# NOT ALL CALIBRATION DATA ARE EQUAL: OPTIMIZING CALIBRATION DATA FOR POST-TRAINING QUANTIZATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

An often overlooked source of performance variation in post-training quantization (PTQ) is the calibration data used to estimate quantization-related statistics. Even under the same model and quantization configuration, different calibration data can lead to substantially different performance. We propose CalibSelect, a calibration data assessment and selection framework based on model responses. CalibSelect introduces two metrics: the calibration activation range score (CARS), which measures the spread of activation responses while mitigating the influence of extreme samples, and the layer sensitivity discrimination score (LSDS), which measures how clearly and stably calibration data reveal differences in quantization sensitivity across layers. Based on these metrics, CalibSelect adopts a two-stage selection strategy that first identifies several promising calibration data sets from the candidate data pool and then refines them through indicator-guided local optimization. CalibSelect is compatible with different PTQ methods and does not modify their underlying algorithms. Experiments on Qwen2.5-7B and LLaMA3.1-8B using AWQ and AutoRound show that CalibSelect consistently outperforms random sampling and existing selection methods. The largest reduction in quantized model perplexity reaches 8.23%.

## 1 INTRODUCTION

Recent large language models (LLMs), including GPT-5 (OpenAI, 2025), Kimi K2.5 (Kimi Team, 2026), and Gemma 4 (Gemma Team, 2026), have advanced language understanding, text generation, and complex reasoning. Their storage, computation, and memory demands nevertheless remain a deployment barrier under resource and cost constraints (Lin et al., 2025; Sheng et al., 2023). Post-training quantization (PTQ) reduces these costs by converting pretrained models to low-bit representations without full retraining (Xia et al., 2024). Methods such as GPTQ (Frantar et al., 2023), AWQ (Lin et al., 2024b), SmoothQuant (Xiao et al., 2023), OmniQuant (Shao et al., 2024), and FlatQuant (Sun et al., 2025) preserve model performance through weight reconstruction, salient-weight protection, scaling, and quantization-parameter optimization. Despite their different quantization objectives and optimization strategies, they share a dependence on calibration data.

Calibration data provide the model responses used to estimate quantization parameters and guide reconstruction based optimization (Williams & Aletras, 2024; Nagel et al., 2020; Li et al., 2021). For weight quantization, calibration data provide activation information for weight reconstruction, adaptive rounding, salient-weight identification, and sensitivity analysis (Nagel et al., 2020; Li et al., 2021; Frantar et al., 2023; Lin et al., 2024b). When activations are also quantized, they additionally determine activation range and outlier estimates used for clipping, shifting, and scaling (Yao et al., 2022; Wei et al., 2023; Xiao et al., 2023; Shao et al., 2024). The choice of calibration data therefore affects the quantization parameters and optimization signals on which PTQ depends.

Prior studies show that the size, source, and sequence length of calibration data can affect PTQ performance (Wu et al., 2023; Williams & Aletras, 2024; Williams et al., 2025). Increasing the calibration budget can improve performance, but the gains may diminish as more samples are added (Hubara et al., 2021). These findings reveal the relationship between basic calibration data



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

properties and quantization performance. However, it remains difficult to assess how calibration data influence the quantization process relying solely on external data properties.

Recent studies turn to model activations to guide calibration data selection. CoLA (He et al., 2025) selects representative samples near cluster centers through activation-based clustering. ACDM (Maoxiong et al., 2026) selects samples with layer-wise activation statistics closely similar to those of task-specific reference data. These studies demonstrate the value of model activations for calibration data selection beyond basic data properties. A complementary question is whether calibration data can be assessed based on the properties that directly affect the quantization process.

***This motivates our central question: Can we identify high-quality calibration data before PTQ by assessing their quantization-relevant characteristics?***

Calibration data directly influence quantization by providing the activation responses used for parameter estimation and optimization. Even when only weights are quantized, the resulting output error depends on the model activations. Calibration activations therefore inform salient-weight identification and channel-wise scaling. A calibration data set that captures only a narrow range of activation responses may provide incomplete information about the effects of weight quantization on model outputs. For methods that also quantize activations, calibration data additionally inform activation range and scale estimation (Jacob et al., 2018). An overly narrow calibration range may fail to cover activation values encountered after PTQ, causing clipping and increased quantization error. Conversely, a range expanded by a few extreme activation values may produce a larger quantization step size, reducing the resolution for typical activations (Xiao et al., 2023). Effective calibration data should therefore cover a broad range of representative activation responses without being dominated by extreme samples.

Furthermore, Transformer layers can exhibit different quantization error, even under the same 4-bit configuration (Wang et al., 2026). These differences arise from variations in weight distributions, input activations, and error propagation. Calibration data play a key role in capturing these differences and guiding quantization optimization. For example, reconstruction-based PTQ uses calibration data to guide the weight reconstruction by minimizing the output discrepancy between the original and quantized models (Frantar et al., 2023). If calibration data fail to capture the influence of important weights on model outputs, their associated errors may be underrepresented in the reconstruction objective. As a result, these errors may remain insufficiently corrected, leading to larger output deviations. This is similar to gradient-based optimization, where unrepresentative training data may produce gradients that fail to reflect the desired optimization direction. Calibration data should therefore reveal differences in quantization sensitivity across layers to provide more informative signals for subsequent quantization. Another important consideration is the stability of sensitivity estimates, which should not be dominated by a few individual samples. Effective calibration data should therefore reveal layer-wise differences in quantization sensitivity while providing stable sensitivity estimates.

Based on this motivation, we introduce the calibration activation range score (CARS) and the layer sensitivity discrimination score (LSDS). First, CARS assesses activation response coverage by analyzing the mean absolute activation, root mean square, and high-percentile activation values of calibration data across Transformer layers. Quantile-based activation ranges are then computed to measure activation coverage while reducing the influence of extreme samples. Second, LSDS evaluates how clearly and stably calibration data capture layer-wise differences in quantization sensitivity. Taking Transformer layers as the evaluation unit, LSDS measures the output error induced by quantizing individual layers and uses resampling to assess the stability of sensitivity estimates. The evaluation can be extended to finer granularity, such as the channel level, to accommodate different PTQ methods.

We further propose CalibSelect, a calibration data assessment and selection framework that integrates CARS and LSDS to guide data selection for PTQ. Evaluating all possible calibration data is computationally expensive, particularly because LSDS requires fake quantization to measure layer-wise sensitivity. CalibSelect therefore employs a two-stage search strategy to identify high-quality calibration data. In the first stage, exploration, CalibSelect randomly samples multiple candidate calibration data sets from the data pool and evaluates each candidate set using CARS and LSDS. The two metrics are robustly normalized to account for differences in scale and distribution, then combined into a weighted quality score. CalibSelect ranks the candidate calibration sets by their

weighted quality scores and retains the top-$k$ candidates as starting points for the second stage, focusing the search on promising regions of the candidate data pool. In the second stage, local refinement, each retained candidate calibration set is independently optimized through iterative sample replacement. At each iteration, a randomly selected sample in the current candidate calibration set is replaced with an unselected sample from the data pool. The new candidate calibration set is evaluated using the joint quality score, and the replacement is accepted only if the score improves. The search terminates when no improvement is observed for a predefined number of consecutive iterations or the maximum number of iterations is reached. The highest-scoring candidate calibration set across all search trajectories is selected as the final calibration data. This two-stage strategy combines broad exploration with indicator-guided local refinement, avoiding exhaustive search while optimizing calibration data. CalibSelect operates independently of PTQ algorithms and requires no changes to their quantization objectives or optimization procedures.

The main contributions are summarized as follows:

- We propose CalibSelect, a PTQ-oriented calibration data selection framework that combines quantization-relevant metrics with a two-stage search strategy to identify high-quality calibration data.
- We introduce two metrics, CARS and LSDS, to assess calibration data quality through activation response coverage and stable layer-wise quantization sensitivity discrimination.
- CalibSelect reduces perplexity (PPL) by up to 7.26% compared with CoLA and 8.23% compared with ACDM across the evaluated LLMs and PTQ methods, while improving downstream task accuracy after PTQ by up to 0.46 percentage points against random sampling.

## 2 Related Work

**LLM Post-Training Quantization** LLM PTQ methods preserve performance at low precision through weight reconstruction, activation-aware scaling, clipping, and transformations of weights and activations. GPTQ (Frantar et al., 2023) uses activations for layer-wise weight reconstruction, while AWQ (Lin et al., 2024b) uses activation information to identify salient weights and determine channel-wise scaling. AutoRound (Cheng et al., 2025) optimizes weight rounding using calibration data. SmoothQuant (Xiao et al., 2023) mitigates activation outliers through equivalent channel-wise scaling. OmniQuant (Shao et al., 2024) and AffineQuant (Ma et al., 2024) learn clipping parameters or equivalent transformations from calibration data. DuQuant (Lin et al., 2024a), SpinQuant (Liu et al., 2025), and FlatQuant (Sun et al., 2025) reshape weight and activation distributions through permutations, rotations, or learnable transformations. These approaches primarily optimize the quantization procedure while treating the calibration data as a given input. CalibSelect complements this line of work by optimizing the calibration data supplied to that procedure.

