CALM: A CKA-Guided Adaptive Layer-Wise Modularization Framework for LLM Quantization

Anonymous ACL submission

Abstract

Uniform PTQ methods for large language models often overlook substantial heterogeneity in layer-wise quantization behavior. We present CALM, a CKA-guided Adaptive Layer-wise Modularization framework that selects quantization methods on a per-layer basis. For each layer, CALM evaluates multiple PTQ candidates and uses Linear Centered Kernel Alignment (CKA) to measure representational fidelity with respect to the full-precision model. The best candidate for each layer is then composed into a heterogeneous quantized model, with no retraining or fine-tuning required. Across Llama and Qwen models, CALM yields lower perplexity and stronger downstream performance than strong single-method PTQ baselines, while also remaining competitive with recent mixed-precision approaches. Our results indicate that layer-wise method selection is an effective complement to bit-width design in low-bit LLM quantization.

1 Introduction

Post-training quantization (PTQ) (Frantar et al., 2023; Gholami et al., 2021; Nagel et al., 2020) has become a key technique for compressing large language models (LLMs). Recent methods such as GPTQ (Frantar et al., 2022), AWQ (Lin et al., 2024), SmoothQuant (Xiao et al., 2023), OmniQuant (Shao et al., 2024), SpinQuant (Liu et al., 2025), and QuaRot (Ashkboos et al., 2024) improve low-bit quantization from different perspectives, including weight reconstruction, salient channel protection, activation outlier suppression, learnable calibration, rotation-based transformation, and distribution reshaping.

Despite these advances, most PTQ methods still rely on an implicit assumption: a single quantization algorithm can be uniformly applied to all Transformer layers. Once a method such as GPTQ, AWQ, or SmoothQuant is selected, the same quantization mechanism is typically used throughout the entire model. This design is simple and deployment-friendly, but it overlooks substantial layer-wise heterogeneity in LLMs.

PTQ errors in LLMs are highly structured and vary with network depth, channel structure, and activation outliers. Shallow layers are often more affected by massive activation outliers, where smoothing-based transformations can better migrate quantization difficulty from activations to weights. In contrast, middle and deeper layers may suffer more from weight quantization error, salient channel distortion, or Hessian-sensitive directions, where activation-aware weight protection or second-order reconstruction can be more suitable (An et al., 2025; Raman et al., 2025; Park and Chung, 2026). Therefore, low-bit LLM quantization is not only about *how many bits* each layer should use, but also about *which quantization mechanism* each layer should adopt.

Existing layer-wise adaptive quantization mainly focuses on mixed precision, which assigns different bit-widths to layers or channels to balance model size, efficiency, and accuracy. This line of work primarily answers *how many bits* each layer should use, but leaves an orthogonal question underexplored: under the same bit-width, *which quantization mechanism* should be assigned to each layer?

> **Method heterogeneity.** Can layer-wise adaptive quantization move beyond *bit-width allocation* to *method assignment*, selecting a suitable PTQ mechanism for each layer under a fixed low-bit budget?

We therefore formulate fixed-bit cross-method PTQ assignment as an adaptation dimension orthogonal to bit allocation: the bit-width remains fixed, while the quantization mechanism is selected layer by layer. This perspective is motivated by the complementary behaviors of existing PTQ methods: SmoothQuant mitigates activation-side out-

liers, AWQ protects activation-aware salient weight channels, and GPTQ performs approximate second-order weight reconstruction. Thus, different methods may be preferable for different layers even under the same target bit-width.

To this end, we propose CALM, a CKA-guided Adaptive Layer-wise Modularization framework for method-heterogeneous PTQ. CALM treats each Transformer layer as an independently quantizable module, evaluates multiple PTQ candidates under the same bit-width, and uses Linear CKA to select the candidate that best preserves the full-precision representation. The selected layers are then statically assembled into a heterogeneous quantized model, requiring no retraining, fine-tuning, runtime routing, or input-dependent switching.

We evaluate CALM on Llama and Qwen models. Under fixed low-bit constraints, CALM improves over uniform single-method PTQ baselines and remains competitive with strong mixed-precision and transformation-based methods. Ablations show that the gains come from candidate-method complementarity, CKA-guided selection, and sequential layer-wise assembly, while introducing no measurable inference overhead. Our contributions are summarized as follows:

❶ We formulate *method-heterogeneous PTQ* as fixed-bit cross-method layer-wise assignment, introducing an adaptation dimension orthogonal to mixed-precision quantization. ❷ We propose **CALM**, a training-free framework that uses Linear CKA to select the PTQ method best preserving full-precision representations for each Transformer layer. ❸ Across Llama and Qwen models, CALM improves over uniform PTQ baselines, remains competitive with mixed-precision and transformation-based methods, and incurs negligible runtime overhead.

2 Related Work

Representative low-bit quantization methods: GPTQ (Frantar et al., 2022) (second-order reconstruction), AWQ (Lin et al., 2024) (salient channels), SmoothQuant (Xiao et al., 2023) (activation-weight balance). Others improve calibration/rotation/distribution: OmniQuant (Shao et al., 2024), SpinQuant (Liu et al., 2025), QuaRot (Ashkboos et al., 2024), FlatQuant (Sun et al., 2025). These provide complementary mechanisms but apply uniformly across layers. Layer-wise adaptive quantization uses mixed precision with varying bit-widths (Dong et al., 2019; Dettmers et al., 2024; Wei et al., 2022). However, these methods primarily adapt numerical precision, while fixed-bit layer-wise assignment of heterogeneous PTQ mechanisms remains underexplored. CALM keeps fixed bit-width but adapts PTQ mechanisms, differing from scale/rotation tuning methods. It performs cross-method layer-wise assignment among heterogeneous PTQ candidates. Unlike dynamic routing/MoQE methods with input-dependent choices (Shazeer et al., 2017; Fedus et al., 2022; Zhang et al., 2025; Tao et al., 2025; Chitty-Venkata et al., 2025), CALM determines assignments offline using Linear CKA (Kornblith et al., 2019) as label-free proxy, statically assembling without labels, fine-tuning, or runtime switching.

3 Methodology

3.1 Layer-wise Algorithmic Suitability

A central observation behind CALM is that quantization quality is not only method-dependent, but also layer-dependent. Given a fixed bit-width, a PTQ method that preserves the representation well for one Transformer layer may be less suitable for another layer. We refer to this layer-specific preference as algorithmic suitability.

Recent studies suggest that outliers in LLMs are structured rather than random numerical artifacts. Massive activations and channel-wise outliers exhibit different distributional patterns across layers and can lead to distinct quantization difficulties (Raman et al., 2025; An et al., 2025). Moreover, analyses of attention sinks and compression valleys further indicate that representation and compression behaviors vary with network depth (Queipo-de-Llano et al., 2025). These observations motivate our hypothesis that different Transformer layers may prefer different PTQ mechanisms under the same bit-width. For example, layers more affected by activation-side outliers may benefit more from smoothing-based methods, whereas layers more sensitive to salient weight channels or reconstruction errors may be better handled by activation-aware protection or reconstruction-based methods.

To cover these complementary quantization behaviors, we instantiate the candidate pool as

$$\mathcal{P} = \{\mathrm{SmoothQuant}, \mathrm{AWQ}, \mathrm{GPTQ}\}.$$

These methods represent activation-side smoothing, activation-aware salient-channel protection,

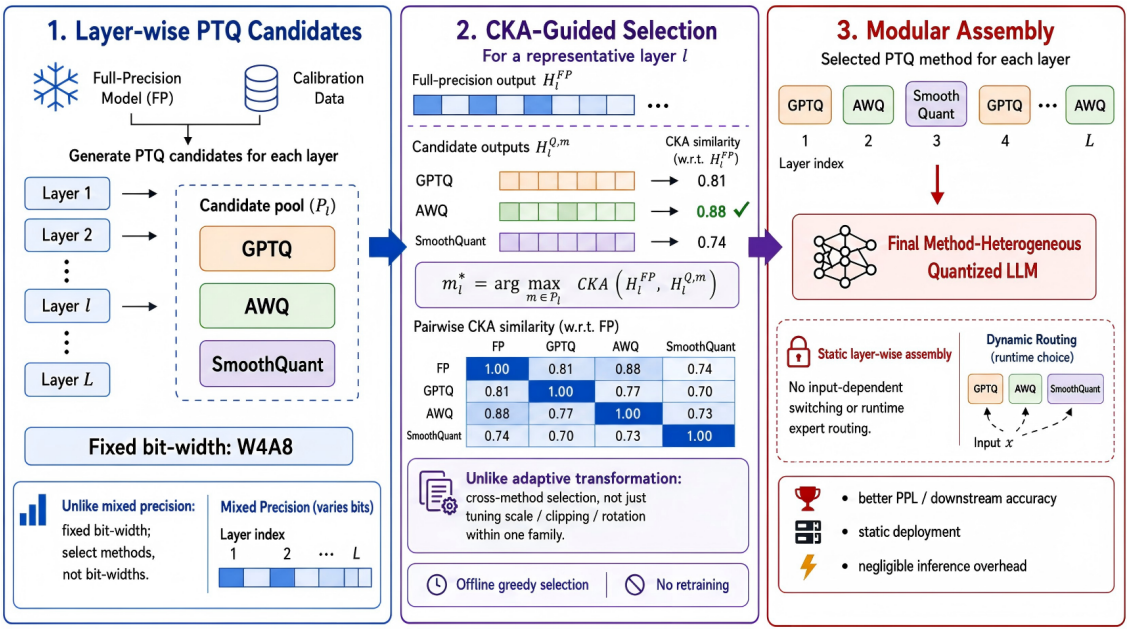

Figure 1: Overview of CALM. CALM constructs layer-wise PTQ candidates under a fixed W4A8 setting, selects the CKA-preferred method for each layer, and statically assembles a method-heterogeneous quantized LLM without retraining or runtime routing.

