol{z}_u$. Evaluating the impact of this perturbation on the loss of a single sample, we expand $\mathcal{L}(\boldsymbol{h}_l)$ around the original activation using a second-order Taylor series:

$$\Delta\mathcal{L}(x) = \mathcal{L}(\boldsymbol{h}_l + \Delta\boldsymbol{h}_l) - \mathcal{L}(\boldsymbol{h}_l) \approx (\nabla_{\boldsymbol{h}_l}\mathcal{L})^\top \Delta\boldsymbol{h}_l + O\big(\|\Delta\boldsymbol{h}_l\|^2\big); \tag{3}$$

where $\nabla_{\boldsymbol{h}_l}\mathcal{L}$ denotes the gradient of the loss with respect to $\boldsymbol{h}_l$. Although higher-order terms exist, they are intractable to compute and introduce complex cross-unit interactions that make the loss change non-additive. Hence, we restrict our analysis to the first-order approximation as Eq. 4.

$$\Delta\mathcal{L}(x) \approx (\nabla_{\boldsymbol{h}_l}\mathcal{L})^\top \Delta\boldsymbol{h}_l = \sum_{u \in \mathcal{U}_{\text{prune}}} \big(-(\nabla_{\boldsymbol{h}_l}\mathcal{L})^\top \boldsymbol{z}_u\big) \tag{4}$$

For a single unit u, the induced loss change can be written as:

$$\Delta\mathcal{L}_u(x) \approx -(\nabla_{\boldsymbol{h}_l}\mathcal{L})^\top \boldsymbol{z}_u = -\frac{\partial\mathcal{L}}{\partial\boldsymbol{z}_u} \cdot \boldsymbol{z}_u; \tag{5}$$

where $\Delta\mathcal{L}_u(x)$ represents the loss change caused by removing a single unit u and $\frac{\partial\mathcal{L}}{\partial\boldsymbol{z}_u}$ is the gradient of the loss with respect to the unit's output vector. Eq. 5 highlights that the importance of a unit depends jointly on its output magnitude and the sensitivity of the loss to that output.

By aggregating $\Delta\mathcal{L}_u(x)$ for unit u over samples in the calibration set, we obtain the expected change in loss upon removing u, which defines the unit's importance metric for guiding pruning decisions, as formalized in Eq. 9.

$$\mu_u = \frac{1}{|\mathcal{D}_{\text{cal}}|} \sum_{x \in \mathcal{D}_{\text{cal}}} \Delta\mathcal{L}_u(x) \tag{6}$$

3.3 Uncertainty-Aware Pruning

In practice, the loss change $\Delta\mathcal{L}_u(x)$ can exhibit highly skewed or heavy-tailed behavior, as detailed in Appendix D. In such cases, the empirical mean μ_u may underestimate the importance of the individual unit u. For instance, μ_u can be close to zero even when u makes substantial contributions under rare but critical inputs. Moreover, the calibration dataset $\mathcal{D}_{\text{cal}}$ is typically limited in size, e.g., only hundreds of sequences, further limiting the confidence in the estimated importance.

Considering these issues, we adopt the UCB strategy (Auer et al., 2002). UCB balances *exploitation* of high-reward options with *exploration* of uncertain ones by augmenting the mean estimate with an uncertainty-dependent bonus. A natural correspondence exists between this and the pruning problem: each unit can be treated as an option, where the expected loss reduction μ_u corresponds to its average reward, and the standard deviation of $\Delta\mathcal{L}_u(x)$ over inputs captures the uncertainty in that estimated reward.

Consequently, the UCB enhanced importance score for neuron u on sequence x is then given by:

$$s_u(x) \;=\; \mu_u(x) + \alpha \cdot \sigma_u(x); \tag{7}$$

where σ_u is the standard deviation of $\Delta\mathcal{L}_u$ over the calibration dataset, and $\alpha \geq 0$ is a tunable parameter that controls the emphasis on uncertainty. A larger α leads to a more conservative criterion, retaining neurons that may have low average contribution but high variability within the sequence, thereby ensuring high confidence when identifying low-importance units.

3.4 Implementation Workflow

LASP proceeds sequentially over layers $1, 2, \ldots, l, \ldots, L$. For each layer l, the pruning process involves three main steps: selecting the pruning granularity, computing the importance scores, and removing the units chosen for pruning.

