

Beyond Outliers: A Data-Free Layer-wise Mixed-Precision Quantization Approach Driven by Numerical and Structural Dual-Sensitivity

Hengyuan Zhang¹, Xinrong Chen², Zunhai Su¹, Xiao Liang³, Jing Xiong¹,
Wendong Xu¹, He Xiao¹, Chaofan Tao¹, Wei Zhang⁴, Ruobing Xie⁴,
Lei Jiang⁴, Hayden Kwok-Hay So¹, Ngai Wong¹†

¹The University of Hong Kong ²Peking University

³University of California, Los Angeles ⁴Tencent

hengyuan.zhang88@gmail.com

Abstract

Layer-wise mixed-precision quantization (LMPQ) enables effective compression under extreme low-bit settings by allocating higher precision to sensitive layers. However, existing methods typically treat all intra-layer weight modules uniformly and rely on a single numerical property when estimating sensitivity, overlooking their distinct operational roles and structural characteristics. To address this, we propose NSDS, a novel calibration-free LMPQ framework driven by Numerical and Structural Dual-Sensitivity. Specifically, it first mechanistically decomposes each layer into distinct operational roles and quantifies their sensitivity from both numerical and structural perspectives. These dual-aspect scores are then aggregated into a unified layer-wise metric through a robust aggregation scheme based on MAD-Sigmoid and Soft-OR to guide bit allocation. Extensive experiments demonstrate that NSDS consistently achieves superior performance compared to various baselines across diverse models and downstream tasks, without relying on any calibration data. The code is available at [the provided link](#).

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable performances across a wide range of applications, including complex reasoning (Li et al., 2025; Liang et al., 2025; Yu et al., 2025; Liang et al., 2026; Yang et al., 2026), multilingual comprehension (Qin et al., 2025; Zhang et al., 2025c; Bu et al., 2025), and various specialized domains (Zhang et al., 2024; Xiao et al., 2025; Zhang et al., 2025b,d; Chang et al., 2025; Xiong et al., 2026; Chen et al., 2026). However, their massive scale requires substantial computational and memory resources for deployment. Post-Training Quantization (PTQ) (Frantar et al.,

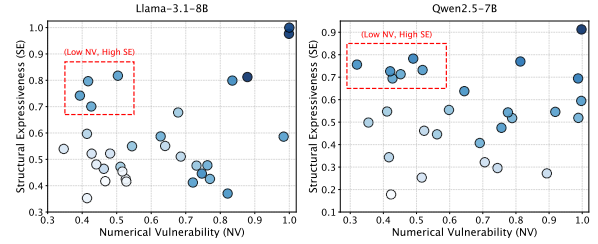


Figure 1: Layer-wise sensitivity across two LLMs. Each dot represents a layer. Darker dots indicate more severe perplexity degradation (ΔPPL) when quantizing that layer. See §2.2 for the calculation of Numerical Vulnerability and Structural Expressiveness. Red dashed boxes highlight that layers with low numerical but high structural sensitivity still suffer significant degradation. See Appendix A for more results.

2023) has emerged as a dominant compression technique, reducing memory footprint and inference cost while preserving model capabilities without the need for resource-intensive retraining.

Standard PTQ typically applies uniform precision across all layers, which often triggers severe performance degradation under extreme low-bit regimes (e.g., < 4 bits) (Li et al., 2023). Layer-wise Mixed-Precision Quantization (LMPQ) alleviates this issue by strategically allocating higher precision to crucial layers and fewer bits elsewhere. Since bit-width remains uniform within each layer, LMPQ preserves tensor contiguity and kernel regularity, making it more hardware-friendly than other mixed-precision schemes (Xiao et al., 2026). Existing LMPQ methods are generally either search-based or criterion-based. Since search-based optimization is computationally prohibitive and difficult to scale (Wang et al., 2019; Yu et al., 2020; Lee et al., 2025), criterion-based methods that efficiently estimate layer sensitivity via proxy metrics have become the preferred paradigm. Depending on whether calibration data is required, they can be further divided into calibration-based and calibration-free methods. Calibration-based methods may overfit to the calibration set and re-

† Corresponding author.

quire heavy forward and backward passes, which becomes increasingly impractical for large-scale LLMs (Williams and Aletras, 2024; Landge et al., 2025). Consequently, calibration-free methods that directly estimate sensitivity from model weights offer a more robust, efficient, and universally applicable solution for LMPQ.

Despite their efficiency, existing calibration-free LMPQ methods exhibit two major limitations: **1)** they typically treat all weight modules within a layer uniformly when estimating sensitivity, ignoring that different modules play distinct operational roles; **2)** they rely on a single numerical property, such as outlier statistics or parameter distribution, and do not consider the structural expressiveness of the weights. As shown in Fig. 1, a layer with minimal numerical outliers can still suffer a significant performance drop after quantization if its structural expressiveness is compromised.

To address these limitations, we propose a calibration-free LMPQ approach driven by Numerical and Structural Dual-Sensitivity (NSDS) to comprehensively estimate layer sensitivity. Specifically, we first mechanistically decompose a model layer into distinct operational components, categorizing them into *Detectors* and *Writers*. We then quantify the sensitivity of each component from two perspectives: Numerical Vulnerability (NV) and Structural Expressiveness (SE). Finally, we aggregate these dual-aspect scores into a unified layer-wise metric using robust MAD-Sigmoid normalization and Soft-OR operations, and perform bit allocation based on the resulting ranking. Extensive experiments on various LLMs and benchmarks demonstrate the effectiveness of our NSDS framework.

To summarize, our contributions are as follows:

1) We propose NSDS, a calibration-free LMPQ framework that estimates sensitivity from numerical and structural perspectives and integrates them through a robust aggregation scheme based on MAD-Sigmoid and Soft-OR, yielding a comprehensive criterion for bit allocation.

2) NSDS is the first to integrate the operational roles of weight components into quantization sensitivity estimation. We mechanistically decompose layers into operational components and design role-aware structural metrics for each component.

3) Extensive experiments validate the superiority of our NSDS framework. Without relying on any calibration data, NSDS consistently achieves competitive or superior performance compared to

various baselines across different LLM families and downstream tasks. We also conduct further analysis to offer valuable insights for future research.

2 The NSDS Framework

In this section, we introduce the NSDS framework, which comprehensively estimates layer sensitivity without requiring any calibration data. We first describe how a layer is mechanistically decomposed into distinct operational components (§2.1). Next, we present the metric to estimate the sensitivity from Numerical Vulnerability and Structural Expressiveness (§2.2). Finally, we aggregate these scores into a unified layer-wise metric via MAD-Sigmoid and Soft-OR, and perform bit allocation based on the resulting ranking (§2.3). We present the overview of NSDS framework in Fig 2.

2.1 Mechanistic View of the Layer

Previous methods typically treat all weight modules within a layer uniformly when estimating quantization sensitivity, overlooking the functional heterogeneity inherent in different components of the layer. To address this, we decompose the layer based on mechanistic interpretability principles (Elhage et al., 2021; Ferrando et al., 2024; Duan et al., 2026; Zhang et al., 2026). Typically, a layer consists of a Multi-Head Attention followed by a Feed-Forward Network.

Multi-Head Attention (MHA). Given an input $X^{(l-1)} \in \mathbb{R}^{N \times d_{\text{model}}}$ from layer $l - 1$, the MHA operation for head h in layer l is defined as:

$$\text{Attn}^{(l,h)}(X^{(l-1)}) = \text{softmax} \left(\frac{(X^{(l-1)} W_Q^{(l,h)})(X^{(l-1)} W_K^{(l,h)})^T}{\sqrt{d_{\text{head}}}} \right) \cdot X^{(l-1)} W_V^{(l,h)} W_O^{(l,h)}. \quad (1)$$

where $W_Q^{(l,h)}$, $W_K^{(l,h)}$, $W_V^{(l,h)}$, $W_O^{(l,h)}$ denote the query, key, value, and output projections for head h in layer l .¹ Following Elhage et al. (2021), Eq. 1 can be rewritten to expose two distinct components:

$$\begin{aligned} \text{Attn}^{(l,h)}(X^{(l-1)}) &= \text{softmax} \left(\frac{X^{(l-1)} \cdot W_Q^{(l,h)} W_K^{(l,h)T} \cdot X^{(l-1)T}}{\sqrt{d_{\text{head}}}} \right) \\ &\quad \cdot X^{(l-1)} \cdot W_V^{(l,h)} W_O^{(l,h)} \\ &= \text{softmax} \left(\frac{X^{(l-1)} W_{QK}^{(l,h)} X^{(l-1)T}}{\sqrt{d_{\text{head}}}} \right) X^{(l-1)} W_{OV}^{(l,h)} \end{aligned} \quad (2)$$

¹For mechanistic decomposition, we equivalently split W_O into per-head components $W_O^{(h)}$. See Appendix C for details.

NSDS: A Data-Free Numerical and Structural Dual-Sensitivity Framework for Layer Sensitivity Estimation

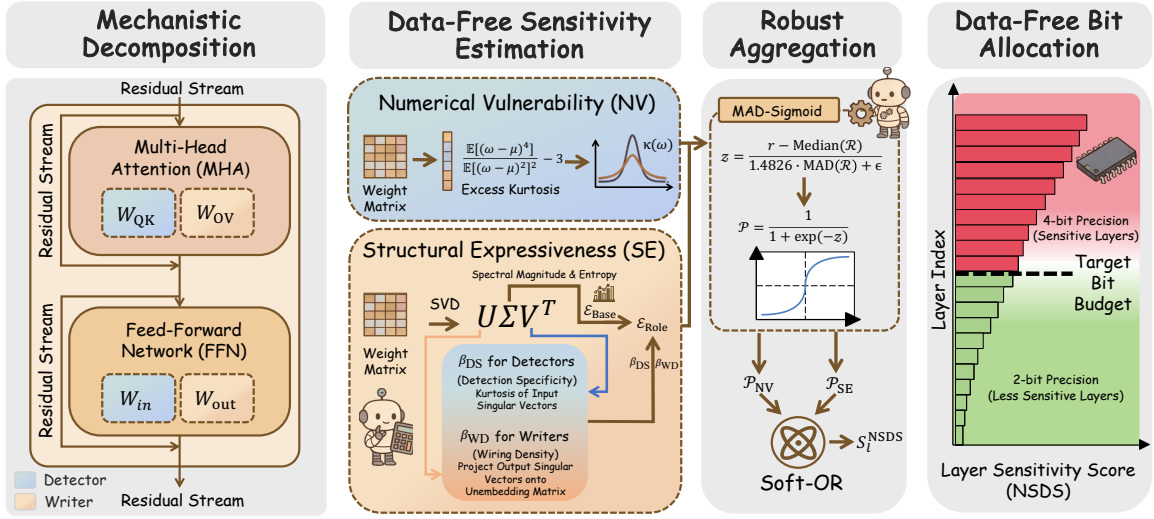


Figure 2: Overview of our NSDS framework for data-free layer sensitivity estimation. The left panel illustrates the mechanistic decomposition of a layer into distinct operational components, categorized as *Detectors* and *Writers* (§2.1). The middle panel presents the dual-view sensitivity estimation: Numerical Vulnerability (NV) and Structural Expressiveness (SE) (§2.2). The right panel shows the robust aggregation process, where MAD-Sigmoid normalizes heterogeneous scores and Soft-OR integrates them into a unified layer-wise sensitivity metric, which is then used for bit allocation under a target budget (§2.3). See Appendix B for the algorithmic description of the NSDS framework.

where $W_{\text{QK}}^{(l,h)} = W_Q^{(l,h)} W_K^{(l,h)T}$ and $W_{\text{OV}}^{(l,h)} = W_V^{(l,h)} W_O^{(l,h)}$, both $\in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$. The $W_{\text{QK}}^{(l,h)}$ component computes the attention pattern to *detect* which tokens should be attended to. The $W_{\text{OV}}^{(l,h)}$ component moves and transforms information from attended tokens, *writing* it into the residual stream.