Figure 2: Layer-Wise CKA score distribution of Llama-3.1-8B under different quantization methods.

and second-order weight reconstruction, respectively. CALM does not assume that any one of them is globally optimal. Instead, it treats them as candidate operators whose suitability is determined separately for each layer.

Figure 2 illustrates this layer-dependent behavior: the method with the highest representation similarity varies across network depth, and no single PTQ method consistently dominates all layers. This motivates a layer-wise method selection strategy, where each layer is assigned the PTQ method that best preserves its full-precision representation.

3.2 The CALM Framework

Motivated by the observed layer-wise heterogeneity in algorithmic suitability, we propose CALM, whose overall pipeline is illustrated in Algorithm 1 and Figure 1. The core objective of CALM is to allow each Transformer layer to select the quantization method that best preserves its representational structure, rather than enforcing a single homogeneous PTQ strategy across the entire model.

Why CKA instead of MSE? Different weights and activation channels in LLMs are not equally important. A few activation-induced salient channels can disproportionately affect model outputs. Thus, low-bit quantization should preserve these critical channels rather than only minimizing average error. However, MSE mainly measures numerical discrepancy and does not explicitly capture channel salience or outlier structures. For CALM, relying on MSE may bias selection toward locally reconstruction-friendly methods while underestimating methods that preserve functionality through salient-channel protection or activation-outlier mitigation. Linear CKA instead measures representation-level alignment between full-precision and quantized hidden states, making it better suited for assessing whether the overall representation structure is preserved. We validate this choice in Section 4.6.1 and provide details in Appendix A.2.

CALM consists of three main stages. First, we construct a candidate pool of PTQ algorithms, including GPTQ, AWQ, and SmoothQuant. Although SpinQuant achieves strong accuracy, we

Algorithm 1 CALM: CKA-Guided Layer-Wise PTQ Selection

Require: FP model $M_{\text{FP}} = \{\ell_1, \ldots, \ell_L\}$, calibration set $\mathcal{D}_{cal}$, PTQ pool $\mathcal{P}$, bit-width B
Ensure: Hybrid quantized model M_{Hybrid}

Cache FP activations $\{H_{\text{FP}}^{(0)}, \ldots, H_{\text{FP}}^{(L)}\}$ on $\mathcal{D}_{cal}$
$M_{\text{Hybrid}} \leftarrow \emptyset, \quad H_{\text{Q}}^{(0)} \leftarrow H_{\text{FP}}^{(0)}$
for $l = 1$ to L **do**
  for each $m \in \mathcal{P}$ **do**
    $\hat{\ell}_l^m \leftarrow \text{Quantize}_m(\ell_l, H_{\text{Q}}^{(l-1)}, B)$
    $S_m \leftarrow \text{CKA}\Big(H_{\text{FP}}^{(l)}, \hat{\ell}_l^m(H_{\text{Q}}^{(l-1)})\Big)$
  end for
  $m_l^* \leftarrow \arg\max_{m \in \mathcal{P}} S_m$
  $M_{\text{Hybrid}}.\text{append}(\hat{\ell}_l^{m_l^*})$
  $H_{\text{Q}}^{(l)} \leftarrow \hat{\ell}_l^{m_l^*}(H_{\text{Q}}^{(l-1)})$
end for
return M_{Hybrid}

exclude it from the default pool to avoid additional rotation-based runtime overhead.

Second, we decouple an L-layer LLM into L quantizable modules. For each layer l, every method in the candidate pool is applied independently, producing multiple quantized candidates for the same layer.

Third, we compute the CKA similarity between each candidate's output activation and the corresponding full-precision activation. For layer l, CALM selects the candidate with the highest CKA score under the calibration set and appends it to the hybrid model. This sequential selection encourages each quantized layer to preserve the full-precision representation consumed by subsequent layers.

3.3 Greedy Layer-Wise Selection Strategy

Finding the globally optimal layer-wise assignment requires searching over $|\mathcal{P}|^L$ method combinations, which is infeasible for modern LLMs. CALM therefore adopts a greedy strategy that selects one layer at a time, yielding an $O(L|\mathcal{P}|)$ search while providing a favorable accuracy–cost trade-off in practice.

The selection follows the sequential nature of Transformer inference: preserving each layer's representation helps stabilize the inputs to subsequent layers and mitigates error accumulation. Formally, the method for layer l is selected as

$$m_l^* = \arg\max_{m \in \mathcal{P}} \text{CKA}\Big(\mathbf{H}_{FP}^{(l)}, f_m\Big(\mathbf{H}_Q^{(l-1)}; \mathbf{W}_l\Big)\Big), \tag{1}$$

where $\mathbf{H}_{FP}^{(l)}$ is the full-precision target activation and $f_m(\cdot; \mathbf{W}_l)$ denotes layer l quantized by method m. Conditioning on $\mathbf{H}_Q^{(l-1)}$ allows the decision to account for perturbations introduced by previously selected quantized layers.

We further examine stronger offline search variants, including Beam-CKA and Overlapping Block-CKA, in Appendix A.8. These variants explore larger assignment spaces but provide only marginal gains over Greedy-CKA in our experiments. Although greedy selection does not guarantee a globally optimal assignment, it consistently improves over uniform single-method baselines and offers a practical balance between search efficiency and quantization quality. Therefore, we use Greedy-CKA as the default selection strategy in CALM.

3.4 Complexity and Inference Efficiency

CALM incurs a one-time offline search cost. For an L-layer Transformer and a PTQ pool $\mathcal{P}$, greedy selection requires $O(L|\mathcal{P}|)$ candidate evaluations, which is more expensive than a uniform PTQ pipeline but remains linear in depth and is performed only during calibration.

After selection, CALM becomes a static heterogeneous quantized model with no runtime routing, input-dependent switching, or extra controller. Method heterogeneity only affects offline calibration and export: all selected layers share the same W4A8 runtime format, with packed INT4 weights and INT8 activation interfaces. Rescaling factors from AWQ or SmoothQuant are folded into weights or adjacent affine parameters before inference. Therefore, CALM reuses standard low-bit kernels without method-specific dispatch, and introduces only negligible runtime overhead on A800 GPUs, as shown in Appendix A.6.

4 Experiments

Our experiments are divided into three parts: PPL evaluation, downstream task evaluation, and configuration studies. The PPL and downstream experiments are designed to demonstrate the superiority of our quantized models, while the configuration experiments primarily illustrate how algorithm performance varies under different settings. These configuration studies also serve as ablation studies, helping to isolate and analyze the impact of individual design choices.

4.1 Experimental Setup

Models. We evaluate CALM on two representative LLM families, including Llama-3.1-8B (Team, 2024), Llama-3.2-1B, Llama-3.2-3B (Meta AI,

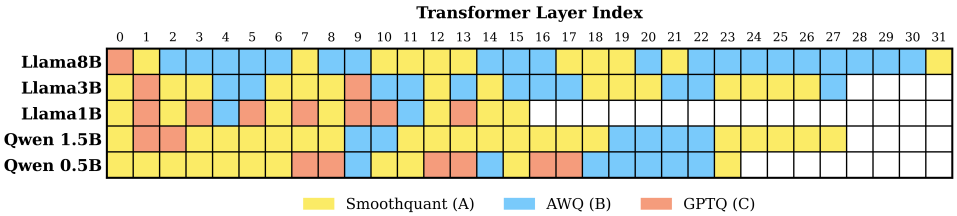

Figure 3: Layer-wise distribution of PTQ methods selected by CALM across five LLMs. The selected method varies across depth and model families, indicating that no single PTQ method dominates all Transformer layers under the evaluated W4A8 setting.