Pruning Granularity Selection. For MLP layers, a unit is an individual neuron. For attention layers, the granularity can be more varied, ranging from individual weights within the projection matrices, specific dimensions of queries, keys, or values, to entire attention heads. In this work, for both efficiency and simplicity, we define a unit as an entire attention head, which entails pruning the complete set of query, key, and value projections associated with it.

Importance Score Computation. In practice, the loss for each sequence can be decomposed into token-level contributions, allowing a hierarchical formulation of unit importance. For a unit u, we compute its mean and standard deviation on a single sequence x as:

$$\begin{aligned}
\mu_u(x) &= \frac{1}{|x|}\sum_{i=1}^{|x|}\Big(-\frac{\partial\mathcal{L}(x_i)}{\partial \boldsymbol{z}_u(x_i)}\cdot \boldsymbol{z}_u(x_i)\Big) \\
\sigma_u(x) &= \sqrt{\frac{1}{|x|}\sum_{i=1}^{|x|}(-\frac{\partial\mathcal{L}(x_i)}{\partial \boldsymbol{z}_u(x_i)}\cdot \boldsymbol{z}_u(x_i) - \mu_u(x))^2}
\end{aligned} \tag{8}$$

The overall importance score S_u of unit u over the calibration dataset $\mathcal{D}_{\text{cal}}$ is then obtained by averaging sequence-level scores over all sequences:

$$S_u = \frac{1}{|\mathcal{D}_{\text{cal}}|}\sum_{x\in\mathcal{D}_{\text{cal}}} s_u(x) \tag{9}$$

This computation can be efficiently implemented using a moving average over the dataset, yielding stable estimates of both the mean and variance for each unit. These estimates are subsequently used to rank units within each layer and guide the pruning process.

Pruned Unit Selection. For both MLP and attention layers, units are ranked according to their importance scores within each layer. Based on the desired pruning ratio, the units with the lowest scores are selected and removed. Pruning an MLP neuron corresponds to removing the entire associated row or column from the weight matrix, while pruning an attention unit corresponds to removing the entire attention head.

Algorithm 1 summarize the pruning procedure. As we can see, the above three steps are applied independently to each layer, ensuring that pruning respects the model's structure and avoids excessive degradation concentrated in a few critical layers.

4 Experiments

4.1 Experimental Settings

Foundation Large Language Model. To showcase the effectiveness and versatility of our method, we test it over three open-source large language model families that are widely used, includ-

Algorithm 1: Loss-Aligned Structured Pruning

Input: Pretrained model f, calibration set $\mathcal{D}_{\text{cal}}$, pruning ratio r, coefficient α
Output: Pruned model f_{pruned}
for *each target layer* l *in* f **do**
  Initialize score $S_u = 0$ for each unit $u \in \mathcal{U}_l$;
  Initialize counter $n = 0$;
  for *each sequence* $x \in \mathcal{D}_{\text{cal}}$ **do**
    Forward x through f and compute activations at layer l;
    Backward on f to obtain the activation gradients of layer l;
    Increment counter: $n \leftarrow n + 1$;
    for *each unit* $u \in \mathcal{U}_l$ **do**
      Calculate sequence-level mean and standard deviation on $\Delta\mathcal{L}_u(x)$ to get $s_u(x)$;
      Update score using running average: $S_u \leftarrow S_u + \frac{1}{n}(s_u(x) - S_u)$;
  Rank units according to $S_u, u \in \mathcal{U}_l$ and prune the lowest $\lfloor r \times |\mathcal{U}_l| \rfloor$ units;

ing LLaMA model family (Touvron et al., 2023a), Vicuna-v1.5 model family (Zheng et al., 2023), and LLaMA2 model family (Touvron et al., 2023b).

Evaluation Datasets. To assess the model performance in the task-agnostic setting, we perform a zero-shot task on common sense reasoning datasets: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-easy and ARC-challenge (Clark et al., 2018). All these tasks are evaluated with the lm-eval-harness library (Gao et al., 2021). Additionally, we complement our evaluation with a perplexity (PPL) analysis on WikiText2 validation set.

Calibration Dataset. Follow the settings from SliceGPT, our calibration dataset contains 128 sequences of length 2048 sampled from the first shard of the WikiText-2 training set (Merity et al., 2016). To ensure consistency, we utilize the same calibration data for all pruning algorithms.