Feed-Forward Network (FFN). The FFN takes the post-attention residual $X_{\text{mid}}^{(l)} = X^{(l-1)} + \sum_{h=1}^H \text{Attn}^{(l,h)}(X^{(l-1)})$ as input and is defined as:

$$\text{FFN}^{(l)}(X_{\text{mid}}^{(l)}) = \sigma(X_{\text{mid}}^{(l)} W_{\text{in}}^{(l)}) W_{\text{out}}^{(l)} \quad (3)$$

where $W_{\text{in}}^{(l)} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ffn}}}$, $W_{\text{out}}^{(l)} \in \mathbb{R}^{d_{\text{ffn}} \times d_{\text{model}}}$, and $\sigma(\cdot)$ is the activation function.² This operation can be viewed as *key-value memory retrieval* (Geva et al., 2021). As the keys, $W_{\text{in}}^{(l)}$ computes the neuron activation pattern, *detecting* which neuron features are relevant to the input. As the values, $W_{\text{out}}^{(l)}$ *writes* the activated neuron features back into residual stream according to their activation strengths.

Prediction as a Sum of Component Outputs. To analyze the global effect of these components, we adopt the residual stream perspective where every component contributes via addition (Mickus et al., 2022). The unnormalized output logits y for

an L -layer model with H attention heads can be expressed as:

$$y = \left(X^{(0)} + \sum_{l=1}^L \sum_{h=1}^H \text{Attn}^{(l,h)}(X^{(l-1)}) + \sum_{l=1}^L \text{FFN}^{(l)}(X_{\text{mid}}^{(l)}) \right) W_U \quad (4)$$

where $X^{(0)}$ denotes the initial state of residual stream and $W_U \in \mathbb{R}^{d_{\text{model}} \times V}$ is the unembedding matrix for a vocabulary of size V . Based on this information flow, we categorize the components into two operational roles:

- **Detector:** $W_{\text{QK}}^{(l,h)}$ and $W_{\text{in}}^{(l)}$ function as detectors.³ They compute attention or neuron activation patterns to detect related tokens or neuron features.
- **Writer:** $W_{\text{OV}}^{(l,h)}$ and $W_{\text{out}}^{(l)}$ function as writers. They process and write the detected information back into the residual stream, directly influencing the output logits in Eq. 4.

2.2 Estimating Numerical Vulnerability (NV) and Structural Expressiveness (SE)

We comprehensively estimate layer sensitivity from two complementary perspectives: NV and SE.

²Modern SwiGLU-based LLMs include a gate projection $W_{\text{gate}}^{(l)} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ffn}}}$. See Appendix D.1 for more details.

³For LLMs with $W_{\text{gate}}^{(l)}$, we categorize $W_{\text{gate}}^{(l)}$ as a detector. See Appendix D.1 for more details.

Numerical Vulnerability (NV). We utilize *excess kurtosis* to estimate the numerical vulnerability of each component. Kurtosis measures the “tailedness” of a distribution. A larger kurtosis value indicates a heavy-tailed distribution with a higher density of extreme outliers. Quantizing components with such extreme outliers stretches the quantization range, easily triggering severe quantization degradation (Xiao et al., 2023; Pham et al., 2025; Su et al., 2026a,b,c). Given a weight matrix W of a component, we first flatten it into a 1D vector w . The excess kurtosis κ is then computed as:

$$\kappa(w) = \frac{\mathbb{E}[(w - \mu)^4]}{(\mathbb{E}[(w - \mu)^2])^2} - 3 \quad (5)$$

where μ denotes the mean of flattened vector w .

Structural Expressiveness (SE). We estimate structural expressiveness of each component by using its spectral magnitude and spectral entropy. Specifically, for a component W , we perform Singular Value Decomposition (SVD) such that $W = U\Sigma V^T$. Let $\sigma = [\sigma_1, \sigma_2, \dots, \sigma_k]$ denote the vector of its singular values.

The spectral magnitude, defined as the sum of the singular values ($\|\sigma\|_1 = \sum_{i=1}^k \sigma_i$), captures the total structural energy. Meanwhile, the spectral entropy (H) captures the representational richness and information capacity (Roy and Vetterli, 2007; Xiong et al., 2025). To compute the spectral entropy, we first normalize the singular values into a distribution $p_i = \sigma_i / \sum_{j=1}^k \sigma_j$. The spectral entropy is then computed as the Shannon entropy:

$$H(\sigma) = - \sum_{i=1}^k p_i \log p_i \quad (6)$$

Based on these theoretical properties, we formulate the base structural expressiveness ($\mathcal{E}_{\text{base}}$) as:

$$\mathcal{E}_{\text{base}} = \|\sigma\|_1 \times \exp(H(\sigma)) \quad (7)$$

However, $\mathcal{E}_{\text{base}}$ is derived from raw singular values, reflecting intrinsic spectral properties of each component while overlooking its operational role. A singular component with a small singular value may still play an important mechanistic role if it exhibits specific detection capabilities or targeted writing impacts. To bridge this gap, we reweight the raw singular values based on component’s operational role. We propose differentiated reweighting factors: *Detection Specificity* (β_{DS}) for Detectors and *Writing Density* (β_{WD}) for Writers.

1) β_{DS} for Detectors ($W_{\text{up}}, W_{\text{QK}}$). Detectors rely on their input singular vectors (V) to detect patterns from the residual stream. To quantify this detection specificity, we use the excess kurtosis of these input singular vectors. Specifically, for the i -th singular component, the reweighting factor $\beta_{\text{DS}}^{(i)}$ is computed as:

$$\beta_{\text{DS}}^{(i)} = \kappa(v_i) \quad (8)$$

where v_i is the i -th column vector of V , and $\kappa(\cdot)$ denotes the excess kurtosis. A higher excess kurtosis in v_i signifies that the detector is highly sharp and focuses on extremely specific patterns or features.

2) β_{WD} for Writers ($W_{\text{down}}, W_{\text{OV}}$). Writers rely on their output singular vectors (U) to write the processed information into the residual stream. Inspired by the logits lens framework (nostalgebraist, 2020; Millidge and Black, 2022), we quantify this writing density by projecting these output singular vectors onto the vocabulary unembedding matrix (W_U). Specifically, for the i -th singular component, the reweighting factor $\beta_{\text{WD}}^{(i)}$ is defined as the L_1 norm of this projection magnitude:

$$\beta_{\text{WD}}^{(i)} = \|W_U^T u_i\|_1 \quad (9)$$

where u_i is the i -th column vector of U . A larger $\beta_{\text{WD}}^{(i)}$ indicates a stronger projection onto the vocabulary space, thereby exerting a greater impact on the final output distribution.

By element-wisely substituting the raw singular values σ_i with their reweighted counterparts (i.e., $\sigma_i \leftarrow \sigma_i \cdot \beta_{\text{DS}}^{(i)}$ or $\sigma_i \leftarrow \sigma_i \cdot \beta_{\text{WD}}^{(i)}$) back into Eq. 7, we obtain the final role-aware structural expressiveness ($\mathcal{E}_{\text{role}}$) for each component.

2.3 Score Aggregation and Bit Allocation

Robust Normalization via MAD-Sigmoid. The estimated NV and SE scores generally exhibit different scales and variances. Direct aggregation would cause one metric to dominate due to scale mismatch. To align these metrics, we convert them into robust Z-scores (Iglewicz and Hoaglin, 1993) based on the Median Absolute Deviation (MAD).

Let $\mathcal{C} = \{W_{\text{QK}}, W_{\text{OV}}, W_{\text{in}}, W_{\text{out}}\}$ denote the set of component types in each layer, with $c \in \mathcal{C}$. For a set of raw scores $\mathcal{R} = \{r^{(l,c)}\}_{l=1}^L$ (i.e., the SE or NV raw scores of component c across all layers), we compute:

$$z^{(l,c)} = \frac{r^{(l,c)} - \text{Median}(\mathcal{R})}{1.4826 \cdot \text{MAD}(\mathcal{R}) + \epsilon} \quad (10)$$

where $\text{MAD}(\mathcal{R}) = \text{Median}(|r^{(l,c)} - \text{Median}(\mathcal{R})|)$, and $\epsilon > 0$ is a small constant for numerical stability. The factor 1.4826 scales MAD to be comparable to a standard deviation under normality. We then map $z^{(l,c)}$ into the $(0, 1)$ range using the Sigmoid function to facilitate probabilistic-style aggregation: $\mathcal{P}^{(l,c)} = \frac{1}{1 + \exp(-z^{(l,c)})}$.