**Calibration Data for Post-Training Quantization** Calibration data affect PTQ performance through several factors. Williams & Aletras (2024) analyze calibration-corpus source and random sampling, while Hubara et al. (2021) study calibration-set size. Other studies examine sequence length (Oh & Oh, 2025) and language diversity in multilingual quantization (Chimoto et al., 2026). Williams et al. (2025) and Xiao et al. (2026) explore self-generated calibration data. Model-internal responses also provide criteria for selecting samples from existing corpora: Maoxiong et al. (2026) use activation-centroid distance to select representative samples, and He et al. (2025) jointly consider representativeness and diversity in activation space. Yang et al. (2026) study target-aware selection to preserve uncertainty behavior after quantization.

## 3 Method

### 3.1 Problem Formulation and Framework Overview

**Problem Definition** Given a candidate data pool $D = \{d_i\}_{i=1}^{N}$ of $N$ samples and a calibration budget $m$ ($m < N$), our goal is to select $m$ samples from $D$ as the calibration data $S$ for PTQ, where $|S| = m$. CalibSelect evaluates each candidate calibration data set as a whole using the

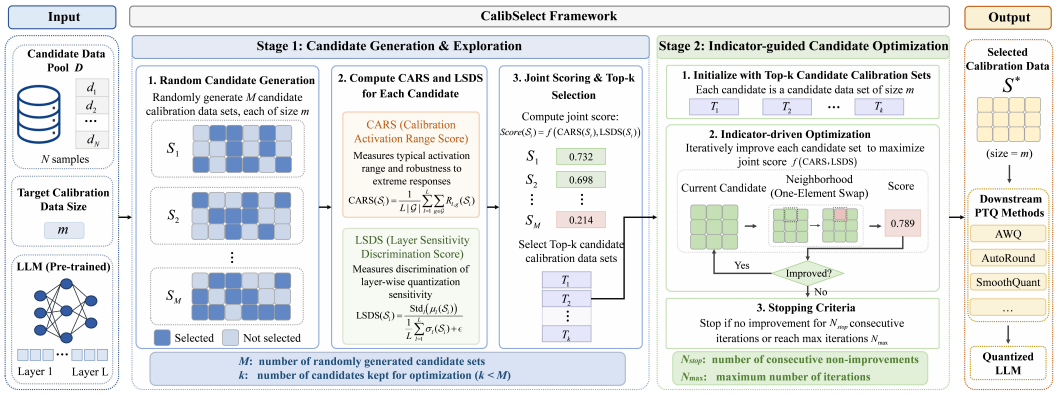



Figure 1: Overview of CalibSelect. Stage 1 generates and evaluates multiple candidate calibration sets using CARS and LSDS. Stage 2 further optimizes these candidate sets through indicator-guided local optimization to obtain the final calibration data.

model responses. Its objective is to identify a calibration data set that provide informative calibration signals for PTQ under the given budget.

**Framework Overview** As shown in Fig. 1, CalibSelect consists of two stages. In *candidate generation & exploration*, we randomly sample $M$ candidate calibration data sets $\{S_i\}_{i=1}^{M}$ from $D$, each containing $m$ samples. Each candidate set is evaluated using CARS and LSDS. Since the two metrics differ in scale and distribution, their values are robustly normalized and then combined into a joint score $Score(S_i)$. The top-$k$ candidates $\{T_j\}_{j=1}^{k}$, with $k < M$, are retained as the starting points for the second stage. In *indicator-guided candidate optimization*, each retained candidate set is optimized independently. At each iteration, we replace one selected sample with an unselected sample from $D$ and accept the replacement only if it increases the joint score. Each search stops after $N_{stop}$ consecutive iterations without improvement or $N_{max}$ iterations in total. We return the highest-scoring candidate $S^*$ across all searches. Through this two-stage strategy, CalibSelect first identifies promising calibration candidates through broad screening and then further improves them through local optimization guided by CARS and LSDS. The detailed algorithm is provided in Appendix A.

### 3.2 CARS Computation

As discussed in the Introduction, calibration data should capture a broad range of activation responses without being dominated by a few extreme samples. To assess this property, we propose CARS, which characterizes the activation response coverage of a calibration data set using quantile-based statistics. Fig. 2 illustrates the design rationale of CARS, and its computation is detailed below.

Let $S = \{d_i\}_{i=1}^{m}$ be a calibration data set, where $d_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,B_i})$ contains $B_i$ tokens. For each sample $d_i \in S$, we feed it into the full-precision model and record the input hidden states $H_{i,l} \in \mathbb{R}^{B_i \times C}$ of each layer $l$, where $C$ denotes the hidden dimension. We then collect the hidden states into an activation set $\mathcal{A}_{i,l} = \{H_{i,l,b,c},\ b = 1, \ldots, B_i,\ c = 1, \ldots, C\}$ to characterize the distribution of activation values. Since $\mathcal{A}_{i,l}$ contains a large number of activation values, directly comparing raw activation sets across samples is inconvenient. We therefore summarize $\mathcal{A}_{i,l}$ using four statistics: the mean absolute activation, root-mean-square activation, 95th percentile, and 99th percentile of absolute activation values:

$$a_{i,l}^{(1)} = \frac{1}{B_i C} \sum_{a \in \mathcal{A}_{i,l}} |a| \qquad a_{i,l}^{(2)} = \sqrt{\frac{1}{B_i C} \sum_{a \in \mathcal{A}_{i,l}} a^2}$$
$$a_{i,l}^{(3)} = Q_{0.95}\left(\{|a| : a \in \mathcal{A}_{i,l}\}\right) \qquad a_{i,l}^{(4)} = Q_{0.99}\left(\{|a| : a \in \mathcal{A}_{i,l}\}\right) \tag{1}$$

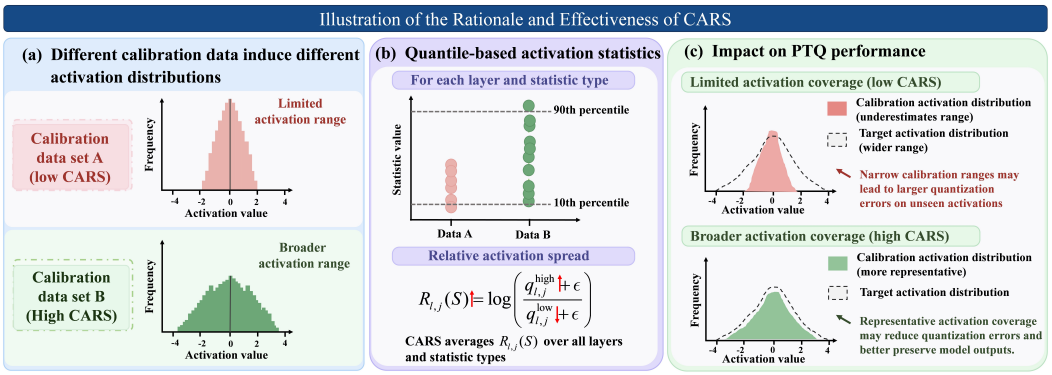



Figure 2: Illustration of the design rationale and effectiveness of CARS.

where $Q_\alpha(\cdot)$ denotes the $\alpha$-quantile. These statistics form the vector $\mathbf{a}_{i,l} = \left[a_{i,l}^{(1)}, a_{i,l}^{(2)}, a_{i,l}^{(3)}, a_{i,l}^{(4)}\right]$ for sample $d_i$ at layer $l$.

For $m$ samples and $L$ layers, the activation tensor is:

$$\mathbf{A}_S = \left[a_{i,l}^{(j)}\right] \in \mathbb{R}^{m\times L\times J},\ J = 4. \tag{2}$$

Mean absolute activation and root-mean-square characterize the general activation scale, while the 95th and 99th percentiles capture the upper-tail behavior of the activation. For each layer $l$ and statistic $j$, we take quantiles across the calibration data set:

$$q_{l,j}^{\text{low}} = Q_{0.10}\left(\left\{a_{i,l}^{(j)}\right\}_{i=1}^{m}\right) \qquad q_{l,j}^{\text{high}} = Q_{0.90}\left(\left\{a_{i,l}^{(j)}\right\}_{i=1}^{m}\right) \tag{3}$$

The interval $I_{l,j}(S) = [q_{l,j}^{\text{low}}, q_{l,j}^{\text{high}}]$ defines the quantile coverage range of calibration activation statistics. We measure its logarithmic span, with $\epsilon > 0$ for numerical stability:

$$R_{l,j}(\mathcal{S}) = \log\left(\frac{q_{l,j}^{\text{high}} + \epsilon}{q_{l,j}^{\text{low}} + \epsilon}\right) \tag{4}$$

The CARS score averages these logarithmic spans over layers and statistics:

$$\text{CARS}(S) = \frac{1}{LJ}\sum_{l=1}^{L}\sum_{j=1}^{J} R_{l,j}(S) = \frac{1}{4L}\sum_{l=1}^{L}\sum_{j=1}^{4}\log\left(\frac{q_{l,j}^{\text{high}} + \epsilon}{q_{l,j}^{\text{low}} + \epsilon}\right) \tag{5}$$

### 3.3 LSDS Computation

As discussed in the Introduction, high-quality calibration data should effectively reveal differences in quantization sensitivity across Transformer layers while providing stable sensitivity estimates. To evaluate these properties, we propose LSDS, which characterizes the layer-wise sensitivity of a calibration data set through a controlled quantization operation. Fig. 3 illustrates the design of LSDS, and its computation is detailed below.