Baseline Setup. We compare our method with two variants of SliceGPT (Ashkboos et al., 2024): the first is the original SliceGPT, which applies PCA-based structured pruning on the weight matrices, representing a typical structured pruning approach; the second is a version of SliceGPT fine-tuned with LoRA (Hu et al., 2022) on 4,000 sequences from the Alpaca training set (Taori et al., 2023), each with a length of 1,024 tokens. To eliminate potential external factors—such as varying pruning ratios across layers with different sensitivities—all methods adopt a uniform pruning strategy, ensuring a fair and consistent comparison.

Table 1: LLaMA-1, LLaMA-2, and Vicuna perplexity results on WikiText2 test set. The calibration set size and sequence length are 128 and 2048, respectively.

Ratio	Method	LLaMA-1				LLaMA-2		Vicuna-v1.5	
		7B	13B	30B	65B	7B	13B	7B	13B
-	Dense	5.47	4.88	4.10	3.53	5.47	4.88	6.78	5.95
20%	SliceGPT (w/o FT)	7.00	6.13	5.27	4.59	6.86	6.04	8.13	7.84
	LASP	6.18	5.38	4.54	3.98	6.16	5.38	6.86	5.91
25%	SliceGPT (w/o FT)	7.67	6.64	5.70	4.99	7.56	6.61	8.84	8.99
	LASP	6.75	5.72	4.83	4.23	6.73	5.72	7.46	6.95
30%	SliceGPT (w/o FT)	8.70	7.35	6.32	5.49	8.64	7.44	9.94	11.33
	LASP	7.28	6.18	5.14	4.54	7.29	6.16	7.98	6.86

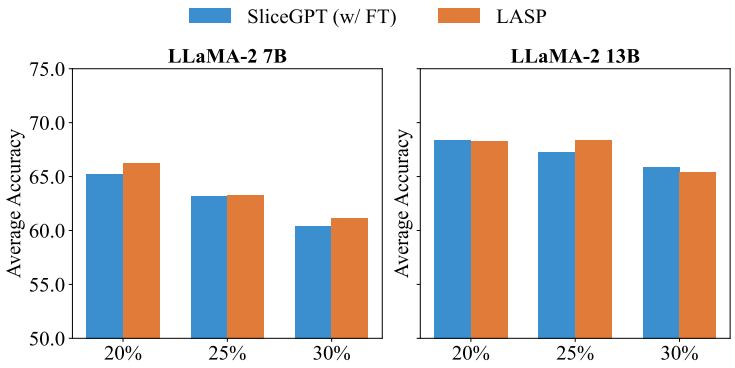

Figure 2: Average accuracy comparison of SliceGPT (w/ FT) and our method across different pruning ratios (20%, 25%, 30%) on LLaMA-2 7B and 13B models.

4.2 LANGUAGE MODELING

In this subsection, we present results on WikiText-2 using LLaMA-1, LLaMA-2, and Vicuna-v1.5 models. Table 1 reports the perplexity under different pruning ratios, i.e., 20%, 25%, 30%. LASP consistently achieves lower perplexity than SliceGPT at the same pruning ratio across all models. These results demonstrate that LASP maintains higher accuracy under structured pruning across different model families.

4.3 ZERO-SHOT PERFORMANCE

Table 2: Evaluation results on multiple benchmarks.

Model	Ratio	Method	ARC-c	ARC-e	BoolQ	HellaSwag	PIQA	Winogrande	Average
LLaMA-2 7B	-	Dense	43.34	76.39	77.74	75.98	78.07	69.22	70.12
	20%	SliceGPT (w/o FT)	32.84	60.81	48.77	58.98	69.31	64.48	55.86
		LASP	40.70	73.32	70.15	71.36	76.93	65.35	66.30
	25%	SliceGPT (w/o FT)	32.25	61.23	51.52	54.31	66.21	62.90	54.73
		LASP	39.08	70.45	67.40	66.62	75.30	61.09	63.32
	30%	SliceGPT (w/o FT)	29.01	55.93	38.59	49.10	63.32	62.66	49.76
		LASP	36.18	69.78	64.34	63.90	73.45	59.35	61.16
LLaMA-2 13B	-	Dense	48.29	79.42	80.58	79.37	79.16	72.14	73.16
	20%	SliceGPT (w/o FT)	38.65	71.25	44.95	62.79	70.78	67.56	59.33
		LASP	41.72	73.74	70.34	75.95	77.58	68.67	67.93
	25%	SliceGPT (w/o FT)	35.83	65.69	40.88	57.39	68.00	68.19	56.00
		LASP	41.72	74.62	75.96	74.22	76.82	66.85	68.37
	30%	SliceGPT (w/o FT)	32.50	59.42	38.74	52.16	64.47	65.58	52.15
		LASP	38.74	72.69	72.26	70.52	74.76	63.61	65.43