2024), Qwen2.5-1.5B, and Qwen2.5-0.5B (Yang et al., 2024).

Datasets. For language modeling evaluation, we report perplexity on C4 (Raffel et al., 2020) and WikiText-2 (Merity et al., 2017). For downstream evaluation, we use GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), HellaSwag (Zellers et al., 2019), and MMLU (Hendrycks et al., 2021).

Quantization Setting. All models are quantized under the W4A8 setting, i.e., 4-bit weights and 8-bit activations.

Baselines. We compare CALM with strong PTQ baselines, including GPTQ (Frantar et al., 2022), AWQ (Lin et al., 2024), SmoothQuant (Xiao et al., 2023), SpinQuant (Liu et al., 2025), and FlatQuant (Sun et al., 2025).

Calibration. We use 128 randomly sampled sequences from the training split of C4 with a sequence length of 2048 tokens for calibration. The same calibration set, sequence length, group size, and activation collection procedure are used across all compared methods unless otherwise specified.

4.2 CKA-Guided Modular Quantization Results

Figure 3 illustrates the distribution of CKA-preferred quantization methods selected by CALM across Llama and Qwen models of different scales, including Llama-3.1-8B, Llama-3.2-3B, Llama-3.2-1B, Qwen2.5-0.5B, and Qwen2.5-1.5B. Under the evaluated candidate pool, the selected method varies across network depth, and no single PTQ method is selected for all layers. Instead, SmoothQuant, AWQ, and GPTQ appear in interleaved layer-wise patterns.

This pattern is consistent with our motivation that different layers may favor different quantization mechanisms. Some shallow layers tend to select SmoothQuant, suggesting stronger sensitivity to activation-side outliers, whereas AWQ or GPTQ is more often selected in many middle and deeper layers, suggesting greater sensitivity to salient weight channels or reconstruction errors. This should not be interpreted as a rigid depth-wise rule, since the preference varies across model families and layer positions. CALM therefore selects methods layer by layer based on representation preservation rather than relying on a manually specified assignment rule.

Overall, Figure 3 provides empirical evidence that the CKA-preferred PTQ method varies across layers in the evaluated models. This supports layer-wise method selection as a useful alternative to enforcing a single uniform PTQ pipeline under the same bit-width.

4.3 Perplexity Evaluation on LLMs

We evaluate perplexity (PPL, lower is better) for different quantization methods across multiple LLMs. The compared methods include FP16, AWQ, GPTQ, SmoothQuant, SpinQuant, FlatQuant, and our proposed approach, CALM. We use C4 and WikiText-2 (Wiki2) to assess performance on web text and relatively high-quality written text, respectively. Table 1 summarizes the perplexity results of different quantization methods across multiple LLMs.

Overall, all quantization methods increase PPL relative to the FP16 baseline, but the magnitude of degradation depends on the model size, architecture, and quantization method. Model size has a clear influence on quantization robustness. For larger models such as Llama-3.1-8B, the increase in PPL on both C4 and Wiki2 remains relatively modest across different methods, indicating stronger tolerance to quantization noise. In contrast, smaller models such as Llama-3.2-1B and Qwen2.5-0.5B show much larger PPL increases under the same setting, suggesting that they are more sensitive to quantization-induced errors.

Method	Llama-3.1-8B		Llama-3.2-1B		Llama-3.2-3B		Qwen2.5-1.5B		Qwen2.5-0.5B	
	C4 ↓	Wiki2 ↓	C4 ↓	Wiki2 ↓	C4 ↓	Wiki2 ↓	C4 ↓	Wiki2 ↓	C4 ↓	Wiki2 ↓
FP16	12.28	6.41	22.48	11.41	17.24	9.53	15.95	10.50	22.34	13.52
AWQ	13.56	8.12	23.97	12.23	18.89	9.91	17.26	14.05	25.98	16.21
GPTQ	14.12	8.81	24.07	12.18	18.99	9.82	17.54	14.50	26.04	17.03
SmoothQuant	13.64	7.32	24.01	12.10	18.87	9.80	17.33	13.68	25.34	15.67
SpinQuant	13.39	7.48	23.56	11.89	18.77	9.75	16.97	13.31	25.57	15.34
FlatQuant	13.05	7.16	23.12	11.78	17.54	**9.61**	16.77	13.02	25.12	15.12
CALM	**12.72**	**6.89**	**22.88**	**11.70**	**17.41**	9.63	**16.59**	**12.87**	**24.87**	**14.87**

Table 1: Perplexity (PPL) evaluation. We compare CALM with the FP16 baseline and SOTA PTQ methods.

CALM achieves the best or near-best perplexity performance across almost all evaluated settings. For example, on Llama-3.1-8B, CALM attains a C4 PPL of 12.72, which is the lowest among all quantized variants and remains close to the FP16 baseline of 12.28. On Qwen2.5-1.5B, CALM achieves the best Wiki2 PPL of 12.87, improving over SpinQuant (13.31) by 0.44 points. Even on the more quantization-sensitive Qwen2.5-0.5B model, CALM still obtains the lowest C4 PPL of 24.87, compared with 25.98 for AWQ and 26.04 for GPTQ.

4.4 Downstream Evaluation

To evaluate whether quantized models preserve practical task capability beyond language modeling, we report results on four representative downstream benchmarks. Following standard evaluation settings, we use 8-shot evaluation on GSM8K, 0-shot evaluation on HumanEval, 10-shot evaluation on HellaSwag, and 5-shot evaluation on MMLU. Accuracy is used as the primary metric, where higher values indicate better performance. Table 2 summarizes the results.

Overall, quantization reduces downstream accuracy relative to the FP16 baseline, with reasoning and code-generation tasks being more sensitive than HellaSwag and MMLU. For example, on Llama-3.1-8B, GSM8K accuracy drops from 77.17 under FP16 to 74.33 under CALM and 71.89 under GPTQ, while HumanEval drops from 60.71 to 59.67 under CALM and 58.45 under GPTQ. This suggests that more demanding generation tasks are particularly sensitive to quantization-induced layer-wise perturbations.

Across most models and tasks, CALM achieves the best or competitive performance among quantized methods. On Llama-3.2-3B, CALM obtains the best results across all four downstream tasks, reaching 60.22 on GSM8K, 47.02 on HumanEval, 69.95 on HellaSwag, and 59.27 on MMLU. On Qwen2.5-1.5B, CALM also achieves the strongest GSM8K, HumanEval, HellaSwag, and MMLU scores among quantized variants. Even on the smaller Qwen2.5-0.5B model, CALM improves GSM8K accuracy to 25.12, compared with 21.87 for AWQ and 20.22 for GPTQ.

4.5 Performance at Lower Bit-widths

We further evaluate CALM under a more challenging 3-bit setting. As shown in Table 3, reducing the precision to 3 bits causes substantial degradation for all methods, although the severity varies across quantization strategies.

Conventional PTQ baselines often deteriorate sharply at 3 bits, especially on reasoning and code-generation tasks. In contrast, CALM remains consistently stronger than the competing quantized baselines under the same setting. On Llama-3.1-8B, CALM achieves 13.55/7.60 PPL on C4/Wiki2 and 61.50/50.50/74.50/60.20 on GSM8K/HumanEval/HellaSwag/MMLU, outperforming all other 3-bit baselines in Table 3. A similar trend is observed on Llama-3.2-3B, where CALM again delivers the best overall performance among the compared quantized methods.

These results suggest that layer-wise method selection remains beneficial even in more aggressive low-bit regimes. Rather than relying on a single PTQ method across the entire network, CALM better adapts to layer-level sensitivity and therefore degrades more gracefully at lower bit-widths.

4.6 Configuration Experiments

We conduct configuration studies on similarity metrics, candidate diversity, method-versus-bit heterogeneity, and greedy layer-wise selection. Additional granularity, efficiency, and extended search analyses are deferred to the appendix A.5.