Score Aggregation via Soft-OR. We define the Soft-OR operator as:

$$\text{Soft-OR}(\{\mathcal{P}_i\}) = 1 - \prod_i (1 - \mathcal{P}_i) \quad (11)$$

For two terms, Soft-OR reduces to $\mathcal{P}_1 + \mathcal{P}_2 - \mathcal{P}_1\mathcal{P}_2$, which increases whenever either term is large. Thus, Soft-OR emphasizes the most sensitive components rather than averaging them.⁴

Let $\kappa^{(l,c)}$ and $\mathcal{E}_{\text{role}}^{(l,c)}$ be the NV and SE raw scores for component c at layer l . The aggregation process proceeds as follows: **1)** Apply MAD-Sigmoid to normalize $\kappa^{(l,c)}$ and $\mathcal{E}_{\text{role}}^{(l,c)}$ into $\mathcal{P}_{\text{NV}}^{(l,c)}$ and $\mathcal{P}_{\text{SE}}^{(l,c)}$; **2)** Apply Soft-OR across all components within layer l to obtain the layer-wise sensitivities: $S_l^{\text{NV}} = \text{Soft-OR}(\{\mathcal{P}_{\text{NV}}^{(l,c)}\}_{c \in \mathcal{C}})$ and $S_l^{\text{SE}} = \text{Soft-OR}(\{\mathcal{P}_{\text{SE}}^{(l,c)}\}_{c \in \mathcal{C}})$; **3)** Merge the layer-wise scores to derive the final NSDS metric:

$$S_l^{\text{NSDS}} = \text{Soft-OR}(S_l^{\text{NV}}, S_l^{\text{SE}}) \quad (12)$$

Data-Free Layer-wise Bit Allocation. Given a target average-bit budget $\bar{b} \in [2, 4]$, the ratio of 4-bit layers under the assumption of equal-sized layers is $\rho = \frac{\bar{b}-2}{4-2}$. This yields a closed-form number of 4-bit layers $L_4 = \text{round}(\rho \cdot L)$, where L is the total number of layers. Thus, we allocate 4-bit precision to L_4 most sensitive layers (i.e., with the highest S_l^{NSDS}), and assign 2-bit precision to the remaining $L_2 = L - L_4$ less sensitive layers.

3 Experiments

3.1 Experiment Settings

Evaluation Datasets and Metrics. We evaluate our method on both language modeling and language reasoning benchmarks. For language modeling, we report the Perplexity (PPL) on the WikiText-2 (Merity et al., 2016) and C4 (Raffel et al., 2020). For language reasoning, we measure the accuracy across ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019),

PIQA (Bisk et al., 2019), BoolQ (Clark et al., 2019), WinoGrande (Sakaguchi et al., 2021), and TruthfulQA (Lin et al., 2021).

Models and Baselines. In our experiments, we employ models from the Llama-2 (Touvron et al., 2023), Llama-3.1 (Grattafiori et al., 2024), and Qwen2.5 (Qwen et al., 2024) families, focusing on their 7B/8B and 13B/14B variants. In the main experiments, we compare our NSDS with calibration-free layer-wise sensitivity metrics, including: MSE, EWQ (Behtash et al., 2025), ZD (Dumitru et al., 2024), and KurtBoost (Zhang et al., 2025a). For further analysis, we extend our comparison to several calibration-based layer-wise sensitivity metrics, including: LIM (Dumitru et al., 2024), LSAQ (Zeng et al., 2025), LLM-MQ (Li et al., 2023), and LieQ (Xiao et al., 2026). Additionally, we compare our method against SliM-LLM (Huang et al., 2025), a recent calibration-based group-wise mixed-precision method, to thoroughly demonstrate the efficacy of our framework. See Appendix E for the details of baseline methods.

Implementation Details. In this work, we employ HQQ (Badri and Shaji, 2023), a calibration-free quantization method, as the default backend and conduct all evaluations under a target average-bit budget of $\bar{b} = 3$, unless otherwise specified. We set ϵ in Eq. 10 to 10^{-12} . For QK and OV components, NV and SE scores are first computed per head and then averaged across heads. To filter out noise during the SE score computation (§2.2), we truncate all SVD matrices (including the unembedding matrix W_U) to retain only the top singular components that cumulatively account for 90% of the total variance. For settings that require calibration data, we randomly sample 128 sequences from Pile (Gao et al., 2020), each with 2048 tokens. See Appendix D for more implementation details.

3.2 Performance of NSDS

Table 1 reports the performance of the baseline methods and our proposed NSDS framework across various language reasoning and language modeling benchmarks for both Llama-3.1-8B and Qwen2.5-7B models. The results demonstrate that our NSDS consistently outperforms all baseline methods.

For instance, on Qwen2.5-7B, our NSDS surpasses the ZD baseline by 2.26%, 1.58%, and 3.07% on ARC-C, HellaSwag, and PIQA, respectively. Furthermore, relative to the baseline KurtBoost on Llama-3.1-8B, NSDS achieves accuracy

⁴To mitigate numerical saturation across n components, here Soft-OR is computed as $1 - \prod_{i=1}^n (1 - \mathcal{P}_i)^{\frac{1}{n}}$.

	Language Reasoning ($\uparrow$)						Language Modeling ($\downarrow$)	
	ARC-C	Hellaswag	PIQA	BoolQ	Winogrande	TruthfulQA	Wikitext-2	C4
Llama-3.1-8B								
FP16	57.76	81.97	80.09	82.11	77.35	28.40	6.24	8.95
MSE	38.73	64.21	68.58	60.73	65.82	23.95	9.22	12.35
EWQ	40.17	67.16	71.49	64.52	67.43	23.34	8.38	11.27
ZD	39.45	66.94	69.63	62.88	67.51	24.49	8.42	11.51
KurtBoost	41.87	68.22	70.76	67.76	69.69	25.15	7.74	10.65
NSDS (Ours)	43.16	69.67	73.31	66.89	72.28	26.43	7.25	9.97
Qwen2.5-7B								
FP16	63.82	80.22	78.73	84.65	76.01	39.05	6.85	10.44
MSE	53.68	67.35	72.92	69.85	65.18	27.82	11.65	14.22
EWQ	51.85	67.52	72.10	67.96	64.35	26.95	11.48	14.05
ZD	55.82	70.45	73.01	72.95	65.25	28.94	9.23	12.15
KurtBoost	54.25	69.92	72.48	70.40	65.72	28.35	9.84	13.68
NSDS (Ours)	57.08	71.56	75.25	73.18	68.56	31.15	8.61	11.83

Table 1: Results of baseline methods and our NSDS on Llama-3.1-8B and Qwen2.5-7B, evaluated on six language reasoning and two language modeling benchmarks. See Appendix E for the details of baseline methods.

gains of 3.08% on ARC-C, 2.13% on Hellaswag, and a more substantial 3.60% on PIQA. On language modeling tasks, NSDS also effectively minimizes the perplexity degradation, outperforming KurtBoost by 0.49 on Wikitext-2. Based on these observations, we derive the following conclusion:

Key Finding 1

By jointly considering numerical and structural sensitivity, our NSDS enables comprehensive sensitivity estimation, leading to more effective layer ranking.

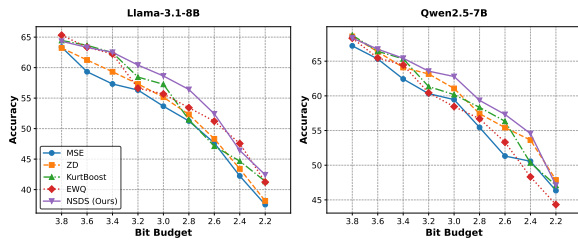


Figure 3: Average accuracy of NSDS and baselines on language reasoning benchmarks across different bit budgets for Llama-3.1-8B and Qwen2.5-7B.

3.3 Further Analysis

NSDS Across Different Bit Budgets. We report the average accuracy of NSDS and baseline methods for Llama-3.1-8B and Qwen2.5-7B across different bit budgets in Fig. 3. As shown in Fig. 3, all methods achieve similar performance at higher bit budgets. However, as the bit budget decreases, the baseline methods experience an early and sharp decline. For instance, the performance of EWQ and KurtBoost begins to degrade noticeably around 3.2

bits. In contrast, NSDS maintains a stable accuracy and does not show significant degradation until the bit budget drops to 2.6. Overall, we conclude that:

Key Finding 2

NSDS shows superior robustness across different bit budgets, maintaining high performance at lower bit budgets where baselines suffer severe degradation.

	Llama-2-13B		Qwen2.5-14B	
	Acc. ($\uparrow$)	PPL ($\downarrow$)	Acc. ($\uparrow$)	PPL ($\downarrow$)
FP16	67.23	5.67	73.25	7.22
MSE	59.91	9.38	64.29	10.62
EWQ	61.68	7.61	63.36	11.13
ZD	60.99	7.98	65.42	10.25
KurtBoost	62.08	7.51	65.13	10.41
NSDS (Ours)	63.01	7.11	66.15	9.96

Table 2: Average PPL and accuracy of NSDS and baselines on language modeling and reasoning benchmarks for larger-scale LLMs. See Table 3 for detailed results.

NSDS on Larger Model Scale. Having verified the effectiveness of our NSDS framework on 7B and 8B models, we further evaluate its scalability and generalization to larger-scale LLMs. As shown in Table 2, NSDS consistently outperforms all baseline methods in terms of both average PPL and average language reasoning accuracy. In particular, compared to the strong baseline KurtBoost, NSDS achieves an average accuracy improvement of 1.50% and a PPL reduction of 0.40 on Llama-2-13B. Similarly, on the Qwen2.5-14B model, our method surpasses the KurtBoost baseline by 1.57%

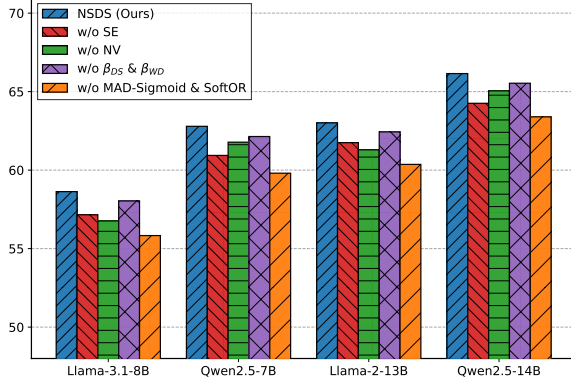


Figure 4: Average accuracy of the ablation analysis on NSDS. “w/o” denotes the exclusion of a specific component from NSDS during layer sensitivity estimation.

in average accuracy and lowers the average PPL by 0.45. These findings demonstrate the robust scalability and broad effectiveness of our NSDS across different model scales and families.

Ablation Analysis of NSDS. We conduct ablation analysis to examine the individual contributions of NV, SE, the reweighting factors (β_{DS} & β_{WD}), and the aggregation operations (MAD-Sigmoid & Soft-OR) within our NSDS framework. We report the average accuracy and PPL on language reasoning and language modeling benchmarks in Fig. 4 and Fig. 8, respectively.