Table 2 presents the results of LLaMA-2 7B and 13B models on zero-shot tasks under different pruning ratios, i.e., 20%, 25%, 30%. Across all tasks and pruning ratios, LASP consistently outperforms SliceGPT. While the performance of both methods improves with model size, LASP maintains a clear advantage. Notably, the 7B model pruned by 25% and the 13B model pruned by 30% still retain around 90% of the original dense model's average accuracy. Moreover, for both model sizes, LASP at 30% pruning ratio surpasses SliceGPT at 20%, highlighting its ability to preserve model quality even under aggressive pruning.

Furthermore, Figure 2 presents a direct comparison between our method and the fine-tuned SliceGPT. For the LLaMA-2 7B model, our method consistently outperforms SliceGPT across all pruning ratios. For the LLaMA-2 13B model, it performs slightly worse than the fine-tuned

SliceGPT at 20% and 30% pruning, but surpasses it at 25% pruning. Overall, across both model scales, our method is comparable to or slightly better than the fine-tuned SliceGPT.

4.4 Ablation Study

In this subsection, we conduct an ablation study on the hyperparameter α. As shown in Figure 3, the perplexity curves for both LLaMA-2 7B and 13B models exhibit a clear U-shape, confirming that an optimal balance is essential for performance.

When α is small, e.g., $\alpha = 0.01$, unit selection relies almost entirely on the average of the first-order loss change approximation. Although this captures the unit's mean contribution, it tends to favor neurons whose true impact is uncertain, resulting in higher perplexity. As α increases, the standard deviation term is gradually incorporated, reflecting the confidence in each neuron's loss contribution and improving performance, which peaks when this confidence is appropriately weighted for optimal neuron selection. Beyond this point, further increasing α can degrade performance due to the numerical scale: as shown in Appendix D, the standard deviation is much larger than the average loss, so when α becomes sufficiently large (e.g., 0.5), the importance score is dominated by the variance and the mean contribution is effectively ignored, which harms fluency and increases perplexity. These observations highlight that a carefully chosen α is critical for balancing average loss contribution and confidence. Ideally, α should be set such that the mean and standard deviation terms can cooperate effectively, with their scales roughly comparable, so that neither term completely dominates the importance score and both contribute meaningfully.

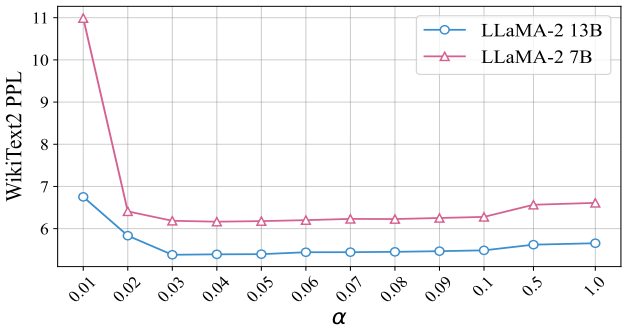

Figure 3: Ablation study of the hyperparameter α on WikiText-2 perplexity. The figure illustrates how performance varies with different values of α for both LLaMA-2 7B and 13B models, highlighting the importance of balancing the first-order loss approximation with the confidence in neuron contributions.

4.5 More Analysis

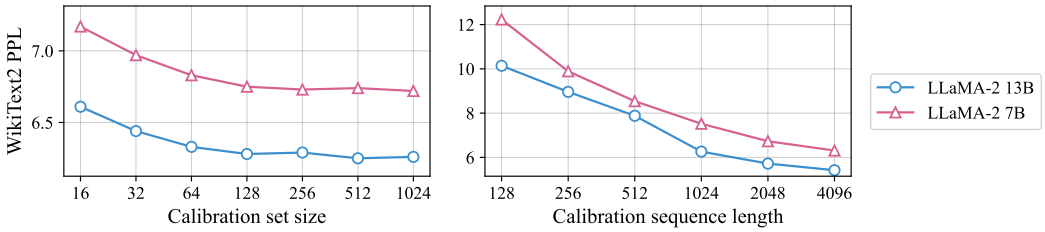

Figure 4: The effect of the calibration set size and sequence length on perplexity of WikiText2.