Method	Llama-3.1-8B				Llama-3.2-3B				Llama-3.2-1B				Qwen2.5-1.5B				Qwen2.5-0.5B			
	GSM	HE	HS	MMLU	GSM	HE	HS	MMLU	GSM	HE	HS	MMLU	GSM	HE	HS	MMLU	GSM	HE	HS	MMLU
FP16	77.17	60.71	77.39	67.31	68.86	48.80	70.05	61.44	36.44	32.14	59.94	46.38	58.03	35.11	65.20	59.19	33.18	26.78	49.65	45.76
AWQ	72.94	58.92	76.78	64.41	57.65	46.69	68.97	58.69	26.44	22.32	56.78	40.23	48.32	26.31	64.12	57.32	21.87	18.76	48.12	43.26
GPTQ	71.89	58.45	76.91	60.54	55.75	46.23	68.35	57.16	24.24	21.43	57.04	40.58	47.12	24.40	63.73	56.91	20.22	19.04	47.38	42.82
SmoothQuant	72.55	58.95	76.83	64.55	56.33	45.97	69.34	58.73	25.87	22.64	56.89	41.32	48.64	25.97	63.98	57.69	22.34	19.84	47.69	43.67
SpinQuant	73.56	59.24	77.23	64.98	57.93	46.12	69.83	58.31	26.13	22.75	57.68	42.64	49.01	26.87	64.87	57.84	23.45	20.64	47.98	43.44
FlatQuant	74.09	59.44	**77.59**	65.23	58.33	46.72	69.85	58.88	27.02	**23.43**	58.01	43.16	49.66	27.64	64.64	57.92	24.11	21.63	48.12	43.88
CALM	**74.33**	**59.67**	77.13	**65.87**	**60.22**	**47.02**	**69.95**	**59.27**	**27.44**	23.21	**58.53**	**43.50**	**51.36**	**29.34**	**64.96**	**58.11**	**25.12**	**22.04**	**48.68**	**44.12**

Table 2: Downstream task performance across five LLMs. Higher accuracy (↑) indicates better performance. (GSM=GSM8K, HE=HumanEval, HS=HellaSwag). Best results are highlighted.

Method	C4↓	W2↓	GSM↑	HE↑	HS↑	MMLU↑
Llama-3.1-8B (3-bit)						
FP16	12.28	6.41	77.17	60.71	77.39	67.31
SpinQuant	14.40	8.55	57.00	46.20	72.10	58.50
SmoothQuant	15.10	8.60	56.50	45.80	71.80	57.20
AWQ	15.80	10.20	55.00	42.10	69.50	54.20
GPTQ	19.25	13.00	53.00	35.50	65.20	45.00
FlatQuant	13.87	7.94	58.97	47.34	73.22	59.32
CALM	**13.55**	**7.60**	**61.50**	**50.50**	**74.50**	**60.20**
Llama-3.2-3B (3-bit)						
FP16	17.24	9.53	68.86	48.80	70.05	61.44
SpinQuant	20.20	11.40	52.50	34.20	64.10	55.00
SmoothQuant	20.80	11.85	50.20	32.50	63.50	53.80
AWQ	21.50	12.50	52.00	28.40	60.20	50.50
GPTQ	27.22	20.34	45.00	20.50	55.80	45.00
FlatQuant	19.21	10.97	54.22	36.37	65.44	56.49
CALM	**18.45**	**10.65**	**60.12**	**38.00**	**66.50**	**57.50**

Table 3: Comparison of W3A8 quantization performance across different quantization methods. GSM, HE, and HS denote GSM8K, HumanEval, and HellaSwag, respectively.

Selection Metric	Llama-3.1-8B			Qwen2.5-1.5B		
	W2↓	GSM↑	MMLU↑	W2↓	GSM↑	MMLU↑
FP16	6.41	77.17	67.31	10.50	58.03	59.19
Random	7.33	71.22	63.48	13.91	48.39	56.37
MSE	7.05	72.15	63.90	13.54	49.27	56.79
Cosine	7.08	72.64	63.88	13.59	49.31	57.03
CKA	**6.89**	**74.33**	**65.87**	**12.87**	**51.36**	**58.11**

Table 4: Impact of similarity metrics for layer-wise PTQ method selection. All quantized variants use the same candidate pool, bit-width, calibration set, and greedy assembly strategy.

4.6.1 Impact of Similarity Metric Selection

CALM relies on a similarity metric to select the most suitable PTQ method for each layer. To verify whether the improvement comes from CKA-guided selection rather than merely mixing heterogeneous quantization methods, we compare Linear CKA with random assignment and two commonly used activation-level similarity metrics: mean squared error (MSE) and cosine similarity. All variants use the same PTQ candidate pool, bit-width, calibration set, and greedy layer-wise assembly strategy.

As shown in Table 4, CKA achieves the best overall performance among all quantized variants on both model families. Random assignment performs noticeably worse, indicating that CALM's gains do not simply come from combining multiple PTQ methods, but from selecting suitable methods for individual layers. MSE improves over random assignment, but it is sensitive to point-wise activation magnitude and can be dominated by outlier channels. Cosine similarity captures directional agreement between activations, but it may ignore broader structural changes in the representation space. In contrast, CKA measures representation-level alignment and is invariant to isotropic scaling and orthogonal transformations, making it a more effective proxy for layer-wise PTQ method selection in our setting.

4.6.2 Impact of PTQ Candidate Diversity

CALM uses $\{\text{SmoothQuant}, \text{AWQ}, \text{GPTQ}\}$ as the default PTQ candidate pool, covering activation-side smoothing, activation-aware salient-channel protection, and reconstruction-based weight quantization. To examine whether CALM is restricted to this default pool or can naturally incorporate stronger PTQ operators, we further expand the candidate pool by adding FlatQuant and SpinQuant. All variants use the same W4A8 setting, calibration data, CKA-guided greedy selection strategy, and evaluation protocol. FP16 is reported as the full-precision reference, while Original CALM denotes the default configuration used in the main experiments.

As shown in Table 5, expanding the PTQ candidate pool consistently improves over Original CALM across the evaluated models and metrics. Adding either FlatQuant or SpinQuant alone brings moderate gains, suggesting that transformation-

Candidate Pool / Method	C4↓	W2↓	GSM↑	MMLU↑
Llama-3.1-8B				
FP16	12.28	6.41	77.17	67.31
Original CALM	12.72	6.89	74.33	65.87
CALM + FlatQuant	12.58	6.80	74.45	65.92
CALM + SpinQuant	12.66	6.81	74.87	65.91
CALM + FlatQuant + SpinQuant	**12.51**	**6.73**	**75.21**	**66.23**
Llama-3.2-3B				
FP16	17.24	9.53	68.86	61.44
Original CALM	17.41	9.63	60.22	59.27
CALM + FlatQuant	17.41	9.61	61.76	59.71
CALM + SpinQuant	17.35	9.61	61.98	59.73
CALM + FlatQuant + SpinQuant	**17.31**	**9.58**	**62.44**	**60.04**
Qwen2.5-1.5B				
FP16	15.95	10.50	58.03	59.19
Original CALM	16.59	12.87	51.36	58.11
CALM + FlatQuant	16.52	12.81	52.01	58.23
CALM + SpinQuant	16.51	12.84	51.89	58.34
CALM + FlatQuant + SpinQuant	**16.32**	**12.72**	**52.34**	**58.75**
Qwen2.5-0.5B				
FP16	22.34	13.52	33.18	45.76
Original CALM	24.87	14.87	25.12	44.12
CALM + FlatQuant	24.77	14.76	26.32	44.48
CALM + SpinQuant	24.70	14.58	26.15	44.52
CALM + FlatQuant + SpinQuant	**24.39**	**14.29**	**27.97**	**44.79**

Table 5: Effect of expanding the PTQ candidate pool across four LLMs. Original CALM uses the default candidate pool $\{\text{SmoothQuant}, \text{AWQ}, \text{GPTQ}\}$. Expanded variants add FlatQuant, SpinQuant, or both under the same W4A8 setting, calibration data, and CKA-guided greedy selection strategy. GSM = GSM8K.

based PTQ operators provide complementary layer-wise choices beyond the default candidate pool. Combining both FlatQuant and SpinQuant yields the strongest quantized results in all four models, indicating that CALM is not tied to a fixed set of three PTQ methods and can naturally benefit from a richer candidate pool.

At the same time, the default candidate pool already provides a strong accuracy–efficiency trade-off, since it avoids additional transformation-related runtime costs while remaining competitive. Therefore, we use $\{\text{SmoothQuant}, \text{AWQ}, \text{GPTQ}\}$ as the main configuration throughout the paper, and view the expanded candidate pool as an optional extension when additional calibration-time search or transformation-based operators are acceptable.

4.6.3 Method-Heterogeneity and Bit-Heterogeneity

Mixed-precision quantization and CALM address different adaptation dimensions: mixed-precision methods vary the bit-width across layers or channels, whereas CALM keeps the target bit-width fixed and varies the PTQ mechanism assigned to each layer. To provide a controlled comparison under a similar average 4-bit budget, we compare CALM with representative bit-heterogeneous methods, including HAWQ-V2 (Dong et al., 2020) and SpQR (Dettmers et al., 2024).