Let $S = \{d_i\}_{i=1}^m$ be a calibration data set, where $d_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,B_i})$ contains $B_i$ tokens. We first compute each sample's average negative log-likelihood (NLL) using the full-precision model:

$$n_{i,full} = -\frac{1}{B_i - 1}\sum_{b=2}^{B_i}\log p_{\theta_{full}}\left(x_{i,b} \mid x_{i,<b}\right) \tag{6}$$

where $\theta_{\text{full}}$ denotes the parameters of the original full-precision model.

To measure the error introduced by quantizing layer $l$, we fake-quantize only that layer's weight matrices and keep all other layers in full precision. Identifying the sensitivity of individual layers

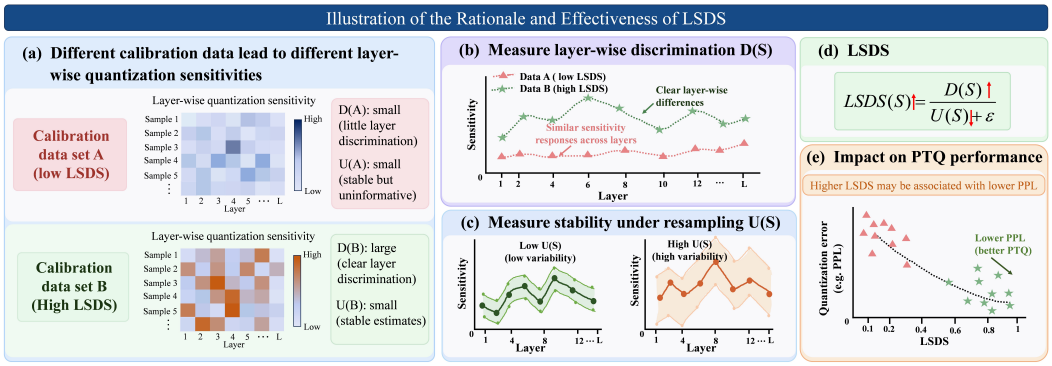



Figure 3: Illustration of the design rationale and effectiveness of LSDS.

becomes difficult when multiple layers are quantized simultaneously, as their quantization errors may propagate and interact across the network. The purpose of this procedure is to observe the local performance change caused by quantizing a single Transformer layer, but not to perform the final PTQ. A standardized fake-quantization operator is used to assess layer-wise sensitivity without requiring an exact reproduction of the PTQ algorithm. Detailed implementation of this operator is provided in Appendix B. Under the resulting model, denoted by $\theta^{l}_{qua}$, the average NLL is:

$$n^{(l)}_{i,qua} = -\frac{1}{B_i - 1} \sum_{b=2}^{B_i} \log p_{\theta^{(l)}_{qua}} \left( x_{i,b} \mid x_{i,<b} \right) \tag{7}$$

We define the sensitivity of layer $l$ on sample $d_i$ as the nonnegative NLL increase:

$$e_{i,l} = \max \left\{ 0,\, n^{(l)}_{i,\text{qua}} - n_{i,\text{full}} \right\} \tag{8}$$

For $m$ samples and $L$ evaluated layers, the sensitivity matrix is defined as $\mathbf{E}_S = [e_{i,l}] \in \mathbb{R}^{m \times L}$, where rows correspond to samples and columns to layers. Since each $e_{i,l}$ depends only on the corresponding sample, model, and fake-quantization operator, sensitivity responses can be reused across different candidate calibration data sets. During the two-stage search, we compute and cache the sensitivity responses of newly encountered samples, while reusing previously cached responses for subsequent candidate set evaluations. For any calibration data set $S$, $\mathbf{E}_S$ is constructed by retrieving the corresponding cached responses and computing only those that are not yet available. This incremental caching strategy avoids redundant model forward passes during candidate set exploration and local optimization, reducing the computational overhead of calibration data selection.

Based on $\mathbf{E}_S$, we next estimate the quantization sensitivity of each Transformer layer. A straightforward approach is to average the sensitivity values of all $m$ samples for each layer. However, a layer's mean sensitivity can be dominated by a few samples, making the estimate unstable when the selected sample changes. We therefore quantify this sensitivity through $P$ resampling trials (Meinshausen & Bühlmann, 2010). A fixed random seed makes scoring deterministic during search. Trial $p$ draws $r$ samples, forming $S^p \subseteq S$, and computes $s^p_l = \frac{1}{r} \sum_{d_i \in S^p} e_{i,l}$. The mean and standard deviation over these trials are:

$$\mu_l = \frac{1}{P} \sum_{p=1}^{P} s^p_l \qquad \sigma_l = \sqrt{\frac{1}{P-1} \sum_{p=1}^{P} \left( s^p_l - \mu_l \right)^2} \tag{9}$$

Here, $\mu_l$ is the mean sensitivity estimate, while $\sigma_l$ measures the variation of this estimate across different resampled sets. A smaller $\sigma_l$ indicates that the estimated sensitivity of layer $l$ is less sensitive to which samples in $S$ are included in each resampled set, and is therefore more stable.

Based on these statistics, LSDS is defined as:

$$\text{LSDS}(S) = \frac{D(S)}{U(S) + \epsilon} = \frac{\text{Std}_l \left( \mu_l \right)}{\frac{1}{L} \sum_{l=1}^{L} \sigma_l + \epsilon} \tag{10}$$

A larger $D(S)$ indicates that the calibration data reveal more distinguishable layer-wise quantization errors, while a smaller $U(S)$ indicates that these responses are less dependent on the particular samples selected. By combining these properties, LSDS favors calibration data that provide clear and stable layer-wise quantization sensitivity.

### 3.4 Joint Calibration Quality Score

We normalize the two metrics using the $M$ candidate calibration data sets generated in the first stage, denoted by $\{S_j\}_{j=1}^{M}$. For each metric $g \in \{\mathrm{CARS}, \mathrm{LSDS}\}$, define:

$$c_g = \mathrm{Median}\left(\{g(S_j)\}_{j=1}^{M}\right) \qquad z_g = \frac{Q_{0.75}(g) - Q_{0.25}(g)}{\tau} \tag{11}$$

where $\tau > 0$ is a fixed scaling factor. $Q_\alpha(g)$ is the $\alpha$-quantile of $\{g(S_j)\}_{j=1}^{M}$. The $c_g$ and $z_g$ provide center and scale estimates that are less sensitive to extreme scores than the mean and standard deviation. Using $c_g$ and $z_g$, we map each metric to $(0, 1)$ by:

$$\widetilde{g}(S) = \frac{1}{1 + \exp\left[-\frac{g(S) - c_g}{z_g + \epsilon}\right]} \tag{12}$$

The center and scale are estimated once and held fixed in the second stage, so all search trajectories use the same transformation for a given budget and configuration. The joint score of calibration data set $S$ is then defined as:

$$Score(S) = w_C \widetilde{\mathrm{CARS}}(S) + w_L \widetilde{\mathrm{LSDS}}(S) \tag{13}$$

where $w_C + w_L = 1$ and, unless specified otherwise, $w_C = w_L = 0.5$. The joint score ranks candidates in the first stage and guides sample replacements in the second stage.

## 4 Experiments

### 4.1 Experimental Setup

We evaluate CalibSelect on Qwen2.5-7B (Yang et al., 2024) and LLaMA3.1-8B (Grattafiori et al., 2024) using two representative PTQ methods, AWQ (Lin et al., 2024b) and AutoRound (Cheng et al., 2025). AWQ is a widely used PTQ method, while AutoRound represents a more recent approach.

We construct candidate data pools from general text in WikiText (Merity et al., 2017), code in CodeParrot (von Werra, 2021) and Python-codes-25K (flytech, 2024), mathematics in GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), and commonsense reasoning in ARC (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018), and HellaSwag (Zellers et al., 2019). The resulting candidate data pools are Code, Math, Common, and Mixed; the Mixed pool combines all four data categories. All quantized models are evaluated on WikiText. This evaluation measures the effect of calibration data source on language modeling performance for LLMs after PTQ. It does not directly assess preservation of domain-specific capabilities. Calibration and evaluation samples from WikiText do not overlap.

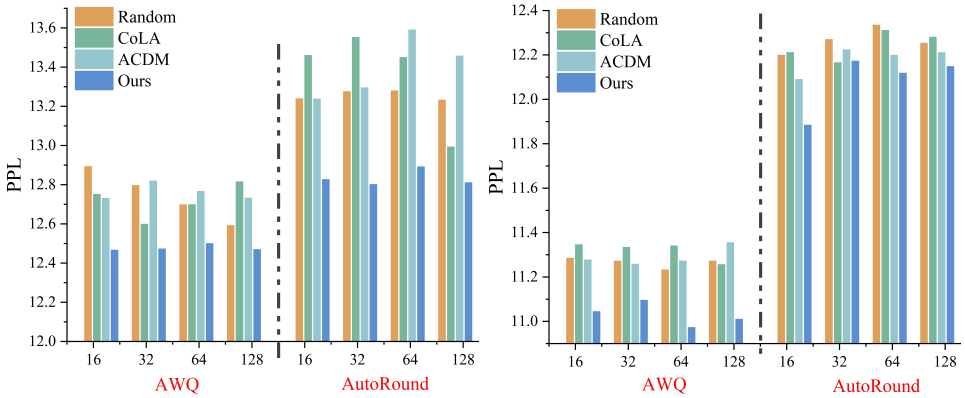
We compare CalibSelect with random sampling (Random), CoLA (He et al., 2025), and ACDM (Maoxiong et al., 2026) at budgets $m \in \{16, 32, 64, 128\}$, where $m$ denotes the number of samples in each calibration data set. Implementation details are provided in Appendix C.