Data sensitivity. We investigate the effect of calibration data size and sequence length on pruning performance, as shown in Figure 4. When fixing the sequence length to 2048, increasing the calibration set size generally reduces perplexity. This improvement can be attributed to the more accurate estimation of both the mean of the first-order loss and the standard deviation, as observing more samples allows these statistics to better reflect the true distribution, thereby enhancing performance; as the number of samples grows, the PPL curve appears to converge, suggesting that these statistics have become sufficiently reliable. When fixing the number of samples to 128 and varying sequence length, perplexity also decreases. This improvement is not only because longer contexts naturally benefit language modeling, but also because longer sequences effectively provide more token-level

observations within each sample, making the sample-level importance estimation more precise and thus further reducing perplexity. Overall, both trends indicate that richer calibration data can lead to more stable pruning decisions and lower perplexity.

Pruning dynamics. To better understand how our method works on each layer, we analyze the pruning dynamics of LLaMA2-7B under different pruning ratios.

Figure 5 illustrates the layer-wise impact of varying pruning ratios on the model's average loss. A key observation is that the effect of pruning varies significantly across the network's depth. For the early layers (approximately layers 1–5), pruning consistently reduces the loss, indicating a high degree of parameter redundancy. In these layers, a pruning ratio of 0.20 achieves a noticeable reduction in loss, suggesting that removing redundant neurons can act as a beneficial regularizer and improve generalization. In contrast, the middle layers (approximately layers 8–18) exhibit minimal redundancy and are strongly correlated with the model's predictive performance. Even moderate pruning in this region leads to an increase in loss, highlighting these layers as a critical bottleneck. For the later layers (approximately layers 18–32), some redundancy is again present, allowing moderate pruning without severely affecting performance. Overall, the redundancy profile across the network follows a clear pattern: early layers are most redundant, middle layers are the least, and later layers are intermediate.

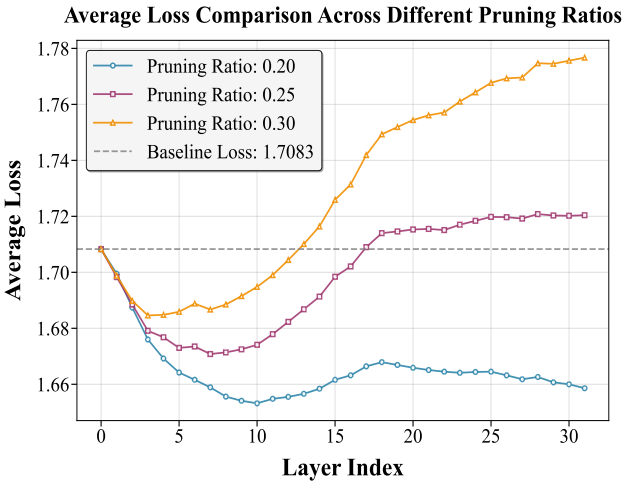

Figure 5: Pruning dynamics of LLaMA-7B under different pruning ratios. The x-axis denotes the layer index (0–31), where each index corresponds to the average loss measured before pruning the respective layer. The y-axis shows the average loss value. Curves with different colors represent different pruning ratios.

Another key observation relates to the redundancy of information processed by the model. While pruning the first few layers produces similar loss values across different pruning ratios, continued pruning reveals a nuanced pattern. For example, applying a 0.20 pruning ratio to up to 10 layers achieves lower loss than pruning only five or six layers at higher ratios (0.25 or 0.30), suggesting that the model's information processing has an optimal capacity. Exceeding this capacity through over-pruning can diminish the effective information available to subsequent layers, limiting further pruning benefits and potentially degrading performance. This highlights the critical balance between structural redundancy and information capacity, indicating that effective pruning must account not only for neuron count but also for the sufficiency of information propagation throughout the network.