As shown in Fig. 4, the results demonstrate consistent performance drops across all models for all ablation variants. Notably, the performance drop is more pronounced when the aggregation functions are excluded (“w/o MAD-Sigmoid & Soft-OR”), resulting in an accuracy drop of up to 4.08% on Qwen2.5-14B. This highlights the crucial role of robust normalization and suitable aggregation operations when combining the SE and NV scores. Additionally, removing the reweighting factors (“w/o β_{DS} & β_{WD} ”) causes a consistent accuracy drop, validating the effectiveness of role-aware singular value refinement. These observations indicate that:

Key Finding 3

Proper normalization and aggregation are crucial for deriving layer-wise scores, and integrating NV with SE outperforms using either metric alone.

Compared with Calibration-based Methods. We compare our data-free NSDS with several calibration-based layer sensitivity metrics, including LIM, LSAQ, LLM-MQ, and LieQ. We report

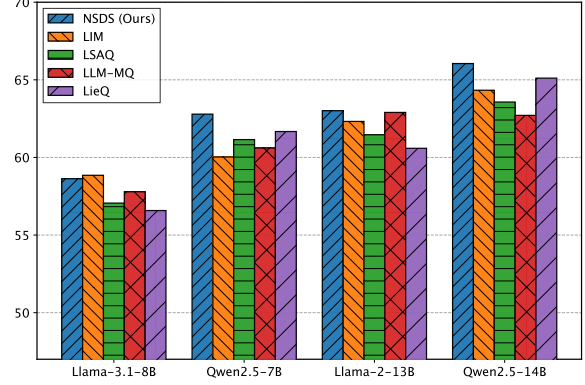


Figure 5: Average accuracy comparison between NSDS framework and calibration-based baselines across generic reasoning benchmarks.

the average accuracy and PPL on language reasoning and language modeling benchmarks in Fig. 5 and Fig. 9, respectively.

As shown in Fig. 5, NSDS consistently ranks in the top two across all four evaluated LLMs, demonstrating cross-model robustness despite requiring no calibration data. Specifically, NSDS achieves the highest average accuracy on three models (Qwen2.5-7B, Llama-2-13B, and Qwen2.5-14B), and ranks second on the Llama-3.1-8B, with only a marginal gap (within 0.49%) behind the best-performing baseline. In contrast, the calibration-based baselines exhibit notable performance fluctuations depending on the target model. For instance, while LIM achieves the top rank on Llama-3.1-8B, its performance drops to the last place on Qwen2.5-7B. Similarly, LLM-MQ performs the second best on Llama-2-13B but falls behind on Qwen2.5-14B. This instability suggests that the effectiveness of calibration-based methods depends on how well the calibration data aligns with the specific target model, making them less robust across different models compared to NSDS. Based on these observations, we derive the following conclusion:

Key Finding 4

Although fully data-free, our NSDS framework maintains strong cross-model robustness, whereas calibration-based methods lack such consistency.

Integration with Other PTQ Backends. Our NSDS is orthogonal to the choice of the PTQ backend. We replace the default HQQ backend with a stronger one, i.e., GPTQ, and compare it against SliM-LLM, a group-wise calibration-based mixed-

precision method. We report the average accuracy and PPL in Fig. 6 and Fig. 10, respectively. As shown in Fig. 6, when integrated with GPTQ, the performance of NSDS improves and shows comparable or even slightly superior performance to SliM-LLM. Notably, SliM-LLM relies on calibration data and group-wise operations that are less hardware-friendly, whereas NSDS is calibration-free and adopts a hardware-friendly layer-wise scheme. These observations indicate that:

Key Finding 5

Our NSDS is orthogonal to the underlying PTQ backend. It serves as a plug-and-play method where integrating a stronger quantization technique leads to better performance.

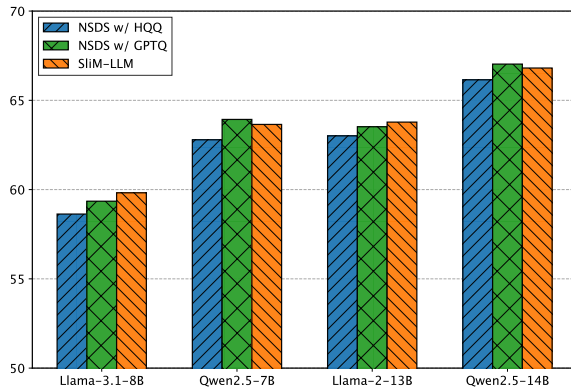


Figure 6: Average accuracy on language reasoning benchmarks when integrating NSDS with different PTQ backends, compared against the SliM-LLM.

4 Related Work

Model Quantization. Quantization techniques for Large Language Models (LLMs) aim to reduce the precision of weights and activations to lower bits without significantly compromising model performance (Zhu et al., 2024; Gong et al., 2025). The two most prominent paradigms are Quantization-Aware Training (QAT) (Liu et al., 2024; Chen et al., 2025) and Post-Training Quantization (PTQ) (Bannar et al., 2019; Badri and Shaji, 2023; Frantar et al., 2023; Lin et al., 2024). QAT incorporates quantization directly into the training phase, allowing the model to adapt to quantization errors. In contrast, PTQ applies quantization strictly after the model has been fully trained. As PTQ avoids the resource-intensive and time-consuming training processes required by QAT, it is more widely used in practice. In this paper, we focus on a specific class of PTQ

known as weight-only quantization. This approach can be further categorized into calibration-based (e.g., GPTQ (Frantar et al., 2023) and AWQ (Lin et al., 2024)) and calibration-free methods (e.g., HQQ (Badri and Shaji, 2023) and BnB (Dettmers et al., 2023)), depending on whether an extra calibration dataset is adopted.

Layer-wise Mixed-Precision Quantization (LMPQ). Standard PTQ methods typically apply uniform bit allocation, often severely degrading performance under extreme low-bit regimes. LMPQ mitigates this by assigning higher precision to crucial layers and fewer bits elsewhere (Zhang et al., 2025a). As bit-width remains uniform within each layer, LMPQ preserves tensor contiguity and kernel regularity, making it more hardware-friendly than other mixed-precision schemes. Prior LMPQ methods are either search-based or criterion-based. Search-based methods (Wang et al., 2019; Yu et al., 2020) are computationally intensive, making them impractical for many real applications. Conversely, criterion-based methods efficiently estimate layer sensitivity via proxy metrics to guide allocation.

Depending on whether calibration data is required, criterion-based methods can be further divided into calibration-based or calibration-free approaches. Calibration-based criteria leverage loss landscape information (Dong et al., 2020; Li et al., 2023), layer output discrepancies (Dumitru et al., 2024; Zeng et al., 2025), or output representation compactness (Xiao et al., 2026). However, they risk overfitting to calibration data and require resource-intensive forward and backward passes. Thus, we focus on the calibration-free paradigm. Existing calibration-free methods rely on heuristics like weight outliers (Dumitru et al., 2024; Zhang et al., 2025a), quantization errors (Kloberdanz and Le, 2023), or parameter distribution entropy (Behr et al., 2025). However, they typically utilize single-faceted metrics focused solely on numerical magnitudes, ignoring the rich structural information of weights and treating all intra-layer modules uniformly. To overcome these limitations, our NSDS framework mechanistically decomposes layers into distinct operational components, comprehensively estimating sensitivity from both numerical and structural perspectives.

5 Conclusion

In this work, we propose NSDS, a novel calibration-free LMPQ framework that estimates layer sensi-

tivity from both numerical and structural perspectives. Unlike previous calibration-free methods that uniformly treat all weight modules and rely solely on single numerical properties, NSDS mechanistically decompose each layer into distinct operational components and introduces role-aware sensitivity modeling that distinguishes between different functional modules within a layer. Through the integration of MAD-Sigmoid normalization and Soft-OR operation, NSDS produces a unified layer-wise ranking to guide bit allocation under a fixed budget. Extensive experiments validate that NSDS consistently achieves competitive or superior performance compared to various baselines across different LLMs and downstream tasks, without relying on any calibration data. Our further analysis also offers valuable insights for future research.

Limitations

While our proposed NSDS framework demonstrates strong performance and robustness across multiple model families and benchmarks, our experiments are limited to models up to 14B parameters due to computational resource constraints. We have not evaluated NSDS on larger-scale models (e.g., 30B, 70B, or beyond), where sensitivity patterns may differ. Future work will evaluate NSDS on larger LLMs and additional model families to further investigate its scalability and generalization.

Acknowledgement

This project is supported in part by the Theme-based Research Scheme (TRS) projects T45-701/22-R and T41-517/25-N, and GRF Project 17203224 of the Research Grants Council (RGC), Hong Kong SAR, and in part by the AVNET-HKU Emerging Microelectronics & Ubiquitous Systems (EMUS) Lab.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. [GQA: Training generalized multi-query transformer models from multi-head checkpoints](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore. Association for Computational Linguistics.

Hicham Badri and Appu Shaji. 2023. Half-quadratic quantization of large machine learning models. *Dan*

Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob, pages 10088–10115.

- Ron Banner, Yury Nahshan, and Daniel Soudry. 2019. Post training 4-bit quantization of convolutional networks for rapid-deployment. *Advances in neural information processing systems*, 32.
- Alireza Behtash, Marijan Fofonjka, Ethan Baird, Tyler Mauer, Hossein Moghimifam, David Stout, and Joel Dennison. 2025. [Universality of layer-level entropy-weighted quantization beyond model architecture and size](#). *Preprint*, arXiv:2503.04704.
- Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. [Piqa: Reasoning about physical commonsense in natural language](#). *Preprint*, arXiv:1911.11641.
- Mengyu Bu, Shaolei Zhang, Zhongjun He, Hua Wu, and Yang Feng. 2025. [AlignX: Advancing multilingual large language models with multilingual representation alignment](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6460–6489, Suzhou, China. Association for Computational Linguistics.
- Yuan Chang, Ziyue Li, Hengyuan Zhang, Yuanbo Kong, Yanru Wu, Zhijiang Guo, and Ngai Wong. 2025. [Treereview: A dynamic tree of questions framework for deep and efficient llm-based scientific peer review](#). *arXiv preprint arXiv:2506.07642*.
- Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. 2025. [EfficientQAT: Efficient quantization-aware training for large language models](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10081–10100, Vienna, Austria. Association for Computational Linguistics.
- Xinrong Chen, Xu Chu, Yingmin Qiu, Hengyuan Zhang, Jing Xiong, Shiyu Tang, Shuai Liu, Shaokang Yang, Cheng Yang, Hayden Kwok-Hay So, and 1 others. 2026. [Residual decoding: Mitigating hallucinations in large vision-language models via history-aware residual guidance](#). *arXiv preprint arXiv:2602.01047*.
- Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. [BoolQ: Exploring the surprising difficulty of natural yes/no questions](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.
- 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.

- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: efficient finetuning of quantized llms. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23*, Red Hook, NY, USA. Curran Associates Inc.
- Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. *Advances in neural information processing systems*, 33:18518–18529.
- Xufeng Duan, Zhaoqian Yao, Xinyu Zhou, Zihao Fu, Beijia Kan, Yibo Wang, Bei Xiao, Hengyuan Zhang, Xuemei Tang, Ercong Nie, and Zhenguang G. Cai. 2026. [Mechanistic interpretability for understanding language abilities in large language models: A survey](#). *Preprints*.
- Razvan-Gabriel Dumitru, Vikas Yadav, Rishabh Maheshwary, Paul-Ioan Clotan, Sathwik Tejaswi Madhusudhan, and Mihai Surdeanu. 2024. [Layer-wise quantization: A pragmatic and effective method for quantizing llms beyond integer bit-levels](#). *Preprint*, arXiv:2406.17415.
- Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, and 1 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12.
- Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R. Costa-jussà. 2024. [A primer on the inner workings of transformer-based language models](#). *Preprint*, arXiv:2405.00208.
- Elias Frantar, Saleh Ashkboos, Torsten Hoefer, and Dan Alistarh. 2023. [OPTQ: Accurate quantization for generative pre-trained transformers](#). In *The Eleventh International Conference on Learning Representations*.
- Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, and 1 others. 2020. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.
- Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. [Transformer feed-forward layers are key-value memories](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Ruihao Gong, Yifu Ding, Zining Wang, Chengtao Lv, Xingyu Zheng, Jinyang Du, Yang Yong, Shiqiao Gu, Haotong Qin, Jinyang Guo, and 1 others. 2025. A survey of low-bit large language models: Basics, systems, and algorithms. *Neural networks*, 192:107856.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, and 1 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Wei Huang, Haotong Qin, Yangdong Liu, Yawei Li, Qinshuo Liu, Xianglong Liu, Luca Benini, Michele Magno, Shiming Zhang, and XIAOJUAN QI. 2025. [Slim-LLM: Saliency-driven mixed-precision quantization for large language models](#). In *Forty-second International Conference on Machine Learning*.
- Boris Iglewicz and David C Hoaglin. 1993. *Volume 16: how to detect and handle outliers*. Quality Press.
- Eliska Kloberdanz and Wei Le. 2023. [Mixquant: Mixed precision quantization with a bit-width optimization search](#). *Preprint*, arXiv:2309.17341.
- Shashank Landge, Abhishek Patil, Tejas kamble, Bhushan Buddhivant, and Priyanka Joshi. 2025. [Intrinsic structure as a proxy for saliency: Svd-based weight preservation for mixed-precision quantization in large language models](#). *Preprint*, arXiv:2512.01343.
- Sangjun Lee, Seung-taek Woo, Jun-gyu Jin, Changhun Lee, and Eunhyeok Park. 2025. [AMQ: Enabling AutoML for mixed-precision weight-only quantization of large language models](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 35532–35550, Suzhou, China. Association for Computational Linguistics.
- Shiyao Li, Xuefei Ning, Ke Hong, Tengxuan Liu, Lunling Wang, Xiuhong Li, Kai Zhong, Guohao Dai, Huazhong Yang, and Yu Wang. 2023. Llm-mq: Mixed-precision quantization for efficient llm deployment. In *The Efficient Natural Language and Speech Processing Workshop with NeurIPS*, volume 9, page 3.
- Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, and 1 others. 2025. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*.
- Xiao Liang, Zhong-Zhi Li, Yeyun Gong, Yang Wang, Hengyuan Zhang, Yelong Shen, Ying Nian Wu, and Weizhu Chen. 2025. Sws: Self-aware weakness-driven problem synthesis in reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.08989*.
- Xiao Liang, Zhong-Zhi Li, Zhenghao Lin, Eric Hancheng Jiang, Hengyuan Zhang, Yelong Shen, Kai-Wei Chang, Ying Nian Wu, Yeyun Gong, and Weizhu Chen. 2026. [Training llms for divide-and-conquer reasoning elevates test-time scalability](#). *Preprint*, arXiv:2602.02477.
- Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024.

- Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6:87–100.
- Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. [Truthfulqa: Measuring how models mimic human falsehoods](#). *Preprint*, arXiv:2109.07958.
- Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2024. [LLM-QAT: Data-free quantization aware training for large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 467–484, Bangkok, Thailand. Association for Computational Linguistics.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. [Pointer sentinel mixture models](#). *Preprint*, arXiv:1609.07843.
- Timothee Mickus, Denis Paperno, and Mathieu Constant. 2022. [How to dissect a Muppet: The structure of transformer embedding spaces](#). *Transactions of the Association for Computational Linguistics*, 10:981–996.
- Beren Millidge and Sid Black. 2022. The singular value decompositions of transformer weight matrices are highly interpretable. In *AI Alignment Forum*, page 17.
- nostalgebraist. 2020. [Interpreting GPT: the logit lens](#).
- Cuong Pham, Hoang Anh Dung, Cuong C. Nguyen, Trung Le, Gustavo Carneiro, Jianfei Cai, and Thanh-Toan Do. 2025. [Adaptive layer-wise transformations for post-training quantization of large language models](#). *Preprint*, arXiv:2511.17809.
- Libo Qin, Qiguang Chen, Yuhang Zhou, Zhi Chen, Yinghui Li, Lizi Liao, Min Li, Wanxiang Che, and Philip S. Yu. 2025. [A survey of multilingual large language models](#). *Patterns*, 6(1):101118.
- Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, and 25 others. 2024. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.
- Olivier Roy and Martin Vetterli. 2007. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pages 606–610. IEEE.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. [Winogrande: an adversarial winograd schema challenge at scale](#). *Commun. ACM*, 64(9):99–106.
- Zunhai Su, Qingyuan Li, HaoZhang, Weihao Ye, Qibo Xue, Yulei Qian, Ngai Wong, and Kehong Yuan. 2026a. [Unveiling super experts in mixture-of-experts large language models](#). In *The Fourteenth International Conference on Learning Representations*.
- Zunhai Su, Bohan Sun, Xialie Zhuang, Shuibai Zhang, He Xiao, Jing Xiong, Hengyuan Zhang, Zhongzhu Zhou, Tiantian Zhang, Ngai Wong, and 1 others. 2026b. Massive activations in hybrid linear attention large language models: Pre-attention spikes and interspike plateaus. *arXiv preprint arXiv:2608.12149*.
- Zunhai Su, Hengyuan Zhang, Wei Wu, Yifan Zhang, Yaxiu Liu, He Xiao, Qingyao Yang, Yuxuan Sun, Rui Yang, Chao Zhang, and 1 others. 2026c. Attention sink in transformers: A survey on utilization, interpretation, and mitigation. *arXiv preprint arXiv:2604.10098*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, and 1 others. 2023. [Llama 2: Open foundation and fine-tuned chat models](#). *Preprint*, arXiv:2307.09288.
- Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. 2019. Haq: Hardware-aware automated quantization with mixed precision. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 8604–8612. IEEE.
- Miles Williams and Nikolaos Aletras. 2024. [On the impact of calibration data in post-training quantization and pruning](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10100–10118, Bangkok, Thailand. Association for Computational Linguistics.
- Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pages 38087–38099. PMLR.
- He Xiao, Qingyao Yang, Dirui Xie, Wendong Xu, Zunhai Su, Runming Yang, Haobo Liu, Wenyong Zhou, Zhengwu Liu, and Ngai Wong. 2026. Exploring layer-wise information effectiveness for post-training quantization in small language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 15753–15764.
- Yijia Xiao, Edward Sun, Tong Chen, Fang Wu, Di Luo, and Wei Wang. 2025. [Trading-r1: Financial trading with llm reasoning via reinforcement learning](#). *Preprint*, arXiv:2509.11420.
- Jing Xiong, Qi Han, Yunta Hsieh, Hui Shen, Huajian Xin, Chaofan Tao, Chenyang Zhao, Hengyuan Zhang,

- Taiqiang Wu, Zhen Zhang, Haochen Wang, Zhongwei Wan, Lingpeng Kong, and Ngai Wong. 2026. [Mmformalizer: Multimodal autoformalization in the wild](#). *Preprint*, arXiv:2601.03017.
- Jing Xiong, Jianghan Shen, Fanghua Ye, Chaofan Tao, Zhongwei Wan, Jianqiao Lu, Xun Wu, Chuanyang Zheng, Zhijiang Guo, Min Yang, and 1 others. 2025. Uncomp: Can matrix entropy uncover sparsity?—a compressor design from an uncertainty-aware perspective. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 4179–4199.
- Zhao Yang, Yuxuan Jiang, Ting-Chih Chen, Lincen Yang, Annie Wong, Chao Gao, Jacob E Kooi, Zhong Li, Jiayang Shi, Kevin Qiu, and 1 others. 2026. Modularized reinforcement learning on llms: From mdp creation to exploration and learning. *arXiv preprint arXiv:2606.21943*.
- Haibao Yu, Qi Han, Jianbo Li, Jianping Shi, Guangliang Cheng, and Bin Fan. 2020. [Search what you want: Barrier panelty nas for mixed precision quantization](#). *Preprint*, arXiv:2007.10026.
- Yiyao Yu, Yuxiang Zhang, Dongdong Zhang, Xiao Liang, Hengyuan Zhang, Xingxing Zhang, Ziyi Yang, Mahmoud Khademi, Hany Awadalla, Junjie Wang, and 1 others. 2025. Chain-of-reasoning: Towards unified mathematical reasoning in large language models via a multi-paradigm perspective. *arXiv preprint arXiv:2501.11110*.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. [HellaSwag: Can a machine really finish your sentence?](#) In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.
- Binrui Zeng, Bin Ji, Xiaodong Liu, Jie Yu, Shasha Li, Jun Ma, Xiaopeng Li, Shangwen Wang, Xinran Hong, and Yongtao Tang. 2025. Lsaq: Layer-specific adaptive quantization for large language model deployment. In *2025 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.
- Feng Zhang, Yanbin Liu, Weihua Li, Jie Lv, Xiaodan Wang, and Quan Bai. 2025a. [Towards superior quantization accuracy: A layer-sensitive approach](#). *Preprint*, arXiv:2503.06518.
- Hengyuan Zhang, Xinrong Chen, Yingmin Qiu, Xiao Liang, Ziyue Li, Guanyu Wang, Weiping Li, Tong Mo, Hayden Kwok-Hay So, and Ngai Wong. 2025b. [Guilomo: Allocating expert number and rank for lora-moe via bilevel optimization with guidedselection vectors](#). *Preprint*, arXiv:2506.14646.
- Hengyuan Zhang, Chenming Shang, Sizhe Wang, Dongdong Zhang, Yiyao Yu, Feng Yao, Renliang Sun, Yujiu Yang, and Furu Wei. 2025c. [ShifCon: Enhancing non-dominant language capabilities with a shift-based multilingual contrastive framework](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4818–4841, Vienna, Austria. Association for Computational Linguistics.
- Hengyuan Zhang, Yanru Wu, Dawei Li, Sak Yang, Rui Zhao, Yong Jiang, and Fei Tan. 2024. Balancing speciality and versatility: a coarse to fine framework for supervised fine-tuning large language model. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 7467–7509.
- Hengyuan Zhang, Shiping Yang, Xiao Liang, Chenming Shang, Yuxuan Jiang, Chaofan Tao, Jing Xiong, Hayden Kwok-Hay So, Ruobing Xie, Angel X Chang, and 1 others. 2025d. Find your optimal teacher: Personalized data synthesis via router-guided multi-teacher distillation. *arXiv preprint arXiv:2510.10925*.
- Hengyuan Zhang, Zhihao Zhang, Mingyang Wang, Zunhai Su, Yiwei Wang, Qianli Wang, Shuzhou Yuan, Ercong Nie, Xufeng Duan, Qibo Xue, Zeping Yu, Chenming Shang, Xiao Liang, Jing Xiong, Hui Shen, Chaofan Tao, Zhengwu Liu, Senjie Jin, Zhiheng Xi, and 9 others. 2026. [Locate, steer, and improve: A practical survey of actionable mechanistic interpretability in large language models](#). *Preprint*, arXiv:2601.14004.
- Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. [A survey on model compression for large language models](#). *Transactions of the Association for Computational Linguistics*, 12:1556–1577.