### 4.2 Overall Performance Comparison

Table 1 reports PPL averaged over the four calibration budgets of 16, 32, 64, and 128. Results for each budget are provided in Appendix D due to space limitations. Across different candidate data pools and quantization methods, CalibSelect achieves lower PPL than Random, CoLA, and ACDM in nearly all settings, demonstrating the effectiveness of the selected calibration data for PTQ. The exception is Qwen2.5-7B with AutoRound on the Code pool, where ACDM performs better. For Qwen2.5-7B with AWQ, PPL averaged across pools decreases to 12.4765, compared with 12.7442 for Random, 12.7154 for CoLA, and 12.7615 for ACDM: relative reductions of 2.10%, 1.88%, and

Table 1: PPL of different calibration data selection strategies. The reported PPL values are averaged over calibration budgets of 16, 32, 64, and 128. The best result is highlighted in bold.

| Model | Pool | AWQ | | | | AutoRound | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Random | CoLA | ACDM | Ours | Random | CoLA | ACDM | Ours |
| Qwen2.5-7B | Mixed | 12.7252 | 12.7362 | 12.8836 | **12.3919** | 13.4321 | 13.5880 | 13.6360 | **12.8788** |
| | Code | 12.6947 | 12.7009 | 12.7086 | **12.5374** | 13.1685 | 13.0322 | **12.8565** | 12.9477 |
| | Math | 12.6478 | 12.8308 | 12.5999 | **12.5191** | 13.2747 | 13.2410 | 13.3499 | **12.8976** |
| | Common | 12.9091 | 12.5938 | 12.8542 | **12.4578** | 13.1485 | 13.5908 | 13.7348 | **12.6041** |
| | *Avg.* | 12.7442 | 12.7154 | 12.7615 | **12.4765** | 13.2559 | 13.3630 | 13.3943 | **12.8320** |
| LLaMA3.1-8B | Mixed | 11.1950 | 11.1972 | 11.3061 | **10.9940** | 12.2941 | 12.2473 | 12.1584 | **12.1343** |
| | Code | 11.3221 | 11.3313 | 11.2803 | **11.0405** | 12.3056 | 12.2780 | 12.2265 | **12.1798** |
| | Math | 11.2577 | 11.4119 | 11.3045 | **11.0657** | 12.2598 | 12.2701 | 12.2272 | **12.1609** |
| | Common | 11.2845 | 11.3324 | 11.2695 | **11.0197** | 12.1955 | 12.1711 | 12.1094 | **11.8464** |
| | *Avg.* | 11.2648 | 11.3182 | 11.2901 | **11.0299** | 12.2637 | 12.2416 | 12.1803 | **12.0803** |




(a) Qwen2.5-7B. (b) LLaMA3.1-8B.

Figure 4: PPL under different calibration budgets for Qwen2.5-7B and LLaMA3.1-8B with AWQ and AutoRound. The PPL is averaged over the Mixed, Code, Math, and Common candidate pools.

2.23%. With AutoRound, the corresponding averages decrease from 13.2559, 13.3630, and 13.3943 to 12.8320, giving reductions of 3.20%, 3.97%, and 4.20%. Overall, the consistent improvements across different quantization methods and candidate data pools demonstrate the effectiveness of CalibSelect across the evaluated PTQ methods and data sources.

To further examine whether the performance gains depend on the calibration budget, Fig. 4 reports the results under different calibration budgets. For each budget, the reported PPL is averaged over the four candidate data pools, namely Mixed, Code, Math, and Common. CalibSelect already shows clear advantages at small budgets of 16 and 32, generally achieving lower PPL than Random, CoLA, and ACDM across both model families and quantization methods. This indicates that effective PTQ can be achieved with high-quality calibration data even when only a limited number of samples are available. As the calibration budget increases, the performance advantage of CalibSelect is largely maintained. These results further show that calibration data selection is particularly important under limited budgets, where the quality of the selected samples has a greater impact on PTQ. This advantage is particularly important for PTQ scenarios where collecting or processing a large amount of calibration data is undesirable.

### 4.3 Ablation Study

We evaluate three variants to isolate the roles of local refinement and the two metrics: *w/o Stage 2*, which selects from the initial candidate calibration data sets without local search; *LSDS only*; and *CARS only*. Table 2 reports the results on the Mixed pool, while results on the other pools are provided in Appendix E.

Table 2: Ablation study on the Mixed calibration data pool. The reported PPL values are averaged over calibration budgets of 16, 32, 64, and 128. Lower PPL indicates better performance.

| Model | Variant | AWQ | AutoRound |
|---|---|---|---|
| Qwen2.5-7B | *w/o Stage 2* | 12.6837 | 13.4077 |
| | *LSDS only* | 12.6012 | 13.2169 |
| | *CARS only* | 12.7450 | 13.1695 |
| | Ours | **12.3919** | **12.8788** |
| LLaMA3.1-8B | *w/o Stage 2* | 11.1096 | 12.1739 |
| | *LSDS only* | 11.2639 | 12.2043 |
| | *CARS only* | 11.3445 | 12.1892 |
| | Ours | **10.9940** | **12.1343** |

CalibSelect achieves the lowest PPL in all four model-quantizer settings on the Mixed pool. Removing the second stage increases PPL from 12.3919 to 12.6837 for Qwen2.5-7B with AWQ and from 12.8788 to 13.4077 with AutoRound. For LLaMA3.1-8B, the corresponding increases are from 10.9940 to 11.1096 and from 12.1343 to 12.1739. The ablation results confirm that the second stage improves the initially selected candidates through local optimization. The joint score also outperforms either metric alone in these settings. These comparisons support their combined use in the evaluated search procedure.

### 4.4 Downstream Task Performance

To assess the impact of calibration data selection on downstream task performance, we evaluate Qwen2.5-7B quantized with AWQ on the BoolQ benchmark, using accuracy as the evaluation metric. We compare CalibSelect against Random across four calibration budgets of 16, 32, 64, and 128. Both methods use the same Mixed candidate data pool and quantization configuration, differing only in the calibration data. Fig. 5 presents the resulting QA accuracy. CalibSelect consistently outperforms Random across all four calibration budgets, achieving accuracy gains of 0.31, 0.22, 0.36, and 0.46 percentage points, respectively. Averaged across the four budgets, CalibSelect improves accuracy from 85.79% to 86.13%, yielding a gain of 0.34 percentage points. These results demonstrate that CalibSelect improves not only PPL, but also downstream QA accuracy in the evaluated setting.

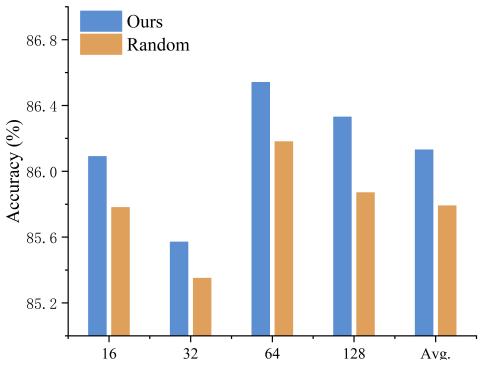



Figure 5: Downstream QA performance under different budgets for Qwen2.5-7B with AWQ using the Mixed pool.

### 4.5 Additional Analysis

Appendix F analyzes the relationship between CARS, LSDS, and PPL. Appendix G analyzes the computational complexity. Appendix H discusses the limitations of the proposed metrics.

## 5 Conclusion

We introduced CalibSelect, a framework for calibration data selection in PTQ. CalibSelect jointly considers layer-wise quantization sensitivity through LSDS and activation coverage through CARS. A two-stage search combines these metrics to screen and refine calibration data. Experiments across different models, quantization methods, candidate data pools, and calibration budgets demonstrate that CalibSelect consistently identifies calibration data that achieve lower PPL. These results highlight the importance of calibration data quality in PTQ. In future work, we will explore more effective quantization strategies and operator-level optimizations, with the goal of further improving both quantization accuracy and inference efficiency.

### AI USE STATEMENT

In this work, we used generative AI tools solely for language polishing, grammatical correction, and limited code cleanup and implementation refinement. We did not use generative AI tools for research idea generation, methodology design, or figure creation. Any AI-assisted code modifications were restricted to implementation-level improvements, such as code restructuring and debugging support, without changing the underlying methodology. All AI-assisted revisions were manually reviewed and verified by the authors to ensure technical accuracy and consistency with the original content.

### REPRODUCIBILITY STATEMENT

To support reproducibility, we provide detailed descriptions of CalibSelect in the main paper, including the definitions and computation procedures of LSDS and CARS, together with the two-stage calibration data selection strategy. The experimental setup specifies the models, quantization methods, calibration budgets, candidate data pools, and evaluation protocols used in our experiments. The Appendix provides additional implementation details, including an algorithmic description of CalibSelect.