5 Conclusion

In this work, we have proposed LASP, a loss-aligned structured pruning method for LLMs that directly evaluates the contribution of model units to overall performance. By integrating activation magnitudes with gradients and leveraging the UCB strategy, LASP effectively identifies and removes low-importance units while mitigating uncertainty from limited calibration data. Our implementation further reduces storage overhead through running statistics. Extensive experiments across multiple LLMs and benchmarks demonstrate that LASP consistently outperforms state-of-the-art pruning methods, achieving a favorable trade-off between efficiency and model performance. These results highlight the potential of LASP for enabling the practical deployment of LLMs without significant loss in predictive capability. Beyond empirical effectiveness, our approach provides a new perspective by modeling the loss directly at the unit-output level. We hope that this formulation will inspire further theoretical insights into LLM pruning.

REFERENCES

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=vXxardq6db.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations. *arXiv preprint arXiv:2308.06767*, 2023.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions, 2019. URL https://arxiv.org/abs/1905.10044.

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

Elias Frantar and Dan Alistarh. Spdy: Accurate pruning with speedup guarantees. In *International Conference on Machine Learning (ICML)*, 2022.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning, PMLR 202:10323-10337, 2023.*, 2023. URL https://openreview.net/forum?id=vXxardq6db.

Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, and et al. A framework for few-shot language model evaluation, version v0.0.1, September 2021.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *CoRR*, abs/2103.13630, 2021.

Jianping Goulami, Baosheng Yu, Stephen John Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Itay Hubara, Brian Chmiel, Moshe Island, Ron Banner, Seffi Naor, and Daniel Soudry. Accelerated sparse neural training: A provable and efficient method to find n:m transposable masks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. URL https://arXiv.org/abs/2001.08361.

Woosuk Kwon, Sehoon Kim, Michael W. Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. A fast post-training pruning framework for transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2305.11627.

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

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. URL https://arxiv.org/abs/2303.08774.

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

Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=PxoFut3dWW.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a. URL https://arxiv.org/abs/2302.13971.

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, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b. URL https://arxiv.org/abs/2307.09288.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

A LLM USAGE

In this work, large language models are mainly used as research assistants to support literature exploration and text refinement. Specifically, we leverage LLMs to (i) improve the clarity and conciseness of our writing, and (ii) assist in locating relevant conference papers on pruning methods, including both unstructured and structured approaches, and locating key survey articles concerning the broader topics of model compression, pruning, and quantization.

B IMPLEMENTATION OF USING SELF-SUPERVISED LOSS

B.1 THEORY ANALYSIS

We aim to evaluate the change in loss $\Delta\mathcal{L}$ caused by pruning a unit across the entire calibration dataset $\mathcal{D}_{\text{cal}}$. Formally, this is defined as the average change over all sequences in $\mathcal{D}_{\text{cal}}$:

$$\Delta\mathcal{L}_u = \frac{1}{|\mathcal{D}_{\text{cal}}|} \sum_{x \in \mathcal{D}_{\text{cal}}} \Delta\mathcal{L}_u(x). \tag{10}$$

To understand this average, we first analyze the loss on a single sequence x. In maximum likelihood training, the loss on a sequence $\mathbf{x} = (x_1, x_2, \ldots, x_{|x|})$ is defined as the negative log-likelihood, which can be expressed as the average of token-level losses:

$$\mathcal{L}(x) = -\frac{1}{|x|} \sum_{i=1}^{|x|} \log p(x_i \mid x_{<i}), \tag{11}$$

where $|x|$ is the length of the sequence.

Accordingly, the loss change on x caused by pruning a unit is

$$\Delta\mathcal{L}_u(x) = \frac{1}{|x|} \sum_{i=1}^{|x|} \Delta\big(-\log p(x_i \mid x_{<i})\big). \tag{12}$$

$$\approx \frac{1}{|x|} \sum_{i=1}^{|x|} \left(-\frac{\partial \log p(x_i \mid x_{<i})}{\partial \boldsymbol{z}_u(x_i)} \cdot \boldsymbol{z}_u(x_i) \right), \tag{13}$$