A Visualization of NSDS Sensitivity Scores

We present the visualization of Numerical Vulnerability (NV), Structural Expressiveness (SE), and our NSDS scores across layers on Llama-3.1-8B and Qwen2.5-7B in Fig. 7.

As estimated by our NSDS metric, the layer-wise sensitivity exhibits distinct distribution patterns on Llama-3.1-8B and Qwen2.5-7B. Specifically, for Llama-3.1-8B, the most crucial and sensitive layers are predominantly located at the absolute bottom, the bottom-to-middle sections, and the top-most layers. Conversely, for Qwen2.5-7B, the high-sensitivity regions are primarily concentrated at the bottom and the middle-to-top sections.

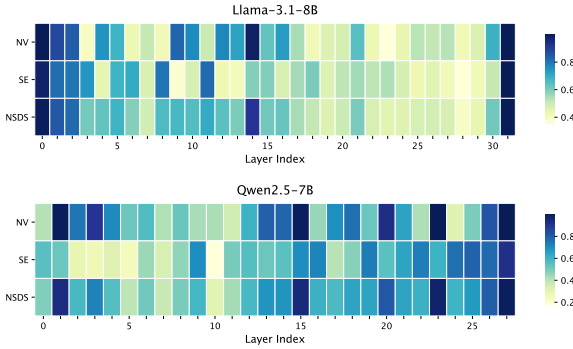


Figure 7: Visualization of NV, SE, and NSDS scores across layers on Llama-3.1-8B and Qwen2.5-7B. Darker colors indicate higher sensitivity score.

B Algorithmic Description of the NSDS Framework

To provide a clear and structured overview of our method, we summarize the entire NSDS framework in Algorithm 1. The procedure consists of three main phases. First, we iterate through all layers and mechanistically decompose each layer into distinct components (Detectors and Writers). For each component, we compute its Numerical Vulnerability (NV) using excess kurtosis, and its Structural Expressiveness (SE) through role-aware spectral capacity. Second, to ensure scale-invariant aggregation, we apply a robust MAD-Sigmoid normalization, followed by a Soft-OR operation. This operation aggregates the component-level metrics into a unified layer-wise sensitivity score, emphasizing the most vulnerable parts of the layer without suffering from numerical saturation. Finally, based on the aggregated NSDS scores, we perform a deterministic, data-free bit allocation to strictly

Algorithm 1 NSDS Framework for Data-Free Layer-wise Bit Allocation

Require: Pre-trained LLM with L layers, unembedding matrix W_U , component set $\mathcal{C} = \{W_{QK}, W_{OV}, W_{in}, W_{out}\}$, target average-bit budget $\bar{b} \in [2, 4]$.

Ensure: Layer-wise bit allocation scheme $\mathcal{B} = \{b_1, b_2, \dots, b_L\}$.

- 1: **% Phase 1: Metric Estimation (NV and SE)**
- 2: **for** $l = 1$ **to** L **do**
- 3: **for** each component $c \in \mathcal{C}$ **do**
- 4: Extract weight matrix $W^{(l,c)}$.
- 5: Compute NV score: $\kappa^{(l,c)} \leftarrow \text{Kurtosis}(W^{(l,c)})$.
- 6: Perform SVD: $U, \Sigma, V^T \leftarrow \text{SVD}(W^{(l,c)})$ with singular values σ_i .
- 7: **if** $c \in \text{Detectors}(W_{QK}, W_{in})$ **then**
- 8: Compute reweighting factors: $\beta^{(i)} \leftarrow \kappa(v_i)$.
- 9: **else if** $c \in \text{Writers}(W_{OV}, W_{out})$ **then**
- 10: Compute reweighting factors: $\beta^{(i)} \leftarrow \|W_U^T u_i\|_1$.
- 11: **end if**
- 12: Update singular values: $\sigma_i \leftarrow \sigma_i \cdot \beta^{(i)}$.
- 13: Compute role-aware SE score: $\mathcal{E}_{role}^{(l,c)} \leftarrow \|\sigma\|_1 \times \exp(H(\sigma))$.
- 14: **end for**
- 15: **end for**
- 16: **% Phase 2: Robust Normalization and Soft-OR Aggregation**
- 17: Normalize all $\kappa^{(l,c)}$ into $\mathcal{P}_{NV}^{(l,c)}$ via MAD-Sigmoid (Eq. 10).
- 18: Normalize all $\mathcal{E}_{role}^{(l,c)}$ into $\mathcal{P}_{SE}^{(l,c)}$ via MAD-Sigmoid (Eq. 10).
- 19: **for** $l = 1$ **to** L **do**
- 20: Aggregate NV: $S_l^{NV} \leftarrow 1 - \prod_{c \in \mathcal{C}} \left(1 - \mathcal{P}_{NV}^{(l,c)}\right)^{\frac{1}{|\mathcal{C}|}}$
- 21: Aggregate SE: $S_l^{SE} \leftarrow 1 - \prod_{c \in \mathcal{C}} \left(1 - \mathcal{P}_{SE}^{(l,c)}\right)^{\frac{1}{|\mathcal{C}|}}$
- 22: Compute NSDS score: $S_l^{NSDS} \leftarrow S_l^{NV} + S_l^{SE} - S_l^{NV} \cdot S_l^{SE}$.
- 23: **end for**
- 24: **% Phase 3: Data-Free Bit Allocation**
- 25: Calculate number of 4-bit layers: $L_4 \leftarrow \text{round}\left(\frac{\bar{b}-2}{2} \cdot L\right)$.
- 26: Sort all layers in descending order based on S_l^{NSDS} .
- 27: Assign $b_l = 4$ to the top L_4 layers, and $b_l = 2$ to the remaining layers.
- 28: **return** $\mathcal{B} = \{b_1, b_2, \dots, b_L\}$.

satisfy the target average-bit budget, prioritizing higher precision for the most sensitive layers.

C Splitting the Output Projection Matrix

The original implementation of multi-head attention often considers a concatenation of each attention head output before projecting it into the global weight matrix $W_O \in \mathbb{R}^{H \cdot d_{\text{head}} \times d_{\text{model}}}$. To properly decouple the components mechanistically, we split the concatenated W_O into per-head weight matrices $W_O^{(h)} \in \mathbb{R}^{d_{\text{head}} \times d_{\text{model}}}$. By doing this, the individual matrices $W_V^{(h)}$ and $W_O^{(h)}$ can be mathematically joined into a single matrix $W_{OV}^{(h)}$ for each head h .

D Implementation Details of the NSDS

In §2, we introduced the high-level formulation of the proposed NSDS metric. This appendix provides further technical details of the NSDS framework.

D.1 SwiGLU and Gate Projection

Modern Large Language Models (e.g., Llama, Qwen) frequently replace standard activation functions with SwiGLU. In this architecture, the Feed-Forward Network contains an additional gate projection weight W_{gate} . The operation is defined as:

$$\text{FFN}(X) = (\text{Swish}(XW_{\text{gate}}) \odot XW_{\text{in}}) W_{\text{out}} \quad (13)$$

Mechanistically, W_{gate} reads from the residual stream and computes a gating distribution. It acts as an informational valve, *detecting* whether specific features should be activated. Because its role is to compute a distribution that scales the hidden features, we categorize W_{gate} as a **Detector**, identical to the role of W_{in} and W_{QK} .

D.2 Handling Grouped Query Attention

To reduce memory overhead, many recent LLMs employ Grouped Query Attention (GQA) (Ainslie et al., 2023), where multiple query heads share a single key and value head. When computing the W_{QK} and W_{OV} components for models utilizing GQA, we replicate (broadcast) the shared Key (W_K) and Value (W_V) matrices across their corresponding Query (W_Q) and Output (W_O) heads. This ensures that every attention head h possesses independent, fully formed $W_{\text{QK}}^{(h)}$ and $W_{\text{OV}}^{(h)}$ matrices for computing NV and SE scores in §2.2.

D.3 SVD Top-90% Variance Truncation

When computing SE score in §2.2, incorporating the entire spectrum of singular values introduces long-tail noise. To capture the most principal structural features, we truncate the SVD matrices U and V , as well as the singular values Σ , to retain only the top components that cumulatively account for 90% of the total variance (energy). All subsequent calculations for $\mathcal{E}_{\text{role}}$ also operate on this truncated Top-90% subspace. Similarly, for unembedding matrix W_U used in $\beta_{\text{WD}}^{(i)}$, we pre-compute its Top-90% SVD subspace to filter out vocabulary noise.

D.4 Robust Sub-Linear Reweighting

When reweighting the singular values Σ using factor $\beta_{\text{DS}}^{(i)}$, extreme outliers in $\beta_{\text{DS}}^{(i)}$ can cause an expo-

ponential explosion, completely overshadowing the base structural capacity. To prevent mathematical collapse and filter out negative noise, we apply a robust sub-linear transformation to the raw reweighting factor x :

$$\beta_{\text{DS}}^{(i)} = \log(1 + \text{ReLU}(x)) \quad (14)$$

The ReLU operation elegantly sets the weight of completely flat or uniform distributions (where kurtosis < 0) to zero. The $\log(1 + \cdot)$ operation provides sub-linear scaling, ensuring that highly specific detectors are rewarded adequately without triggering numerical instability.