### ACKNOWLEDGMENTS

Anonymous for double-blind review.

## REFERENCES

Wenhua Cheng, Weiwei Zhang, Heng Guo, Haihao Shen, and Zaner Ma. Signroundv2: Toward closing the performance gap in extremely low-bit post-training quantization for llms. *arXiv preprint arXiv:2512.04746*, 2025.

Everlyn Asiko Chimoto, Mostafa Elhoushi, and Bruce Bassett. Calibrating beyond english: Language diversity for better quantized multilingual llms. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4822–4838, 2026.

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.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

flytech. Python-codes-25k. Hugging Face Dataset, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. In *Proceedings of International Conference on Learning Representations*, 2023.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Bowei He, Lihao Yin, Huiling Zhen, Shuqi Liu, Han Wu, Xiaokun Zhang, Mingxuan Yuan, and Chen Ma. Preserving llm capabilities through calibration data curation: From analysis to optimization. In *Proceedings of Neural Information Processing Systems*, volume 38, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Accurate post training quantization with small calibration sets. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 4466–4475, 2021.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Kimi Team. Kimi k2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations*, 2021.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. Duquant: Distributing outliers via dual transformation makes stronger quantized llms. In *Advances in Neural Information Processing Systems*, volume 37, pp. 87766–87800, 2024a.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 87–100, 2024b.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. In *Proceedings of Machine Learning and Systems*, volume 7, 2025.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations. In *International Conference on Learning Representations*, 2025.

Yuexiao Ma, Huixia Li, Xiawu Zheng, Feng Ling, Xuefeng Xiao, Rui Wang, Shilei Wen, Fei Chao, and Rongrong Ji. Affinequant: Affine transformation quantization for large language models. In *International Conference on Learning Representations*, 2024.

Maoxiong, Benben Hai, Anzheng Wang, Xiangdong Liu, Lingwei Meng, Tingting Wang, Shuli Zheng, Yixin Zhou, and Haibiao Chen. Understanding and selecting calibration data for llm quantization: From sensitivity analysis to activation-based curation. In *The 1st International Workshop on Sustainable and Efficient Language, Vision, and Action Models*, 2026.

Nicolai Meinshausen and Peter Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 7197–7206, 2020.

Jaehoon Oh and Dokwan Oh. Beyond fixed-length calibration for post-training compression of llms. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 19355–19366, 2025.

OpenAI. Gpt-5 system card. *arXiv preprint arXiv:2601.03267v1*, 2025.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Qiao Yu, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. In *Proceedings of International Conference on Learning Representations*, pp. 45472–45496, 2024.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Re, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single gpu. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 31094–31116, 2023.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, Xin Jiang, Wulong Liu, and Jun Yao. Flatquant: Flatness matters for llm quantization. In *Proceedings of International Conference on Machine Learning*, 2025.

Leandro von Werra. Codeparrot dataset. Hugging Face Dataset, 2021.

Shigeng Wang, Chao Li, Yangyuxuan Kang, Jiawei Fan, Zhonghong Ou, and Anbang Yao. Sliderquant: Accurate post-training quantization for llms. *Proceedings of the International Conference on Learning Representations*, pp. 108556–108585, 2026.

Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. Outlier suppression+: Accurate quantization of large language models by equivalent and effective shifting and scaling. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1648–1665, 2023.

Miles Williams and Nikolaos Aletras. 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)*, pp. 10100–10118, 2024.

Miles Williams, George Chrysostomou, and Nikolaos Aletras. Self-calibration for language model quantization and pruning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 10149–10167, 2025.

Xiaoxia Wu, Haojun Xia, Stephen Youn, Zhen Zheng, Shiyang Chen, Arash Bakhtiari, Michael Wyatt, Reza Yazdani Aminabadi, Yuxiong He, Olatunji Ruwase, Leon Song, and Zhewei Yao. Zeroquant(4+2): Redefining llms quantization with a new fp6-centric strategy for diverse generative tasks. *arXiv preprint arXiv:2312.08583*, 2023.

Yifei Xia, Fangcheng Fu, Wentao Zhang, Jiawei Jiang, and Bin Cui. Efficient multi-task llm quantization and serving for multiple lora adapters. In *Proceedings of Neural Information Processing Systems*, 2024.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 38087–38099, 2023.

Haiyang Xiao, Weiqing Li, Jinyue Guo, Guochao Jiang, Guohua Liu, and Yuewei Zhang. Faq: Mitigating quantization error via regenerating calibration data with family-aware quantization. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 21441–21460, 2026.

An Yang, Baosong Yang, Beichen Zhang, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Zhen Yang, Sizai Hou, Kaiwen Zheng, Yaofang Liu, Liang He, Yixuan Chen, and Kangning Cui. Target-aware calibration data selection for preserving uncertainty in quantized language models. *arXiv preprint arXiv:2608.21019*, 2026.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. In *Proceedings of Neural Information Processing Systems*, volume 35, pp. 27168–27183, 2022.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

# A ALGORITHMIC DETAILS OF CALIBSELECT

The complete procedure of CalibSelect is summarized in Algorithm 1. In the first stage, CalibSelect randomly generates multiple candidate calibration data sets, evaluates each candidate set using CARS and LSDS, and estimates robust normalization parameters to compute a unified joint score. The top-$k$ candidates are then retained as initial solutions for the second stage. In the second stage, each initial solution is iteratively refined through one-sample swaps, and an update is accepted only when it improves the joint score. The normalization parameters estimated in the first stage remain fixed throughout this optimization process. Finally, the candidate calibration data set with the highest score is selected as the calibration data for PTQ.

**Algorithm 1** CalibSelect

**Input:** Candidate data pool $D$; calibration data size $m$; number of initial candidates $M$; maximum search steps $N_{max}$; patience $N_{stop}$; score weights $w_L$ and $w_C$; hyper-parameter $k$
**Output:** Selected calibration data $S^*$

**Stage 1: Candidate Generation & Exploration**
**for** $i = 1, \ldots, M$ **do**
  Randomly sample $S_i \subset D$, where $|S_i| = m$
  Compute $\mathrm{LSDS}(S_i)$ based on Eqs. 6-10
  Compute $\mathrm{CARS}(S_i)$ based on Eqs. 1-5
**end for**
Estimate robust normalization parameters $(c_{CARS}, z_{CARS})$ and $(c_{LSDS}, z_{LSDS})$ from $\{\mathrm{CARS}(S_i)\}_{i=1}^{M}$ and $\{\mathrm{LSDS}(S_i)\}_{i=1}^{M}$ based on Eq. 11
**for** $i = 1, \ldots, M$ **do**
  Compute joint score $Score(S_i)$ based on Eqs. 12 and 13
**end for**
Rank $\{\mathcal{S}_i\}_{i=1}^{M}$ by their joint scores
Retain the top-$k$ candidates $\{T_j\}_{j=1}^{k}$ as initial solutions
**Stage 2: Indicator-guided Candidate Optimization**
**for** $j = 1, \ldots, k$ **do**
  $S_{cur} \leftarrow T_j$
  $s_{cur} \leftarrow Score(S_{cur})$
  $n_{noimp} \leftarrow 0$, $n \leftarrow 0$
  **while** $n < N_{max}$ **and** $n_{noimp} < N_{stop}$ **do**
    Generate a neighboring candidate via a one-element swap.
    Randomly select $d_{out} \in S_{cur}$ and $d_{in} \in D \setminus S_{cur}$
    $S' \leftarrow (S_{cur} \setminus \{d_{out}\}) \cup \{d_{in}\}$
    Compute $\mathrm{LSDS}(S')$, $\mathrm{CARS}(S')$, and $Score(S')$
    $s' \leftarrow Score(S')$
    **if** $s' > s_{cur}$ **then**
      $S_{cur} \leftarrow S'$
      $s_{cur} \leftarrow s'$
      $n_{noimp} \leftarrow 0$
    **else**
      $n_{noimp} \leftarrow n_{noimp} + 1$
    **end if**
    $n \leftarrow n + 1$
  **end while**
  Store the optimized solution $\widehat{S}_j \leftarrow S_{cur}$
**end for**
$S^* = \arg \max_{\widehat{S}_j,\ j=1,\ldots,k} Score(\widehat{S}_j)$
**return** $S^*$

## B Implementation of Fake Quantization for LSDS

LSDS uses a controlled quantization perturbation to measure the layer-wise sensitivity of calibration data. We adopt a one-layer fake-quantization strategy to evaluate the effect of quantizing each transformer layer. When evaluating layer $l$, only the weight matrices in that layer are fake-quantized, while all other transformer layers remain in full precision. The resulting change in NLL is used to quantify the sensitivity of layer $l$. The implementation of fake-quantization is described as follows.

For a weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, we partition weights along the input dimension into groups of size $G$. Each group has its own scale, allowing the quantizer to adapt to local weight magnitudes. For a group $W_g$, the quantization scale is computed as:

$$s_g = \frac{\max_{w \in W_g} |w|}{2^{b-1} - 1} \tag{14}$$

We map each weight in the group to an integer in a symmetric quantization range for $b$-bit quantization:

$$Q_g = \text{clip}\left(\text{round}\left(\frac{W_g}{s_g}\right), -(2^{b-1} - 1), 2^{b-1} - 1\right) \tag{15}$$

Rounding maps scaled weights to integers, and clipping restricts them to the specified range.