To account for the variability in unit contributions, we apply the UCB algorithm to obtain a more robust importance metric. Specifically, for unit u the importance score within a sequence is defined as

$$s_u(x) = \mu_u(x) + \alpha \cdot \sigma_u(x), \tag{14}$$

where $\mu_u(x)$ and $\sigma_u(x)$ are defined as the mean and standard deviation of token-level first-order loss changes within sequence x, and α is a hyperparameter controlling the trade-off in UCB:

$$\mu_u(x) \approx \frac{1}{|x|} \sum_{i=1}^{|x|} \left(-\frac{\partial \log p(x_i \mid x_{<i})}{\partial \boldsymbol{z}_u(x_i)} \cdot \boldsymbol{z}_u(x_i) \right), \tag{15}$$

$$\sigma_u(x) \approx \sqrt{\frac{1}{|x|} \sum_{i=1}^{|x|} \left(\left(-\frac{\partial \log p(x_i \mid x_{<i})}{\partial \boldsymbol{z}_u(x_i)} \cdot \boldsymbol{z}_u(x_i) \right) - \mu_u(x) \right)^2}. \tag{16}$$

By aggregating over all sequences in the calibration dataset $\mathcal{D}_{\text{cal}}$, we obtain the overall importance score S_u, which is:

$$S_u = \frac{1}{|\mathcal{D}_{\text{cal}}|} \sum_{x \in \mathcal{D}_{\text{cal}}} s_u(x). \tag{17}$$

For each layer, after obtaining the overall scores $\{S_u\}$, we sort all units in ascending order according to S_u, and then prune the lowest-ranked units according to the target pruning ratio r.

Table 3: coefficient α settings for different models under various pruning ratios.

Model	20%	25%	30%
llama2-7b	0.03	0.03	0.03
llama2-13b	0.03	0.03	0.04
llama-7b	0.03	0.05	0.03
llama-13b	0.03	0.03	0.03
llama-30b	0.07	0.07	0.07
llama-65b	0.12	0.10	0.12
vicuna-v1.5-7b	0.03	0.03	0.03
vicuna-v1.5-13b	0.03	0.0117	0.0175

B.2 Implementation Details

In practice, for the mean term in the significance score, we directly take the empirical average over the calibration dataset. For the σ term, since it is simultaneously affected by both the sample length (which is fixed across all pruning samples) and the hyperparameter α, we merge these two factors into a single adjustable coefficient α for simplicity. This treatment keeps the implementation convenient while retaining the flexibility of controlling the variance penalty. For our approach, the best coefficient α under different pruning ratios can be seen in Table 3.

Moreover, for finetuning the SliceGPT-pruned LLaMA-2 model, we employ LoRA for efficient adaptation. Specifically, we set the training batch size to 3, with LoRA-$\alpha = 10$, rank $r = 32$, and a dropout rate of 0.05. LoRA modules are injected into both attention heads and MLP layers, enabling parameter-efficient fine-tuning while maintaining the performance of the pruned model.

C Efficiency analysis.

Table 4 reports the efficiency statistics of the pruned LLaMA2-13B models under different pruning ratios (PR), including the number of multiply–accumulate operations (MACs), runtime peak memory consumption, inference latency, and model load memory. The reported MACs correspond to the prefill stage, while memory and latency are measured during the generation of 1024 tokens. As the pruning ratio increases, both the computational requirements and memory footprint decrease consistently. For example, at a pruning ratio of 20%, MACs are reduced from 822.64G to 660.25G, runtime memory drops from 25.7 GiB to 20.7 GiB, and inference latency improves from 32.75s to 27.73s. At 25% pruning, further efficiency gains are observed, with memory usage reduced to 19.5 GiB and latency to 26.10s. When the pruning ratio reaches 30%, latency remains stable at 25.89s, while both MACs and memory continue to decline, with runtime memory reduced to 18.2 GiB. These results show that structured pruning substantially reduces computation and memory costs while delivering more efficient inference without introducing instability in runtime behavior. (All results are measured on the WikiText2 test set using a single NVIDIA RTX 4090 48G GPU.)

Table 4: Efficiency statistics of the pruned models under different pruning ratios (PR).

PR (%)	MACs (G)	Runtime Memory (MiB)	Latency (s)	Model Load Memory (MiB)
0%	822.644	25748.890	32.753	24826.792
20%	660.254	20726.856	27.727	19994.854
25%	619.618	19470.430	26.100	18783.917
30%	579.020	18215.437	25.891	17574.151

D First-order loss value visualization

As Figure 6 and Figure 7 show, the first-order loss approximation exhibits a heavy-tailed distribution: for more than 50% of the tokens within a sequence, the value is close to zero, while in a few cases a neuron plays a critical role and yields a large loss change.