Method	Wiki2 ↓	GSM8K ↑	Latency (ms/token) ↓
Llama-3.1-8B (Seq=2048)			
HAWQ-V2	7.25	73.70	5.72
SpQR	6.97	74.22	8.16
CALM	**6.89**	**74.33**	**4.38**
Qwen2.5-1.5B (Seq=2048)			
HAWQ-V2	13.23	49.83	1.58
SpQR	12.92	51.16	2.24
CALM	**12.87**	**51.36**	**1.15**

Table 6: Comparison between Bit-Heterogeneity and CALM under the same average 4-bit constraint.

As shown in Table 6, CALM achieves a favorable accuracy-latency trade-off in our setting. Unlike mixed-precision methods that allocate different numerical capacities across layers, CALM maintains 4-bit weights globally. It adapts the quantization mechanism itself, selecting different PTQ algorithms for layers with different representation and quantization characteristics. These results do not imply that bit heterogeneity is generally inferior; rather, they show that method heterogeneity is a complementary and competitive adaptation dimension under fixed low-bit deployment constraints.

5 Conclusion

We presented CALM, a CKA-guided framework for layer-wise PTQ method selection under fixed low-bit constraints. CALM treats existing PTQ algorithms as candidate operators, selects CKA-preferred methods for individual Transformer layers, and statically assembles a method-heterogeneous quantized model without retraining or runtime routing. Experiments on Llama and Qwen models show that CALM improves over uniform single-method PTQ baselines in perplexity and often preserves stronger downstream performance, with negligible measured inference overhead. These findings suggest that method heterogeneity is a useful adaptation dimension complementary to bit-width allocation in LLM post-training quantization.

Limitations

CALM has several limitations. First, it relies on a predefined PTQ candidate pool, so its performance is bounded by the diversity and quality of the available quantization methods. If none of the candidates is suitable for a particular layer, layer-wise selection may provide limited benefit. Second, CALM uses CKA as a label-free representation fidelity proxy rather than directly optimizing perplexity or downstream accuracy. Although our experiments show that CKA is effective in the evaluated settings, the connection between representation alignment and task performance may vary across models, tasks, and quantization regimes. Third, CALM adopts greedy layer-wise selection for efficiency, which may not fully capture long-range inter-layer dependencies, especially with larger candidate pools or more aggressive quantization settings.

References

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2025. Systematic outliers in large language models. *ArXiv*, abs/2502.06415.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. 2024. Quarot: Outlier-free 4-bit inference in rotated llms. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Krishna Teja Chitty-Venkata, Jie Ye, and Murali Emani. 2025. Mopeq: Mixture of mixed precision quantized experts. In *IEEE/CVF International Conference on Computer Vision, ICCV 2025 - Workshops, Honolulu, HI, USA, October 19-20, 2025*, pages 4082–4091. IEEE.

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. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2024. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, Vienna, Austria.

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. In *Advances in Neural Information Processing Systems*, volume 33, pages 18518–18529. Curran Associates, Inc.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2019. HAWQ: Hessian AWare quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 293–302.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23:120:1–120:39.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. OPTQ: accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. The language model evaluation harness.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. 2019. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 3519–3529.

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. In *Proceedings of the Seventh Annual Conference on Machine Learning and Systems (MLSys)*.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. 2025. SpinQuant: LLM quantization with learned rotations. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Meta AI. 2024. Llama 3.2: Model card and prompt formats. `https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_2/`. Accessed: 2026-05-26.

Markus Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. 2020. Up or down? Adaptive rounding for post-training quantization. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 7197–7206.

Sangki Park and Ki-Seok Chung. 2026. Spinout: Enhanced rotation-based quantization for llm by outlier injection. *IEEE Access*, 14:24082–24095.

Enrique Queipo-de-Llano, Alvaro Arroyo, Federico Barbero, Xiaowen Dong, Michael M. Bronstein, Yann LeCun, and Ravid Shwartz-Ziv. 2025. Attention sinks and compression valleys in llms are two sides of the same coin. *CoRR*, abs/2510.06477.

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. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Rahul Raman, Khushi Sharma, and Sai Qian Zhang. 2025. Rethinking the outlier distribution in large language models: An in-depth study. *ArXiv*, abs/2505.21670.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2024. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

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. 2025. Flatquant: Flatness matters for llm quantization. *Preprint*, arXiv:2410.09426.

Wei Tao, Haocheng Lu, Xiaoyang Qu, Bin Zhang, Kai Lu, Jiguang Wan, and Jianzong Wang. 2025. Moqae: Mixed-precision quantization for long-context LLM inference via mixture of quantization-aware experts. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 10810–10820. Association for Computational Linguistics.

Llama Team. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Xiuying Wei, Ruihao Gong, Yuhang Li, Xianglong Liu, and Fengwei Yu. 2022. QDrop: Randomly dropping quantization for extremely low-bit post-training quantization. In *Proceedings of the Tenth International Conference on Learning Representations (ICLR)*.

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 *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 38087–38099.

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, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

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 Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July*

28- August 2, 2019, Volume 1: Long Papers, pages 4791–4800.

Jinhao Zhang, Yunquan Zhang, Boyang Zhang, Zeyu Liu, and Daning Cheng. 2025. Moqe: Improve quantization model performance via mixture of quantization experts. *CoRR*, abs/2508.09204.

A Appendix

Appendix for CALM: A CKA-Guided Adaptive Layer-Wise Modularization Framework for LLM Quantization

We organize the appendix into six sections:

- Code availability is provided in Appendix A.1;
- Additional details and diagnostic evidence for Linear CKA are provided in Appendix A.2;
- Quantization configuration details are provided in Appendix A.3;
- The impact of quantization granularity is analyzed in Appendix A.4;
- The impact of PTQ candidate diversity is analyzed in Appendix A.5;
- Inference overhead and latency results are provided in Appendix A.6;
- Fine-grained hyperparameter tuning results are provided in Appendix A.7;
- Additional search strategies and offline search costs are provided in Appendix A.8.

A.1 Code Availability

https://anonymous.4open.science/r/CALM-ACL

A.2 CKA as a Representation Fidelity Proxy

This section provides additional details on the use of Linear Centered Kernel Alignment (CKA) as the fidelity proxy in CALM. CKA is used only as a label-free criterion for offline layer-wise method selection. We use it as a practical proxy for measuring how well a quantized layer preserves the representation produced by its full-precision counterpart.

A.2.1 Linear CKA Definition

At Transformer layer l, let $\mathbf{X}_{\mathrm{FP}}^{(l)} \in \mathbb{R}^{n\times d}$ denote the full-precision block output hidden states on the calibration set, and let $\mathbf{Y}_m^{(l)} \in \mathbb{R}^{n\times d}$ denote the corresponding hidden states produced by candidate quantization method m. Here, n is the number of calibration tokens and d is the hidden dimension. In our experiments, CKA is computed on the Transformer block output before it is passed to the next

block, so $d = d_{\text{model}}$. This choice directly measures the representation consumed by subsequent layers and is architecture-agnostic across Llama and Qwen models.

We define the centering matrix as

$$\mathbf{C} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top,$$

where $\mathbf{I}_n$ is the identity matrix and $\mathbf{1}$ is an all-one column vector. The centered feature matrices are

$$\tilde{\mathbf{X}} = \mathbf{C}\mathbf{X}_{\text{FP}}^{(l)}, \qquad \tilde{\mathbf{Y}} = \mathbf{C}\mathbf{Y}_m^{(l)}.$$

The Linear CKA score is then computed as

$$\text{CKA}\left(\mathbf{X}_{\text{FP}}^{(l)}, \mathbf{Y}_m^{(l)}\right) = \frac{\left\|\tilde{\mathbf{Y}}^\top \tilde{\mathbf{X}}\right\|_F^2}{\left\|\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}\right\|_F \left\|\tilde{\mathbf{Y}}^\top \tilde{\mathbf{Y}}\right\|_F}, \tag{2}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. A higher CKA score indicates stronger alignment between the full-precision and quantized representations.

A.2.2 Why CKA is Suitable for CALM

MSE is a natural metric for measuring the numerical discrepancy between full-precision and quantized activations. Given a full-precision representation $\mathbf{X}$ and a quantized representation $\mathbf{Y}$, MSE can be written as

$$\text{MSE}(\mathbf{X}, \mathbf{Y}) = \frac{1}{nd}\|\mathbf{X} - \mathbf{Y}\|_F^2. \tag{3}$$

However, LLM quantization should not treat all channels uniformly. The motivation behind AWQ suggests that a small number of activation-induced salient channels can have a disproportionate influence on model behavior. Therefore, preserving these critical channels can be more important than simply minimizing the average reconstruction error over all dimensions. In other words, the difficulty of low-bit quantization lies not only in reducing the overall numerical deviation, but also in preserving the representation structures associated with salient channels and activation outliers.