The forward pass uses the de-quantized weights:

$$\widehat{W}_g = s_g Q_g \tag{16}$$

Here, $\widehat{W}_g$ is a quantized approximation of the original weight group $W_g$, rather than a recovery of the original full-precision weights. Therefore, the quantization error introduced by the low-bit representation is retained. Using $\widehat{W}_g$ in the forward pass allows us to simulate the effect of low-bit weight quantization with standard floating-point operations, without relying on hardware-specific low-bit inference kernels.

The same fake-quantization operator and configuration are used for all transformer layers and candidate calibration data sets. This ensures that the resulting sensitivity values are measured under a consistent quantization perturbation, so differences across layers and data reflect their different responses to quantization rather than changes in the quantization setting itself.

## C Detailed Implementation and Hyperparameter Settings

**Candidate Data Pools** The Mixed candidate data pool contains 1400 samples collected from four data categories: 200 code samples (100 from CodeParrot and 100 from Python-codes-25K), 400 mathematics samples (200 from GSM8K and 200 from MATH), 300 commonsense reasoning samples (100 each from AI2-ARC, OpenBookQA, and HellaSwag), and 500 general text samples from WikiText. We also construct three domain-specific candidate pools. The Code pool contains 600 samples collected from CodeParrot and Python-codes-25K, the Math pool contains 600 samples from GSM8K and MATH, and the Common pool contains 600 samples from AI2-ARC, OpenBookQA, and HellaSwag. For a fair comparison, all methods select calibration data from the same candidate pool under the same calibration budget. PTQ performance is evaluated using PPL on WikiText, with no overlap between the WikiText samples used for evaluation and those used to construct the calibration candidate pools.

**CalibSelect Hyperparameters** In the first stage, CalibSelect randomly generates $M = 100$ initial candidate calibration data sets from the corresponding candidate data pool. The top $k = 2$ candidates are retained as the starting points for the second stage. In the second stage, the local search is allowed to run for at most $N_{\max} = 100$ iterations and terminates early if no improvement in the joint score is observed for $N_{\text{stop}} = 20$ consecutive iterations. For all experiments, the two components of the joint calibration-quality score are assigned equal weights, with $w_C = w_L = 0.5$. We set the numerical stability constant to $\epsilon = 10^{-8}$.

LSDS uses $P = 30$ resampling trials per candidate calibration data set, each drawing approximately 80% of its samples. This resampling configuration is shared across pools and budgets. The fixed fake-quantization operator described in Appendix B applies 4-bit group-wise weight quantization with $G = 128$, independently of the PTQ algorithm.

Table 3: PPL (↓) comparison under a calibration budget of 16 samples.

| Model | Pool | AWQ | | | | AutoRound | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Random | CoLA | ACDM | Ours | Random | CoLA | ACDM | Ours |
| Qwen2.5-7B | Mixed | 12.7066 | 13.0208 | 12.9197 | **12.3131** | 13.6627 | 13.5741 | 13.5411 | **13.1564** |
| | Code | 12.9215 | **12.5409** | 12.5446 | 12.6623 | 12.7109 | 12.9855 | 13.0020 | **12.5119** |
| | Math | 12.7261 | 12.9497 | 12.5599 | **12.5035** | 13.5815 | 13.4370 | **12.7237** | 12.9167 |
| | Common | 13.2148 | 12.4901 | 12.8960 | **12.3849** | 12.9992 | 13.8410 | 13.6813 | **12.7198** |
| | *Avg.* | 12.8922 | 12.7504 | 12.7301 | **12.4660** | 13.2386 | 13.4594 | 13.2371 | **12.8262** |
| LLaMA3.1-8B | Mixed | 11.3564 | 11.1442 | 11.2273 | **10.9590** | 12.3069 | 12.1692 | **11.9863** | 12.1773 |
| | Code | 11.2952 | 11.4390 | 11.3681 | **11.0945** | 12.2815 | 12.2725 | 12.2230 | **12.1463** |
| | Math | 11.2669 | 11.5289 | 11.1141 | **11.1091** | 12.1203 | 12.3355 | 12.1296 | **12.1158** |
| | Common | 11.2196 | 11.2673 | 11.3962 | **11.0123** | 12.0852 | 12.0665 | 12.0183 | **11.0955** |
| | *Avg.* | 11.2845 | 11.3448 | 11.2765 | **11.0437** | 12.1985 | 12.2109 | 12.0893 | **11.8837** |

Table 4: PPL (↓) comparison under a calibration budget of 32 samples.

| Model | Pool | AWQ | | | | AutoRound | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Random | CoLA | ACDM | Ours | Random | CoLA | ACDM | Ours |
| Qwen2.5-7B | Mixed | 12.7974 | **12.4786** | 12.8492 | 12.6101 | 13.5992 | 13.9672 | 13.7969 | **12.7302** |
| | Code | 12.7593 | 12.8075 | 12.9966 | **12.3373** | 13.0610 | 12.9686 | **12.5810** | 13.0170 |
| | Math | 12.8069 | 12.5292 | 12.5755 | **12.4574** | 13.2104 | **12.9659** | 13.2759 | 13.1713 |
| | Common | 12.8187 | 12.5776 | 12.8540 | **12.4827** | 13.2278 | 14.3041 | 13.5227 | **12.2861** |
| | *Avg.* | 12.7956 | 12.5982 | 12.8189 | **12.4719** | 13.2746 | 13.5515 | 13.2942 | **12.8012** |
| LLaMA3.1-8B | Mixed | 11.1566 | **11.0632** | 11.3368 | 11.0955 | 12.3865 | 12.2786 | 12.1901 | **12.1606** |
| | Code | 11.2864 | 11.3468 | 11.2129 | **11.0754** | 12.2002 | **12.1849** | 12.1870 | 12.2010 |
| | Math | 11.2373 | 11.4572 | 11.3271 | **11.0348** | 12.3496 | 12.2160 | 12.3761 | **12.1654** |
| | Common | 11.4054 | 11.4650 | **11.1536** | 11.1729 | 12.1409 | **11.9812** | 12.1391 | 12.1622 |
| | *Avg.* | 11.2714 | 11.3330 | 11.2576 | **11.0947** | 12.2693 | **12.1652** | 12.2231 | 12.1723 |

**Downstream PTQ Configuration** AWQ and AutoRound use INT4 weight quantization with a group size of 128 unless otherwise specified. For each model, candidate pool, and calibration budget, Random, CoLA, ACDM, and CalibSelect use the same downstream quantization configuration. CalibSelect only changes the calibration data supplied to the PTQ procedure and does not modify the quantization objective or optimization procedure of AWQ or AutoRound.

## D Detailed Results across Calibration Budgets

This appendix presents PPL for calibration budgets of 16, 32, 64, and 128, complementing the averaged results reported in Table 1 and Fig. 4. Results are provided across different candidate data pools, models, and quantization methods, allowing a detailed comparison of Random, CoLA, ACDM, and CalibSelect.

Tables 3-6 present PPL for calibration budgets of 16, 32, 64, and 128. Overall, CalibSelect achieves the lowest PPL in 46 out of 64 individual experimental settings across the candidate data pools. It also obtains the lowest average PPL in 15 out of 16 combinations of model, quantization method, and calibration budget. The advantage is particularly evident under small calibration budgets. With a calibration budget of only 16 samples, CalibSelect reduces the average PPL of Qwen2.5-7B by 3.31%, 2.23%, and 2.07% compared with Random, CoLA, and ACDM, respectively, when using AWQ. With AutoRound, the corresponding reductions are 3.12%, 4.70%, and 3.10%. Similar improvements are observed on LLaMA3.1-8B across the three baselines. These results indicate that CalibSelect can identify calibration data that are effective for PTQ even when only a limited calibration budget is available.

The improvements are not universal. CoLA performs better on the Code pool for Qwen2.5-7B with AWQ at a calibration budget of 16 samples, while ACDM achieves lower PPL on the Code pool for Qwen2.5-7B with AutoRound at a budget of 32 samples. A similar exception occurs for

Table 5: PPL (↓) comparison under a calibration budget of 64 samples.

| Model | Pool | AWQ | | | | AutoRound | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Random | CoLA | ACDM | Ours | Random | CoLA | ACDM | Ours |
| Qwen2.5-7B | Mixed | 12.8739 | 12.6272 | 12.9877 | **12.4215** | 13.0317 | 13.8580 | 13.7119 | **12.6842** |
| | Code | 12.7405 | 12.6946 | **12.5017** | 12.6080 | 13.3332 | 13.2639 | **12.6667** | 13.2266 |
| | Math | 12.5115 | 12.9493 | 12.6782 | **12.4522** | 13.4930 | 13.5635 | 14.1342 | **12.9439** |
| | Common | 12.6645 | 12.5214 | 12.8960 | **12.5172** | 13.2581 | 13.1101 | 13.8446 | **12.7090** |
| | *Avg.* | 12.6976 | 12.6981 | 12.7659 | **12.4997** | 13.2790 | 13.4489 | 13.5893 | **12.8909** |
| LLaMA3.1-8B | Mixed | 11.1835 | 11.3040 | 11.3361 | **11.0524** | 12.2374 | 12.2515 | 12.2227 | **12.0852** |
| | Code | 11.2512 | 11.3166 | 11.3145 | **10.9423** | 12.4647 | 12.4196 | **12.2237** | 12.2446 |
| | Math | 11.2373 | 11.3234 | 11.2562 | **10.9731** | 12.3213 | 12.2127 | 12.2324 | **12.1813** |
| | Common | 11.2557 | 11.4137 | 11.1820 | **10.9214** | 12.3134 | 12.3575 | 12.1160 | **11.9607** |
| | *Avg.* | 11.2319 | 11.3394 | 11.2722 | **10.9723** | 12.3342 | 12.3103 | 12.1987 | **12.1179** |

Table 6: PPL (↓) comparison under a calibration budget of 128 samples.