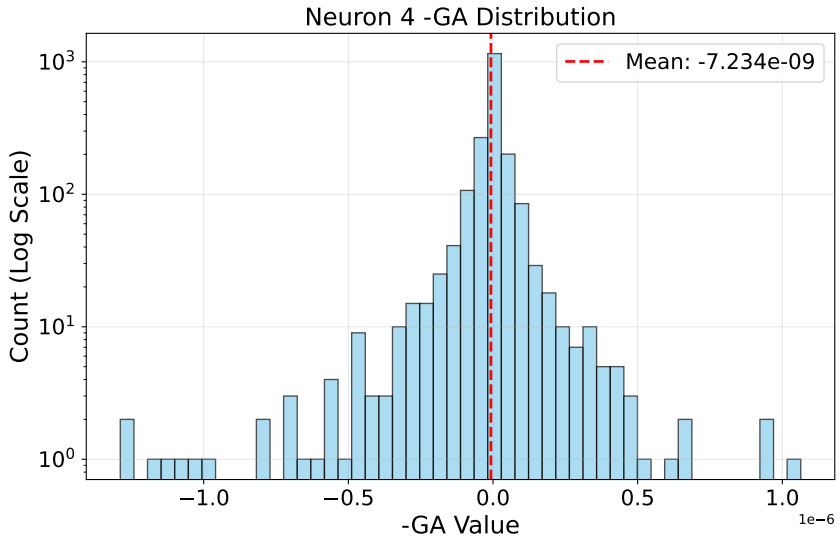

Figure 6: Distribution of $-\mathrm{GA}$ values for the 4-th neuron in the first MLP layer. Most values concentrate near zero, while a few significant deviations highlight the necessity of incorporating uncertainty into the pruning criterion.

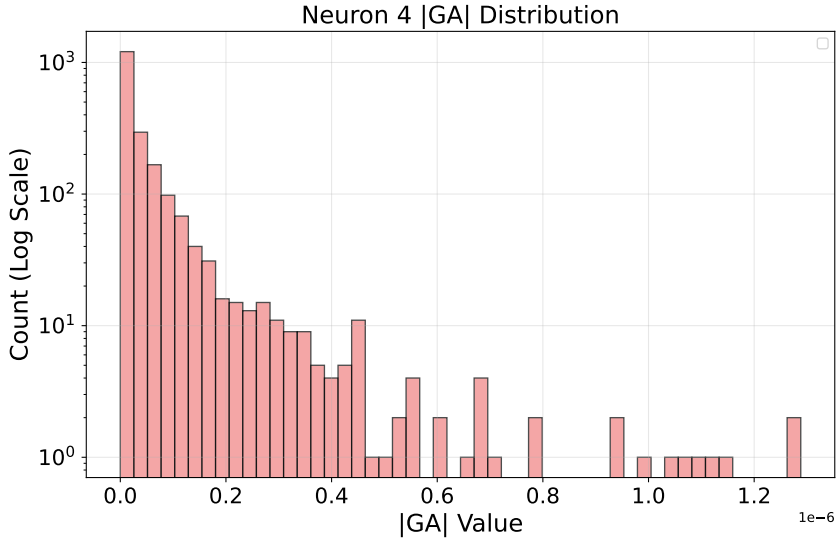

Figure 7: Distribution of $|GA|$ values for the 4-th neuron in the first MLP layer. The distribution in the figure clearly shows a long-tail distribution.

Therefore, relying solely on the empirical mean of $\Delta\mathcal{L}_u(\mathbf{x})$ observed on a calibration set $\mathcal{D}_{\text{cal}}$ to estimate a unit's importance is therefore risky. On the one hand, the limited size of $\mathcal{D}_{\text{cal}}$ introduces uncertainty into the estimation; on the other hand, the heavy-tailed nature of $\Delta\mathcal{L}_u$ means that the mean is often dominated by near-zero values, overlooking the substantial influence a neuron may exert on a small subset of critical tokens. To address this, we introduce UCB, which explicitly accounts for rare but significant variations and improves confidence in assessing a unit's influence, thereby enabling more accurate selection.