This makes MSE less suitable as the sole selection criterion in CALM. Although MSE measures the average discrepancy between two activation tensors, it does not explicitly model channel salience or outlier-related structures. As a result, when CALM compares heterogeneous PTQ candidates, MSE may favor methods with smaller local reconstruction errors while underestimating methods that better preserve critical representation structures through salient-channel protection or activation-outlier mitigation.

CALM therefore uses Linear CKA as a representation-level fidelity proxy. Let $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ denote the centered full-precision and quantized representations, respectively. Linear CKA is defined as

$$\text{CKA}(\mathbf{X}, \mathbf{Y}) = \frac{\left\|\tilde{\mathbf{Y}}^\top \tilde{\mathbf{X}}\right\|_F^2}{\left\|\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}\right\|_F \left\|\tilde{\mathbf{Y}}^\top \tilde{\mathbf{Y}}\right\|_F}. \tag{4}$$

Compared with MSE, CKA focuses on normalized representation-level alignment rather than raw numerical discrepancy. It is therefore better suited for comparing PTQ candidates whose benefits may come from different mechanisms, including local reconstruction, salient-channel protection, and activation-outlier smoothing.

Finally, CALM computes the CKA score using the current hybrid-prefix activation rather than the full-precision prefix activation. Specifically, the candidate representation $\mathbf{Y}_m^{(l)}$ is obtained by feeding the current quantized prefix activation $\mathbf{H}_{\text{Q}}^{(l-1)}$ into the candidate quantized layer. Therefore, the CKA score at layer l reflects both the local fidelity of the candidate method and the distribution shift accumulated from previously selected quantized layers. This makes the CKA-based selection criterion consistent with the sequential construction of the final heterogeneous quantized model.

A.2.3 Diagnostic Evidence

To further examine whether CKA reflects representation-level degradation after quantization, we conduct a diagnostic experiment based on linear reconstruction. This experiment is used only for analysis and is not part of the CALM calibration or inference pipeline. Specifically, under a homogeneous quantization method, we identify a layer with a low CKA score and test whether partially restoring its representation alignment is associated with improved model performance.

Let $\mathbf{X}_{\text{FP}} \in \mathbb{R}^{B\times D}$ and $\mathbf{X}_{\text{Q}} \in \mathbb{R}^{B\times D}$ denote the full-precision and quantized activation outputs at this layer, where B is the number of calibration tokens and D is the hidden dimension. We fit a linear transformation matrix $\mathbf{M}$ that maps the quantized representation toward the full-precision representation by solving

$$\min_{\mathbf{M}} \|\mathbf{X}_{\text{FP}} - \mathbf{X}_{\text{Q}}\mathbf{M}\|_F^2. \tag{5}$$

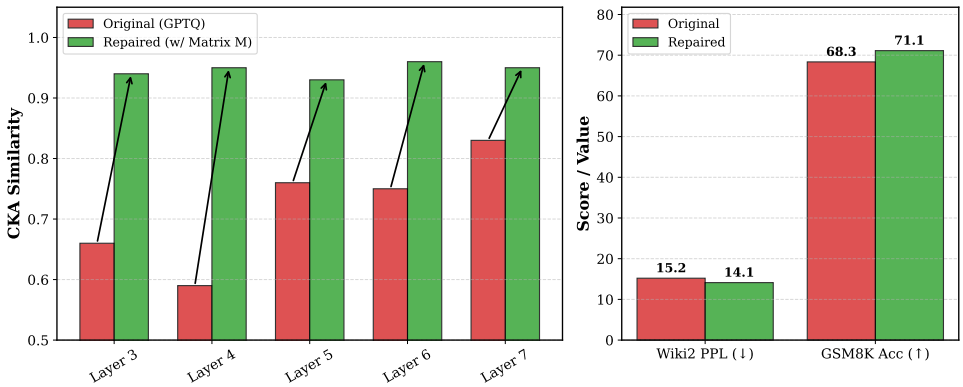

Figure 4: Diagnostic experiment showing that partially restoring the feature space of a low-CKA layer increases representation alignment and improves model-level metrics. This repair is used only for analysis and is not included in the CALM inference pipeline.

This least-squares problem can be solved by standard linear regression, for example using the pseudo-inverse of $\mathbf{X}_{\mathrm{Q}}$. When the dimensions are compatible with the corresponding projection, the fitted transformation can be folded into the quantized weights as $\mathbf{W}'_{\mathrm{Q}} \leftarrow \mathbf{W}_{\mathrm{Q}}\mathbf{M}$. Again, this repair operation is only used to analyze the relationship between representation alignment and model performance.

As shown in Figure 4, applying the fitted transformation increases the CKA score of the selected low-alignment layer. We also observe corresponding improvements in model-level metrics, including lower perplexity and higher GSM8K accuracy. These results suggest that low CKA scores can indicate meaningful representation disruption caused by quantization, and that improving representation alignment at such layers can partially recover model performance.

This diagnostic experiment does not prove a causal or globally optimal relationship between CKA and downstream accuracy. Instead, it provides empirical evidence that CKA is an informative label-free proxy for representation fidelity: layers with low CKA tend to exhibit stronger representation distortion, while methods that better preserve CKA tend to produce stronger quantized models in our experiments.

A.3 Quantization Configuration

To ensure fair comparison and reproducibility, we adopt a unified quantization standard across all methods and ablation studies. Specifically, all models are quantized to 4-bit weights with 8-bit activations (W4A8) using a group size of 128. This rigorous standardization ensures that observed performance gains stem exclusively from algorithmic efficacy rather than hyperparameter variations.

Method	C4 ↓	Wiki2 ↓	GSM8K ↑	HumanEval ↑	HellaSwag ↑	MMLU ↑
Llama-3.1-8B						
Block-8	13.15	7.20	73.05	58.75	76.65	64.70
Block-4	12.89	7.05	73.65	59.10	76.90	65.10
Block-2	12.78	6.94	74.10	59.45	77.05	65.60
CALM	**12.72**	**6.89**	**74.33**	**59.67**	**77.13**	**65.87**
Qwen2.5-1.5B						
Block-8	17.25	13.90	49.10	27.20	63.45	56.90
Block-4	16.95	13.40	50.07	28.10	63.90	57.35
Block-2	16.70	13.05	50.95	28.90	64.35	57.80
CALM	**16.59**	**12.87**	**51.36**	**29.34**	**64.96**	**58.11**

Table 7: Performance comparison of varying quantization granularities.

Furthermore, we keep non-linear and sensitivity-critical components, including normalization layers, embeddings, residual additions, and the KV cache, in FP16/BF16 following common low-bit LLM inference practice. For linear layers, all methods follow the same W4A8 setting: weights are quantized to INT4, and the corresponding activation interfaces use INT8 activations. Within this unified quantization format, CALM focuses on assigning the most suitable PTQ method to each Transformer layer.

A.4 Impact of Quantization Granularity

To investigate the marginal effect of granularity on model fidelity, we compare our proposed layer-wise strategy against coarse-grained block-wise variants. We evaluate three baselines: Block-8, Block-4, and Block-2, where quantization configurations are shared across 8, 4, and 2 consecutive layers, respectively. Table 7 reveals that finer granularity consistently yields superior performance.

Transitioning from Block-8 to Block-2 results in monotonic improvements across tasks. Crucially, CALM surpasses even the finest Block-2 baseline, validating the necessity of per-layer optimization. This superiority stems from the precise handling of sensitive layers characterized by outliers, preventing them from being forced into suboptimal methods inherent in shared block-wise configurations.

A.5 Impact of PTQ Candidate Diversity

Variant	C4↓	W2↓	GSM↑	HE↑	HS↑	MMLU↑
Llama-3.1-8B						
CALM	**12.72**	**6.89**	**74.33**	**59.67**	**77.13**	**65.87**
w/o SQ	13.55	7.65	73.12	58.80	76.50	64.60
w/o AWQ	12.95	7.08	72.85	58.45	76.85	64.20
w/o GPTQ	12.80	6.96	73.98	59.40	77.05	65.50
Llama-3.2-3B						
CALM	**17.41**	**9.63**	**60.22**	**47.02**	**69.95**	**59.27**
w/o SQ	18.50	10.15	58.50	46.10	68.80	58.10
w/o AWQ	17.80	9.85	57.10	45.85	69.10	57.90
w/o GPTQ	17.55	9.70	59.65	46.80	69.60	59.05
Qwen2.5-1.5B						
CALM	**16.59**	**12.87**	**51.36**	**29.34**	**64.96**	**58.11**
w/o SQ	17.45	14.20	49.50	27.50	63.20	56.80
w/o AWQ	16.90	13.45	48.95	27.10	64.05	56.50
w/o GPTQ	16.70	13.05	50.90	29.05	64.40	57.85

Table 8: Ablation study on PTQ candidate diversity across three LLMs. We report perplexity on C4 and WikiText-2, and downstream performance on GSM8K, HumanEval, HellaSwag, and MMLU. Higher accuracy (↑) indicates better performance.