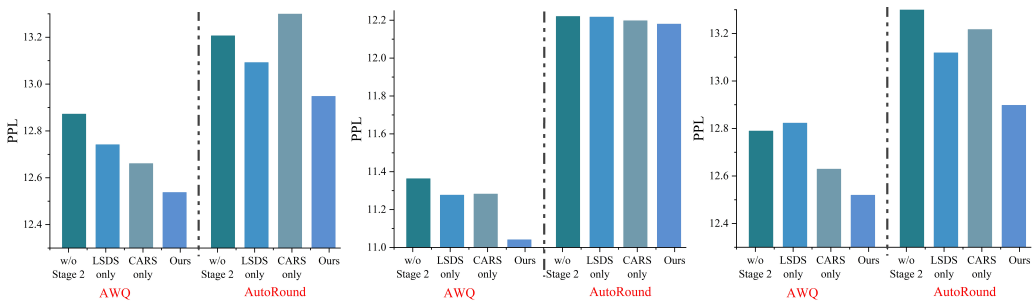
| Model | Pool | AWQ | | | | AutoRound | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Random | CoLA | ACDM | Ours | Random | CoLA | ACDM | Ours |
| Qwen2.5-7B | Mixed | 12.5230 | 12.8181 | 12.7778 | **12.2227** | 13.4346 | 12.9526 | 13.4941 | **12.9446** |
| | Code | **12.3574** | 12.7607 | 12.7913 | 12.5422 | 13.5687 | **12.9108** | 13.1765 | 13.0356 |
| | Math | **12.5466** | 12.8951 | 12.5860 | 12.6634 | 12.8137 | 12.9975 | 13.2658 | **12.5585** |
| | Common | 12.9385 | 12.7860 | 12.7708 | **12.4464** | 13.1088 | 13.1079 | 13.8905 | **12.7016** |
| | *Avg.* | 12.5914 | 12.8150 | 12.7315 | **12.4686** | 13.2315 | 12.9922 | 13.4567 | **12.8101** |
| LLaMA3.1-8B | Mixed | 11.0837 | 11.2774 | 11.3242 | **10.8691** | 12.2459 | 12.2900 | 12.2345 | **12.1142** |
| | Code | 11.4557 | 11.2227 | 11.2256 | **11.0498** | 12.2759 | 12.2349 | 12.2724 | **12.1272** |
| | Math | 11.2894 | 11.3383 | 11.5206 | **11.1460** | 12.2479 | 12.3162 | **12.1705** | 12.1813 |
| | Common | 11.2570 | 11.1839 | 11.3463 | **10.9722** | 12.2423 | 12.2791 | **12.1640** | 12.1670 |
| | *Avg.* | 11.2715 | 11.2556 | 11.3542 | **11.0093** | 12.2530 | 12.2801 | 12.2104 | **12.1474** |

LLaMA3.1-8B with AutoRound at a budget of 32 samples, where CoLA achieves an average PPL of 12.1652 compared with 12.1723 for CalibSelect. Nevertheless, CalibSelect achieves the lowest average PPL in nearly all combinations of model, quantization method, and calibration budget. Taken together, the results demonstrate the effectiveness of CalibSelect across different candidate data pools, models, quantization methods, and calibration budgets.

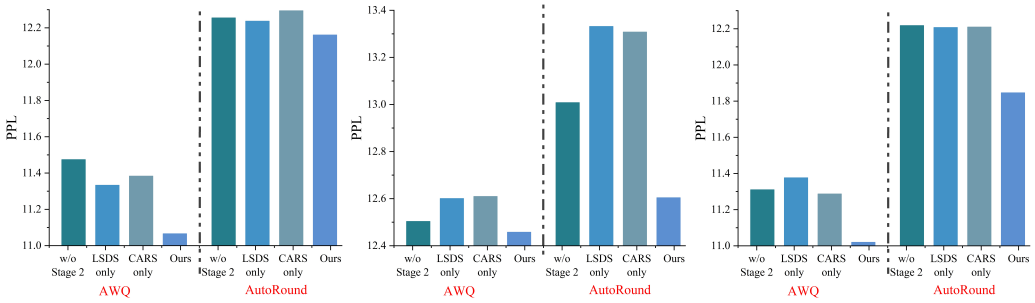
## E Detailed Ablation Results

This appendix presents additional ablation results to examine the contribution of each component of CalibSelect. We evaluate three ablated variants, *w/o Stage 2*, *LSDS only*, and *CARS only*, on the Code, Math, and Common candidate data pools using Qwen2.5-7B and LLaMA3.1-8B with both AWQ and AutoRound. Results on the Mixed candidate data pool are reported in the main text. To reduce the dependence of the comparison on any single calibration budget, the PPL reported for each setting is averaged over the four calibration budgets of 16, 32, 64, and 128.

As shown in Fig. 6, CalibSelect consistently achieves the lowest PPL in all settings. Removing the second stage generally leads to clear performance degradation, indicating that the indicator-guided local optimization further improves the initial candidate calibration data sets obtained in the first stage. Using either metric alone generally results in higher PPL than CalibSelect. On Qwen2.5-7B, this difference is particularly clear: combining the two metrics consistently leads to better quantization performance. The detailed ablation results show that the performance gains of CalibSelect cannot be attributed to a single component. Combining LSDS and CARS provides more information for calibration data selection, while the second stage further improves the selected calibration data through indicator-guided local optimization.




(a) Qwen2.5-7B on Code. (b) LLaMA3.1-8B on Code. (c) Qwen2.5-7B on Math.




(d) LLaMA3.1-8B on Math. (e) Qwen2.5-7B on Common. (f) LLaMA3.1-8B on Common.

Figure 6: Ablation study under different calibration data pools. The reported PPL values are averaged over calibration budgets of 16, 32, 64, and 128.

## F Relationship Between CARS, LSDS, and PTQ Performance

Evaluating a candidate calibration data set based on its final PPL requires a complete downstream PTQ run. CalibSelect instead uses CARS and LSDS as empirical selection proxies. To examine whether the characteristics captured by these metrics are empirically associated with PTQ performance, we compare their values with the resulting PPL. Taking Qwen2.5-7B with AWQ as a representative setting, Table 7 reports the CARS, LSDS, and corresponding PPL under calibration budgets of 16, 32, 64, and 128. For ease of interpretation, we report the original CARS and LSDS values before normalization. To reduce the influence of any particular data source, all values are averaged over the Mixed, Code, Math, and Common candidate data pools for each calibration budget.

As shown in Table 7, under each fixed calibration budget, CalibSelect selects calibration data with higher LSDS and CARS values than random sampling while consistently achieving lower PPL. At a budget of 16 samples, LSDS increases from 1.4340 to 2.4783 and CARS from 0.8650 to 1.0682, while PPL decreases from 12.8922 to 12.4660, corresponding to a reduction of 3.31%. Similar improvements are observed at budgets of 32 and 64 samples, where PPL is reduced by 2.53% and 1.56%, respectively. At a budget of 128 samples, CalibSelect still achieves higher LSDS (3.7909 vs. 3.3055) and CARS (1.0439 vs. 0.9113), together with a lower PPL (12.4686 vs. 12.5914). These results show that, under each fixed calibration budget, higher LSDS and CARS values are consistently associated with lower PPL. This consistent trend across different calibration budgets supports the use of LSDS and CARS as practical criteria for calibration data selection in PTQ. Together with the ablation results in Section E, these findings further support their effectiveness in guiding the selection of calibration data.

## G Computational Complexity Analysis

We analyze the computational complexity of CalibSelect under a conservative upper-bound setting. Let $N$ denote the number of samples in the candidate data pool, $m$ the calibration budget, $L$ the number of evaluated transformer layers, $P$ the number of LSDS resampling trials, and $r$ the number of samples used in each resampling trial. Let $J$ denote the number of activation statistics used by CARS, where $J = 4$ in our implementation. The first stage generates $M$ initial candidate calibration

Table 7: LSDS, CARS, and PPL across calibration budgets for Qwen2.5-7B with AWQ. The values at each budget are averaged over the Mixed, Code, Math, and Common calibration data pools.

| Model | Budget | Random | | | Ours | | |
|---|---|---|---|---|---|---|---|
| | | LSDS | CARS | PPL | LSDS | CARS | PPL |
| Qwen2.5-7B | 16 | 1.4340 | 0.8650 | 12.8922 | 2.4783 | 1.0682 | 12.4660 |
| | 32 | 1.8352 | 0.9647 | 12.7956 | 2.5063 | 1.0563 | 12.4719 |
| | 64 | 2.2656 | 0.9871 | 12.6976 | 3.0551 | 1.0447 | 12.4997 |
| | 128 | 3.3055 | 0.9113 | 12.5914 | 3.7909 | 1.0439 | 12.4686 |

data sets, while the second stage starts from the top-$k$ candidates and performs at most $N_{\max}$ local-search iterations for each starting point.