D.5 Specifics of the QK Component Calculation

While the up-projection (W_{up}) acts as a unidirectional feature reader relying solely on the input singular vectors (V), the QK circuit computes the interaction between Queries and Keys. Therefore, a highly specific attention routing requires both the Query vectors (U) and the Key vectors (V) to be sharp. In our implementation of the $\beta_{\text{DS}}^{(i)}$ for the QK component, the reweighting factor is computed as the product of their respective excess kurtosis values: $\beta_{\text{DS}}^{(i)} = \kappa(v_i) \times \kappa(u_i)$.

E Details of Baselines

In this Appendix, we introduce the details of the compared baselines involved in our experiments.

E.1 Calibration-free Layer-wise Sensitivity Metric

MSE. The Mean Squared Error (MSE) metric estimates layer sensitivity by measuring the total numerical deviation between the full-precision weights and their dequantized (reconstructed) counterparts. To obtain the layer-level sensitivity score, we compute the sum of squared errors across all weight matrices within the layer. Let $\mathcal{W}_l$ denote the set of all weight matrices in layer l . The sensitivity score is calculated as:

$$\mathcal{S}_{\text{MSE}}^{(l)} = \sum_{W \in \mathcal{W}_l} \|W - \hat{W}\|_F^2 \quad (15)$$

where W and $\hat{W}$ are the full-precision and dequantized weight matrices, respectively. A larger $\mathcal{S}_{\text{MSE}}^{(l)}$ indicates a higher total reconstruction error, signifying that the layer is more sensitive and crucial to preserve.

ZD. The Z-score Distribution (ZD) metric estimates layer sensitivity by analyzing the statistical distribution of parameter weights, notably requiring no calibration data. Specifically, it quantifies the importance of a layer based on the proportion of its weights that are significantly larger than the layer’s average. Let $\mathcal{W}_l$ denote the set of all weight matrices in layer l , and let N_l be the total number of scalar weight parameters in this layer. We first compute the mean μ_l and standard deviation σ_l of all weights $w \in \mathcal{W}_l$. The Z-score for each individual weight w is defined as:

$$z = \frac{w - \mu_l}{\sigma_l} \quad (16)$$

The ZD sensitivity score for layer l is then calculated as the fraction of weights whose Z-score strictly exceeds 1:

$$\mathcal{S}_{\text{ZD}}^{(l)} = \frac{1}{N_l} \sum_{W \in \mathcal{W}_l} \sum_{w \in W} \mathbb{I}(z > 1) \quad (17)$$

where $\mathbb{I}(\cdot)$ is the indicator function. According to the original formulation, a smaller $\mathcal{S}_{\text{ZD}}^{(l)}$ score implies a higher layer sensitivity, thus prioritizing the layer for higher-precision quantization.

EWQ. Entropy-Weighted Quantization (EWQ) estimates layer sensitivity by analyzing the entropy of the weight distributions. For a weight matrix $W \in \mathcal{W}_l$ with N elements $\{w_i\}_{i=1}^N$, EWQ first flattens and normalizes the weights into a probability distribution using the softmax function: $p_i = \frac{\exp(w_i)}{\sum_{j=1}^N \exp(w_j)}$. The entropy of the weight matrix, denoted as $H(W)$, is then computed as:

$$H(W) = - \sum_{i=1}^N p_i \log(p_i + \epsilon) \quad (18)$$

where ϵ is a small constant (e.g., 0.01) added for numerical stability.

To obtain the layer-level sensitivity score, EWQ calculates the parameter-weighted average entropy across all weight matrices within the layer. Let $|W| = N$ denote the total number of parameters in the weight matrix W . The sensitivity score for layer l is defined as:

$$\mathcal{S}_{\text{EWQ}}^{(l)} = \frac{\sum_{W \in \mathcal{W}_l} |W| H(W)}{\sum_{W \in \mathcal{W}_l} |W|} \quad (19)$$

A larger $\mathcal{S}_{\text{EWQ}}^{(l)}$ indicates that the layer contains higher variability and captures more complex global information, making it more sensitive to quantization.

KurtBoost. KurtBoost estimates layer sensitivity by analyzing the weight distribution’s kurtosis. For a weight matrix $W \in \mathcal{W}_l$ with N elements $\{w_i\}_{i=1}^N$, the raw kurtosis is computed as:

$$k(W) = \frac{\frac{1}{N} \sum_{i=1}^N (w_i - \bar{W})^4}{\left(\frac{1}{N} \sum_{i=1}^N (w_i - \bar{W})^2 \right)^2} \quad (20)$$

where $\bar{W}$ is the mean of the weights in W . Unlike our NSDS framework which utilizes excess kurtosis (subtracting 3), KurtBoost uses the raw kurtosis directly. The layer-wise sensitivity score $k^{(l)}$ is computed by: $k^{(l)} = \frac{1}{|\mathcal{W}_l|} \sum_{W \in \mathcal{W}_l} k(W)$.

To identify the most sensitive layers, KurtBoost constructs a difference sequence between adjacent layers, defined as $D = \{d_l\} = \{k^{(l+1)} - k^{(l)}\}$ (or alternatively $k^{(l+1)}/k^{(l)}$ for monotonically increasing patterns). Assuming the difference set D follows a normal distribution, a Z-score is computed for each variation:

$$z_l = \frac{|d_l - \mu|}{\sigma} \quad (21)$$

where μ and σ are the mean and standard deviation of D . Layers with significant variations (i.e., $z_l > 3$) are flagged as outliers. During bit allocation, these detected outlier layers are strictly prioritized for high-precision assignment. Once the outliers are preserved, the remaining layers are selected and allocated higher precision based on their descending $k^{(l)}$ score ranking until the target bit budget is exhausted.

E.2 Calibration-based Layer-wise Sensitivity Metrics

LIM. The Layer Importance Metric (LIM) estimates layer sensitivity by calculating the cosine similarity between the layer’s input and output representations. Using a set of calibration data, let $X_{\text{in}}^{(l)}$ and $X_{\text{out}}^{(l)}$ denote the input and output hidden states of layer l , respectively. The sensitivity score is formulated as the complement of their cosine similarity:

$$\mathcal{S}_{\text{LIM}}^{(l)} = 1 - \frac{X_{\text{in}}^{(l)} \cdot X_{\text{out}}^{(l)}}{\|X_{\text{in}}^{(l)}\| \|X_{\text{out}}^{(l)}\|} \quad (22)$$

A higher cosine similarity implies that the layer’s operations are highly redundant and induce minimal transformation. Therefore, a larger $\mathcal{S}_{\text{LIM}}^{(l)}$ indicates a more substantial feature transformation, which corresponds to higher layer sensitivity.

LSAQ. Layer-Specific Adaptive Quantization (LSAQ) evaluates layer sensitivity by measuring the semantic transformation between the layer’s input and output. Similar to LIM, it utilizes calibration data to obtain the hidden states $X_{\text{in}}^{(l)}$ and $X_{\text{out}}^{(l)}$. LSAQ projects these states onto the vocabulary space using the unembedding matrix W_U . It then constructs two sets, $C_{\text{in}}^{(l)}$ and $C_{\text{out}}^{(l)}$, comprising the top- k most probable tokens decoded from the input and output:

$$C_{\text{in}}^{(l)} = f_{\text{top-}k}(X_{\text{in}}^{(l)} W_U), \quad C_{\text{out}}^{(l)} = f_{\text{top-}k}(X_{\text{out}}^{(l)} W_U) \quad (23)$$

The layer sensitivity score is computed as the complement of the Jaccard similarity between these two token sets:

$$\mathcal{S}_{\text{LSAQ}}^{(l)} = 1 - \frac{|C_{\text{in}}^{(l)} \cap C_{\text{out}}^{(l)}|}{|C_{\text{in}}^{(l)} \cup C_{\text{out}}^{(l)}|} \quad (24)$$

A smaller Jaccard similarity (resulting in a higher $\mathcal{S}_{\text{LSAQ}}^{(l)}$) signifies that the layer fundamentally alters the semantic representation of the input tokens. Consequently, layers with higher $\mathcal{S}_{\text{LSAQ}}^{(l)}$ scores are considered more sensitive and are prioritized for higher quantization precision.

LLM-MQ. LLM-MQ assesses layer sensitivity by approximating the perturbation in the model’s loss function caused by quantization, utilizing a first-order Taylor expansion guided by a small calibration dataset. For a full-precision weight matrix $W \in \mathcal{W}_l$ and its corresponding gradient G derived from the calibration data, the sensitivity metric with respect to a candidate bit-width b is calculated as the absolute sum of the element-wise product between the gradient and the quantization error:

$$s_b(W) = \left| \sum G \odot (W - Q_b(W)) \right| \quad (25)$$

where $Q_b(W)$ represents the b -bit quantized weight matrix, and $\odot$ denotes the element-wise multiplication.

To obtain the layer-level sensitivity score, LLM-MQ computes the average sensitivity across all weight matrices within the layer. Thus, the sensitivity score for layer l at bit-width b is defined as:

$$\mathcal{S}_{\text{LLM-MQ}}^{(l)}(b) = \frac{1}{|\mathcal{W}_l|} \sum_{W \in \mathcal{W}_l} s_b(W) \quad (26)$$

A larger $\mathcal{S}_{\text{LLM-MQ}}^{(l)}(b)$ score implies that quantizing this layer induces a more significant deviation in

the loss function, indicating that the layer is highly sensitive and should be preserved in higher precision.

LieQ. LieQ (Layer-wise information effectiveness Quantization) estimates layer sensitivity by analyzing the representational compactness, which measures the concentration of information in the layer’s output representations. For a trained weight matrix $W \in \mathcal{W}_l$ and input hidden states h , it computes the projected activation $Z = Wh$, alongside a baseline $\tilde{Z} = \tilde{W}h$ generated by an untrained weight matrix $\tilde{W}$.