A core assumption behind CALM is that different layers may favor different quantization methods. To evaluate this design choice, we remove one PTQ method at a time from the candidate pool and measure the resulting performance change. Table 8 shows that the full candidate pool consistently yields the strongest overall performance across models and tasks.

For example, on Llama-3.1-8B, CALM achieves a C4 PPL of 12.72, whereas removing SmoothQuant increases it to 13.55. Excluding AWQ also reduces GSM8K accuracy from 74.33 to 72.85. Similar degradations can be observed on other model families and evaluation benchmarks, indicating that no single candidate is entirely redundant in the final selection process.

These results suggest that the candidate PTQ methods provide complementary benefits rather than interchangeable ones. Maintaining a diverse candidate pool, therefore, improves CALM's ability to match different layers with suitable quantization strategies, which in turn contributes to better overall perplexity and downstream performance.

A.6 Negligible Inference Overhead

Method	Fwd. Latency ↓ (ms/token)	Gen. Throughput ↑ (tokens/s)	Fwd. Change	Throughput Change
Llama-3.2-1B (A800, W4A8)				
GPTQ	1.04 ± 0.05	875.2	–	–
CALM	1.05 ± 0.06	869.5	+0.87%	-0.65%
Llama-3.2-3B (A800, W4A8)				
GPTQ	2.50 ± 0.10	398.5	–	–
CALM	2.52 ± 0.12	396.1	+0.80%	-0.60%
Llama-3.1-8B (A800, W4A8)				
GPTQ	4.35 ± 0.15	229.8	–	–
CALM	4.38 ± 0.18	228.3	+0.69%	-0.65%

Table 9: Inference latency and throughput comparison on a single NVIDIA A800 GPU using optimized W4A8 kernels with sequence length 2048. Relative changes are computed with respect to the uniform GPTQ baseline. For throughput, negative values indicate a decrease in tokens per second.

We further evaluate the inference efficiency of CALM using real-world latency measurements on a single NVIDIA A800 GPU. As shown in Table 9, CALM introduces only marginal overhead compared with the uniform GPTQ baseline. Across the evaluated model scales, the forward latency increases by less than 1%, with an average increase of approximately 0.79%. The generation throughput decreases by approximately 0.63% on average. These results indicate that CALM preserves the deployment efficiency of standard low-bit quantization while retaining the accuracy benefits of layer-wise method assignment. Since the final model is statically assembled and uses the same W4A8 runtime format, CALM does not require runtime routing, input-dependent method switching, or method-specific kernel dispatch during inference.

A.7 Fine-Grained Hyperparameter Tuning

The default CALM configuration selects among a fixed set of PTQ algorithms, namely GPTQ, AWQ, and SmoothQuant. In this section, we examine whether CALM can also incorporate method-specific hyperparameters into the offline selection space. We refer to this extended variant as CALM+.

Specifically, we extend the candidate pool from method-level choices to algorithm–parameter variants $\mathcal{P}^+ = \{\mathrm{GPTQ}, \mathrm{AWQ}\} \cup \{\mathrm{SmoothQuant}_\alpha :$

Method	C4 ↓	Wiki2 ↓	GSM8K ↑	HumanEval ↑	HellaSwag ↑	MMLU ↑
Llama-3.1-8B						
CALM	12.72	6.89	74.33	59.67	77.13	65.87
CALM+	**12.61**	**6.72**	**75.35**	**60.05**	**77.26**	**66.32**
Llama-3.2-1B						
CALM	22.88	11.70	27.44	23.21	58.53	43.50
CALM+	**22.45**	**11.58**	**28.52**	**24.15**	**58.90**	**43.95**
Qwen2.5-1.5B						
CALM	16.59	12.87	51.36	29.34	64.96	58.11
CALM+	**16.48**	**12.70**	**52.15**	**29.90**	**65.02**	**58.45**

Table 10: Results of fine-grained hyperparameter tuning (CALM+) on three representative models.

$\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7\}\}$. For SmoothQuant, the hyperparameter α controls the strength of activation-to-weight difficulty migration through the smoothing scale:

$$s = \frac{\max(|X|)^{\alpha}}{\max(|W|)^{1-\alpha}}.$$

All candidates in $\mathcal{P}^{+}$ are evaluated using the same CKA-guided greedy selection rule as CALM. Therefore, CALM+ does not change the inference graph or introduce runtime routing; it only increases the offline calibration search space.

As shown in Table 10, CALM+ provides consistent but moderate improvements over the default CALM configuration across the evaluated models. For example, on Llama-3.2-1B, CALM+ improves GSM8K accuracy from 27.44 to 28.52, while also reducing Wiki2 perplexity from 11.70 to 11.58. Similar trends are observed on Llama-3.1-8B and Qwen2.5-1.5B. These results suggest that method-specific hyperparameters can provide an additional source of layer-wise adaptability.

However, CALM+ also requires a larger calibration-time search space than the default CALM configuration. We therefore use CALM as the main setting in the paper and view CALM+ as an optional extension when additional offline search cost is acceptable.

A.8 Additional Search Strategies and Offline Search Cost

CALM uses Greedy-CKA as the default layer-wise selection strategy due to its favorable accuracy–cost trade-off. To examine whether stronger calibration-time search can further improve layer-wise method assignment, we also evaluate two stronger variants: Beam-CKA and Overlapping Block-CKA. Both variants are used only during offline calibration.

Algorithm 2 Compact Beam-CKA Selection

Require: FP activations $\{\mathbf{H}_{FP}^{(l)}\}_{l=0}^{L}$, PTQ pool $\mathcal{P}$, beam width k
Ensure: Method assignment $a_{1:L}^{*}$

```
B_0 ← {(∅, H_FP^(0), 0)}
for l = 1 to L do
    Expand each beam in B_{l-1} with every m ∈ P
    Compute candidate activations and CKA scores using Eq. (1)
    B_l ← top-k expanded beams by accumulated score
end for
return the highest-scoring assignment in B_L
```

After the assignment is determined, the selected layers are statically assembled into a W4A8 model, so neither variant introduces runtime routing, input-dependent switching, or additional inference-time controllers.

Beam-CKA. Greedy-CKA keeps a single partial assignment and immediately commits the best method for each layer. Beam-CKA instead keeps the top-k partial assignments at each layer. Each beam state consists of a partial method assignment, the corresponding quantized-prefix activation, and the accumulated CKA score. At layer l, each retained beam is expanded with all candidate PTQ methods $m \in \mathcal{P}$, and each expansion is scored by:

$$S_l(m) = \mathrm{CKA}\left(\mathbf{H}_{FP}^{(l)}, f_m^{(l)}(\mathbf{H}_Q^{(l-1)}; \mathbf{W}_l)\right). \quad (6)$$

The accumulated score of a partial assignment is:

$$\mathrm{Score}(a_{1:l}) = \sum_{i=1}^{l} \mathrm{CKA}\left(\mathbf{H}_{FP}^{(i)}, \mathbf{H}_Q^{(i)}(a_{1:i})\right). \quad (7)$$

After each expansion step, only the top-k partial assignments are retained. The final output is the single complete assignment with the highest accumulated score; therefore, Beam-CKA is not an ensemble method.

The forward-level calibration cost of Beam-CKA is approximately $O(kL|\mathcal{P}|)$, compared with $O(L|\mathcal{P}|)$ for Greedy-CKA. In our experiments, we use $k = 2$ as a stronger but still practical search baseline.