CalibSelect computes and caches model-dependent statistics on demand during the two-stage search. Let $U$ denote the number of distinct samples encountered during the search, where $U \leq N$. To establish a conservative upper bound, we assume that all samples in the candidate data pool are eventually visited, i.e., $U = N$.

**Model-dependent statistic extraction** For each sample, LSDS requires one complete forward pass through the full-precision model to obtain the NLL and $L$ additional complete forward passes through the corresponding one-layer fake-quantized model variants. Let $\mathcal{C}_{\text{fwd}}$ denote the cost of one complete forward pass through the entire model. Computing the layer-wise sensitivity responses for $U$ distinct samples therefore requires $\mathcal{O}(U(L+1)\mathcal{C}_{\text{fwd}})$. Under the conservative assumption that all $N$ samples are visited, this cost is bounded by:

$$\mathcal{O}\left(N(L+1)\mathcal{C}_{\text{fwd}}\right) \tag{17}$$

The full-precision forward pass simultaneously provides the hidden states required by CARS. For each sample and transformer layer, CARS computes the mean absolute activation, RMS activation, and the 95th and 99th percentiles over $B_iC$ activation values, where $B_i$ and $C$ denote the sequence length and hidden dimension, respectively. The mean and RMS statistics require $\mathcal{O}(B_iC)$ operations, while exact quantile computation based on sorting requires $\mathcal{O}(B_iC\log(B_iC))$. Therefore, the additional statistic-extraction cost over all $N$ samples and $L$ layers is conservatively bounded by $\mathcal{O}\left(NL\,B_iC\log(B_iC)\right)$. This cost is substantially smaller than the model forward cost and is therefore omitted from the complexity analysis below.

Importantly, during the two-stage search, CalibSelect computes and caches these statistics when a sample is first encountered. Subsequent candidate calibration data set evaluations reuse the cached statistics, requiring additional model forward passes only for previously unseen samples. Consequently, each distinct sample requires model-dependent statistic extraction at most once for a given model and quantization configuration. Therefore, the cost of model-dependent statistic extraction is bounded by $\mathcal{O}\left(N(L+1)\mathcal{C}_{\text{fwd}}\right)$.

**LSDS computation** Once the required sensitivity responses have been cached, LSDS operates only on the corresponding sensitivity values. For a candidate calibration data set, each of the $P$ resampling trials averages the sensitivities of $r$ samples across $L$ layers. The computational complexity of LSDS is therefore:

$$\mathcal{O}(PrL) \tag{18}$$

Since $r$ is a fixed fraction of the calibration budget in our implementation ($r = 0.8m$), this complexity can equivalently be written as $\mathcal{O}(PmL)$.

**CARS computation** Once the required activation statistics have been cached, CARS operates only on the corresponding statistics. For each transformer layer $l$ and activation-statistic type $j$, CARS computes the 10th and 90th percentiles over the $m$ samples in the current candidate. If these percentiles are obtained by sorting the $m$ values, the cost for one layer-statistic pair is $\mathcal{O}(m\log m)$. Since there are $L$ layers and $J$ activation-statistic types, the total complexity of CARS for a candidate calibration data set is:

$$\mathcal{O}(LJm\log m) \tag{19}$$

In our implementation, $J = 4$ is fixed; therefore, the complexity can be written as $\mathcal{O}(Lm\log m)$.

Table 8: Computational complexity of the main components of CalibSelect, assuming that all $N$ samples in the candidate data pool are visited.

| Component | Number / Scope | Complexity |
|---|---|---|
| Full-precision statistics | $N$ samples | $\mathcal{O}(N\mathcal{C}_{\text{fwd}})$ |
| Layer-wise sensitivity extraction | $N \times L$ | $\mathcal{O}(NL\mathcal{C}_{\text{fwd}})$ |
| LSDS evaluation | one candidate | $\mathcal{O}(PrL)$ |
| CARS evaluation | one candidate | $\mathcal{O}(LJm \log m)$ |
| Two-stage candidate scoring | $M + kN_{\max}$ | $\mathcal{O}((M + kN_{\max})(PrL + LJm \log m))$ |
| Stage 1 ranking | $M$ candidates | $\mathcal{O}(M \log M)$ |

**Two-stage candidate search** The search evaluates at most $Q = M + kN_{\max}$ candidates: $M$ in the first stage and up to $N_{\max}$ for each of the $k$ searches in the second stage. Given the computational costs of LSDS and CARS for each candidate calibration data set, the total cost of calibration data evaluation during the two-stage search is:

$$\mathcal{O}\left(Q\left(PmL + Lm \log m\right)\right) \tag{20}$$

The first stage additionally ranks the $M$ initial candidate calibration data sets, which requires $\mathcal{O}(M \log M)$ time. Table 8 summarizes the computational complexity of the main components of CalibSelect under the conservative assumption that all samples in the candidate data pool are visited.

**Overall computational complexity.** The main computational cost of CalibSelect comes from the model-dependent statistic extraction and the two-stage search. Therefore, the overall complexity of CalibSelect is:

$$\mathcal{O}\left(N(L+1)\mathcal{C}_{\text{fwd}} + QLm(P + \log m) + M \log M\right) \tag{21}$$

Statistic extraction is performed once per model, pool, and configuration. Two-stage search is then performed for each calibration budget using the cache. Once a calibration data set is selected, CalibSelect adds no computation to the PTQ procedure.

**Measured runtime** We measure the runtime of CalibSelect on Qwen2.5-7B with a candidate data pool of 1,400 samples. As a conservative estimate, computing and caching statistics for the entire pool takes 1540.73 seconds. Table 9 reports the full-pool extraction time and the subsequent search times using cached statistics. The two-stage search takes 0.54, 0.83, 0.67, and 1.20 seconds for calibration budgets of 16, 32, 64, and 128, respectively.

For comparison, evaluating a single candidate calibration data set of 16 samples using the PPL obtained after full AWQ quantization takes 471.28 seconds. At that rate, evaluating the $M = 100$ initial candidate calibration data sets would take approximately 47128 seconds (13.09 hours), before any local refinement. Under the full-pool caching setting, CalibSelect requires 1540.73 seconds for statistic extraction and 0.54 seconds for the subsequent two-stage search at the same calibration budget. This comparison is conservative, as it excludes the additional candidate calibration data set evaluations required during the second stage when PPL is used as the selection criterion. Consistent with the complexity analysis, the computational cost of CalibSelect is dominated by one-time statistic extraction, while the two-stage search incurs minimal additional overhead by reusing cached statistics, thereby avoiding repeated full PTQ evaluations.

## H Limitations of the Metrics

**CARS defines coverage through quantiles of activation statistics** For each layer $l$ and statistic $j$, the 10th and 90th percentiles across the calibration data set define $I_{l,j}(S) = [q_{l,j}^{\text{low}}, q_{l,j}^{\text{high}}]$. CARS aggregates the relative ranges of these intervals, as defined in Eqs. 4-5. If all samples have identical summaries $a_{i,l}^{(j)} = c_{l,j}$, each interval collapses and $\text{CARS}(S) = 0$, even if the raw activations within individual samples vary substantially. This represents a potential limitation of CARS.

Table 9: Runtime of CalibSelect on Qwen2.5-7B. Statistic extraction is performed once for all samples in the candidate data pool. Two-stage candidate search uses the cached statistics, with runtime reported for each calibration budget.

| Model | Pool Size | Statistic Extraction | Budget | Two-stage Candidate Search |
|---|---|---|---|---|
| Qwen2.5-7B | 1400 | 1540.73s (one-time) | 16 | 0.54 s |
| | | | 32 | 0.83 s |
| | | | 64 | 0.67 s |
| | | | 128 | 1.20 s |

Nevertheless, CARS provides a compact representation of high-dimensional activation responses through a set of statistics. This representation retains key statistical characteristics relevant to activation coverage while reducing the computational and storage costs of processing raw activations. Although the aggregated statistics may not fully capture fine-grained variations in the underlying activation distributions, they provide a practical basis for assessing calibration data quality.

**LSDS characterizes layer-wise sensitivity discrimination and estimation stability** LSDS evaluates calibration data based on differences in quantization sensitivity across layers and the stability of these estimates under resampling. A higher score indicates more pronounced layer-wise sensitivity differences with lower estimation variability. An extreme case arises when all calibration samples produce identical sensitivity vectors $e_{i,l} = v_l$. In this case, the resampling variability becomes zero, and LSDS reduces to $\mathrm{Std}_l(v_l)/\epsilon$, potentially yielding an excessively large score. This is a potential limitation because consistent sensitivity estimates within a calibration set do not necessarily indicate that the calibration data set adequately represents the differences in quantization sensitivity across layers. Nevertheless, our empirical analysis shows that this case rarely occurs in practice, suggesting that its impact on calibration data selection is limited.

LSDS also uses a standardized fake-quantization operator to estimate layer-wise sensitivity without performing full PTQ. However, since different PTQ methods employ different quantization strategies, the estimated sensitivities may not fully reflect the behavior of a specific PTQ method. For calibration data selection tailored to a particular PTQ method, the fake-quantization operator can be adapted to better reflect its quantization behavior.