LieQ performs Singular Value Decomposition (SVD) on the representations to obtain the normalized singular values $p_k = \frac{\sigma_k}{\sum_j \sigma_j}$. The representational compactness is then defined as the exponential of the Shannon entropy of this energy distribution:

$$\text{Compact}(Z) = \exp \left(- \sum_k p_k \log p_k \right) \quad (27)$$

To quantify the structural shift induced by training, LieQ calculates the relative reduction in compactness for each weight matrix. The layer-level sensitivity score $\mathcal{S}_{\text{LieQ}}^{(l)}$ is formulated by averaging these relative changes across all weight matrices (e.g., query, key, and value projections) within the layer:

$$\mathcal{S}_{\text{LieQ}}^{(l)} = \frac{1}{|\mathcal{W}_l|} \sum_{W \in \mathcal{W}_l} \frac{\text{Compact}(\tilde{Z}) - \text{Compact}(Z)}{\text{Compact}(\tilde{Z})} \quad (28)$$

A higher $\mathcal{S}_{\text{LieQ}}^{(l)}$ indicates that the layer has significantly concentrated its information into lower-rank manifolds during training, making its function highly irreplaceable and sensitive to quantization noise.

E.3 Slim-LLM

Slim-LLM (Salience-Driven Mixed-Precision Quantization) achieves low-bit compression by dynamically allocating group-wise bit-widths and calibrating quantizer parameters, utilizing GPTQ as its foundational backend. For a weight matrix $W \in \mathcal{W}_l$, it evaluates the salience of each weight element based on its magnitude and the input activation. The element-wise salience is mathematically approximated as $\delta_{i,j} \approx (w_{i,j} \cdot \|x_j\|_2)^2$, where x_j represents the corresponding activation channel.

To perform Salience-Determined Bit Allocation (SBA), Slim-LLM computes the average salience

for each structured group within the weight matrix. It then assigns different bit-widths (e.g., 2-bit or 3-bit) to these groups by minimizing the Kullback-Leibler (KL) divergence between the outputs of the full-precision and quantized layers, while strictly maintaining the target average bit-width budget.

Furthermore, to protect highly critical weights within each group, SliM-LLM introduces Saliency-Weighted Quantizer Calibration (SQC). It isolates locally salient weights—typically outliers beyond a 3σ threshold and calibrates the quantizer’s scale and zero-point by minimizing a weighted reconstruction loss. Finally, the optimized group quantizers operate within the GPTQ framework, which sequentially quantizes the weights and compensates for the residual quantization errors using the inverse Hessian matrix.

F Details of PTQ Backends

HQQ. Half-Quadratic Quantization (HQQ) is a highly efficient, calibration-free quantization method that optimizes the quantization parameters by minimizing a sparsity-promoting $\ell_{p<1}$ -norm of the quantization error. For a full-precision weight matrix $W \in \mathcal{W}_l$, HQQ aims to find the optimal scaling factor s and zero-point z by formulating the objective as:

$$\arg \min_{z,s} \phi(W - Q_{z,s}^{-1}(Q_{z,s}(W))) \quad (29)$$

where $\phi(\cdot)$ denotes the $\ell_{p<1}$ -norm loss. Unlike the conventional squared error, this non-convex loss function effectively models and captures the heavy-tailed distribution of weight outliers through a hyper-Laplacian distribution. The quantization and de-quantization operators are defined as $Q_{z,s}(W) = \text{round}(W/s + z)$ and $Q_{z,s}^{-1}(W_q) = s(W_q - z)$, respectively.

To tackle this non-convex optimization problem, HQQ employs a Half-Quadratic solver by introducing an auxiliary continuous variable W_e . By fixing the scaling parameter s , the optimization of z is reformulated as:

$$\arg \min_{z,W_e} \phi(W_e) + \frac{\beta}{2} \|W_e - (W - Q_z^{-1}(Q_z(W)))\|_2^2 \quad (30)$$

where β is a penalty parameter. This formulation allows the main problem to be decomposed into two tractable sub-problems that are efficiently solved via alternate optimization. Because HQQ relies exclusively on the weights without requiring any calibration data, it successfully avoids potential

over-fitting and serves as an architecture-agnostic PTQ backend.

GPTQ. GPTQ is a highly efficient calibration-based quantization method that builds upon the Optimal Brain Quantizer (OBQ) framework. For a full-precision weight matrix $W \in \mathcal{W}_l$ and calibration input activations X , GPTQ formulates the quantization process as a layer-wise reconstruction problem. It aims to minimize the output squared error between the full-precision and quantized layer:

$$\arg \min_{\hat{W}} \|WX - \hat{W}X\|_2^2 \quad (31)$$

where $\hat{W}$ represents the quantized weight matrix.

To mitigate the performance degradation caused by precision reduction, GPTQ actively compensates for the quantization error of each weight. It does so by updating the remaining unquantized weights based on the inverse Hessian matrix, H^{-1} , of the input activations. While the original OBQ method quantizes weights individually and strictly requires recomputing H^{-1} at each step, GPTQ innovates by quantizing the weights column-wise.

This column-wise quantization strategy successfully eliminates the expensive repeated calculations of the inverse Hessian matrix, making the optimization highly tractable. Consequently, GPTQ can smoothly scale to massive language models with hundreds of billions of parameters, while concurrently accelerating the generation speed through its heavily optimized mixed-precision matrix multiplication kernels.

G Additional Results in Further Analysis

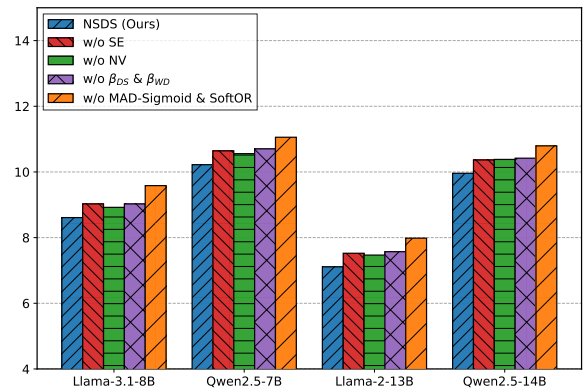


Figure 8: Average perplexity results of the ablation analysis on NSDS across all language modeling benchmarks. “w/o” denotes the exclusion of a specific component from NSDS during layer sensitivity estimation.

	Language Reasoning ($\uparrow$)						Language Modeling ($\downarrow$)	
	ARC-C	Hellaswag	PIQA	BoolQ	Winogrande	TruthfulQA	Wikitext-2	C4
Llama-2-13B								
FP16	59.47	82.22	79.05	80.55	76.16	25.95	4.88	6.47
MSE	46.71	73.27	73.35	72.58	71.63	21.96	8.55	10.21
EWQ	48.14	75.11	75.38	75.22	73.43	22.85	6.77	8.43
ZD	47.89	74.34	74.83	74.01	72.69	22.23	7.08	8.89
KurtBoost	48.54	75.95	75.97	75.91	73.12	23.04	6.62	8.41
NSDS (Ours)	49.74	76.86	76.78	76.75	73.96	23.86	6.23	7.99
Qwen2.5-14B								
FP16	67.24	84.34	81.07	85.23	81.45	40.15	5.29	9.15
MSE	55.94	75.37	74.81	76.94	73.05	29.63	8.93	12.31
EWQ	55.03	74.52	73.95	75.69	72.58	28.44	9.37	12.88
ZD	57.38	76.49	76.32	77.97	73.91	30.45	8.54	11.97
Kurtosis	56.76	76.58	75.49	78.13	73.06	30.77	8.46	12.36
NSDS (Ours)	58.27	77.34	76.78	78.62	74.28	31.58	8.28	11.65

Table 3: Results of baseline methods and our NSDS on Llama-2-13B and Qwen2.5-14B, evaluated on six language reasoning and two language modeling benchmarks.

Language Modeling Results of Ablation Analysis. To comprehensively understand the impact of each component within NSDS, we further evaluate the ablation variants on language modeling benchmarks (WikiText-2 and C4). As illustrated in Fig. 8, the average perplexity results exhibit a degradation trend that aligns with the language reasoning accuracy shown in Fig. 4. Specifically, the “w/o MAD-Sigmoid & Soft-OR” variant consistently yields the most severe PPL increase, reaffirming that proper normalization and aggregation are crucial for obtaining the final layer sensitivity scores. Similarly, omitting SE or NV causes a noticeable PPL surge. The consistent deterioration across all ablation settings on both reasoning and language modeling tasks validates the necessity of our joint numerical-structural design and the refined aggregation strategy.

Language Modeling Results of Calibration-based Methods. To further validate the robustness of our data-free NSDS framework, we compare its average PPL on WikiText-2 and C4 with the calibration-based methods in Fig. 9. The results align with the accuracy results in Fig. 5. Specifically, NSDS consistently maintains a top-two ranking with low PPL across all four LLMs. Meanwhile, calibration-based baselines such as LIM and LLM-MQ show fluctuating PPL scores that depend on the specific target model.

Language Modeling Results of Integration with Other PTQ Backends. We report the average PPL results of NSDS integrated with different PTQ backends, along with the SliM-LLM baseline in Fig. 10. The results exhibit a trend similar to the ac-

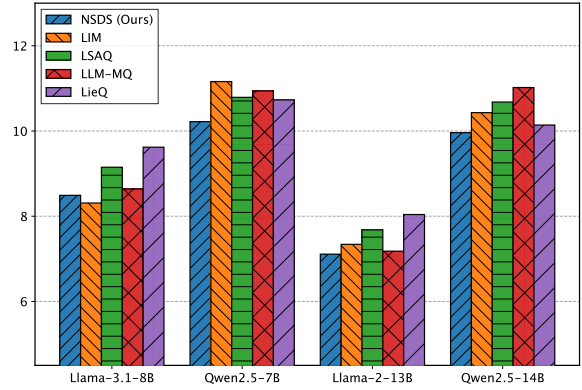


Figure 9: Average perplexity comparison between NSDS framework and calibration-based baselines across generic reasoning benchmarks.

curacy in Fig. 6. Upgrading the backend to GPTQ yields lower PPL across all evaluated models. This indicates that the orthogonality of NSDS holds true for language modeling tasks as well, where upgrading to a stronger PTQ backend yields better performance.

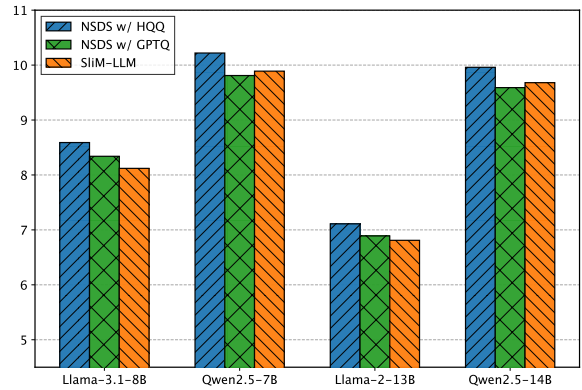


Figure 10: Average perplexity on language modeling benchmarks when integrating NSDS with different PTQ backends, compared against the SliM-LLM.