Overlapping Block-CKA. Beam-CKA keeps multiple prefixes but still expands one layer at a

time. Block-CKA instead jointly searches over a local window of b consecutive layers to capture short-range inter-layer dependencies. For a block starting at layer s, we define:

$$\mathcal{W}_s = \{s, s+1, \ldots, t\}, \quad t = \min(s+b-1, L). \tag{8}$$

Within this window, Block-CKA enumerates all assignments:

$$a_{s:t} = (m_s, \ldots, m_t) \in \mathcal{P}^{|\mathcal{W}_s|}. \tag{9}$$

Starting from the committed prefix activation, the block assignment is evaluated recursively:

$$\begin{aligned} \mathbf{Z}^{(s-1)} &= \mathbf{H}_Q^{(s-1)}, \\ \mathbf{Z}^{(i)} &= f_{m_i}^{(i)}(\mathbf{Z}^{(i-1)}; \mathbf{W}_i), \quad i = s, \ldots, t. \end{aligned} \tag{10}$$

The block score is the average CKA over the window:

$$\mathrm{Score}(a_{s:t}) = \frac{1}{|\mathcal{W}_s|} \sum_{i=s}^{t} \mathrm{CKA}\left(\mathbf{H}_{FP}^{(i)}, \mathbf{Z}^{(i)}\right). \tag{11}$$

To avoid inconsistent choices in overlapping regions, we use a receding-window commitment rule. After selecting the best assignment for the current block, only the first $b - o$ layers are committed, where o is the overlap size. The remaining o layers are treated as look-ahead context and are re-optimized in the next block. For example, with $b = 3$ and $o = 1$, Block-CKA evaluates layers $(1, 2, 3)$ but commits only layers $(1, 2)$. The next window starts from layer 3 and evaluates $(3, 4, 5)$, so layer 3 is reconsidered with future context.

The direct forward-level cost of Block-CKA is:

$$O\left(\left\lceil \frac{L}{b-o} \right\rceil b|\mathcal{P}|^b\right), \tag{12}$$

where $\lceil L/(b-o) \rceil$ is the number of overlapping windows, b is the number of layers evaluated per window, and $|\mathcal{P}|^b$ is the number of assignments in each block. With prefix caching inside each block, the practical cost can be reduced to:

$$O\left(\left\lceil \frac{L}{b-o} \right\rceil \sum_{i=1}^{b} |\mathcal{P}|^i\right), \tag{13}$$

but it still grows exponentially with block size b.

In our setting, $|\mathcal{P}| = 3$, $b = 3$, and $o = 1$. Thus, each block contains up to $3^3 = 27$ complete assignments. Even with prefix caching, each window evaluates approximately $3 + 9 + 27 = 39$ intermediate branches. By comparison, Beam-CKA with $k = 2$ expands only $k|\mathcal{P}| = 6$ branches per layer. Therefore, Block-CKA is more expensive than Beam-CKA, although it captures short-range inter-layer interactions more explicitly.

Algorithm 3 Compact Overlapping Block-CKA Selection

Require: FP activations $\{\mathbf{H}_{FP}^{(l)}\}_{l=0}^{L}$, PTQ pool $\mathcal{P}$, block size b, overlap o
Ensure: Method assignment $a_{1:L}^*$

```
s ← 1, a* ← ∅, H_Q^(0) ← H_FP^(0)
while s ≤ L do
    t ← min(s + b − 1, L)
    Enumerate all assignments a_{s:t} ∈ P^{t−s+1}
    Score each assignment by average block CKA in Eq. (11)
    Select the best block assignment a_{s:t}^best
    Commit its first c = min(b − o, L − s + 1) methods to a*
    Update the committed prefix activation and set s ← s + c
end while
return a*
```

Offline search cost. Table 11 reports the offline calibration-time cost of Greedy-CKA, Beam-CKA, and Block-CKA. All measurements are conducted on a single NVIDIA A800 GPU using 128 C4 calibration sequences with a sequence length of 2048. Search time is reported in minutes, and peak memory is reported in GB.

As shown in Table 11, Greedy-CKA has the lowest calibration-time cost because it evaluates only $|\mathcal{P}|$ candidates per layer. Beam-CKA is more expensive because it maintains k partial assignments, while Block-CKA is the most expensive because it enumerates all assignments within each local window. Given the marginal performance gains of Beam-CKA and Block-CKA over Greedy-CKA, we use Greedy-CKA as the default strategy in the main experiments.

A.9 Reproducibility Details

We provide additional implementation and evaluation details to facilitate reproducibility. Unless otherwise specified, CALM and all compared baselines use the same calibration data, quantization group size, activation collection procedure, and evaluation protocol.

Model	Search Strategy	Candidates	Forward Cost	Time (min)	Mem. (GB)
Llama-3.2-1B					
	Greedy-CKA	3	$O(L\lvert\mathcal{P}\rvert)$	18.6	19.4
	Beam-CKA ($k = 2$)	3	$O(kL\lvert\mathcal{P}\rvert)$	35.2	20.8
	Block-CKA ($b = 3, o = 1$)	3	$O(\lceil \frac{L}{b-o} \rceil b\lvert\mathcal{P}\rvert^b)$	48.5	23.5
Llama-3.2-3B					
	Greedy-CKA	3	$O(L\lvert\mathcal{P}\rvert)$	42.8	32.7
	Beam-CKA ($k = 2$)	3	$O(kL\lvert\mathcal{P}\rvert)$	81.7	34.9
	Block-CKA ($b = 3, o = 1$)	3	$O(\lceil \frac{L}{b-o} \rceil b\lvert\mathcal{P}\rvert^b)$	115.4	38.8
Llama-3.1-8B					
	Greedy-CKA	3	$O(L\lvert\mathcal{P}\rvert)$	118.6	71.6
	Beam-CKA ($k = 2$)	3	$O(kL\lvert\mathcal{P}\rvert)$	224.9	74.2
	Block-CKA ($b = 3, o = 1$)	3	$O(\lceil \frac{L}{b-o} \rceil b\lvert\mathcal{P}\rvert^b)$	319.5	78.5
Qwen2.5-1.5B					
	Greedy-CKA	3	$O(L\lvert\mathcal{P}\rvert)$	24.9	23.1
	Beam-CKA ($k = 2$)	3	$O(kL\lvert\mathcal{P}\rvert)$	47.1	24.5
	Block-CKA ($b = 3, o = 1$)	3	$O(\lceil \frac{L}{b-o} \rceil b\lvert\mathcal{P}\rvert^b)$	66.8	27.0

Table 11: Offline calibration-time search cost of CALM under different layer-wise selection strategies. Search time and peak memory are measured on a single NVIDIA A800 GPU using 128 C4 calibration sequences with sequence length 2048. Greedy-CKA is the default strategy used in the main experiments. Beam-CKA keeps the top-k partial assignments during search. Block-CKA jointly searches over overlapping blocks with block size b and overlap size o, resulting in a higher calibration-time cost due to the enumeration of $|\mathcal{P}|^b$ block assignments. All costs are incurred only during offline calibration and do not affect inference latency.

Evaluation protocol. We follow the standard evaluation protocol implemented in lm-eval-harness (Gao et al., 2024) for all downstream tasks. For GSM8K, we use the standard 8-shot chain-of-thought setting with greedy decoding and report exact-match accuracy after answer extraction. For MMLU, we use the standard 5-shot multiple-choice setting and report average accuracy over all subjects. For HellaSwag, we use the 10-shot setting following the default evaluation protocol and report normalized accuracy. For HumanEval, we use 0-shot code generation with greedy decoding, where `temperature=0`, `top_p=1.0`, and the maximum generation length is set to 2048 tokens. We report pass@1 using the official test cases. No task labels, validation-set feedback, task-specific fine-tuning, prompt tuning, or downstream benchmark examples are used during quantization or layer-wise method selection.

Calibration data and leakage prevention. For all PTQ methods, we randomly sample 128 sequences from the training split of C4 with a sequence length of 2048 tokens. The calibration data are used only for collecting activations, estimating quantization statistics, computing CKA scores, and selecting layer-wise PTQ methods. The calibration samples are disjoint from WikiText-2, GSM8K, HumanEval, HellaSwag, and MMLU evaluation data. For C4 perplexity evaluation, we evaluate on a held-out C4 validation subset and ensure that the sampled calibration sequences are excluded from the evaluation subset. All baselines, including GPTQ, AWQ, SmoothQuant, and CALM, use the same calibration samples to ensure a fair comparison.

Quantization hyperparameters. All methods are evaluated under the W4A8 setting, where weights are quantized to 4 bits and activations are quantized to 8 bits. For weight quantization, we use group-wise quantization with a group size of 128. For GPTQ, we use a damping ratio of 0.01, disable activation-order reordering, and apply group-wise weight quantization with the same calibration activations as CALM. For AWQ, we use the standard activation-aware scaling search with a grid over the scaling ratio and fold the selected scaling factors into the corresponding weights before inference. For SmoothQuant, we set the smoothing coefficient to $\alpha = 0.5$ and fold the resulting channel-wise scaling factors into adjacent linear layers before inference. For CALM, the candidate pool is $\{\text{SmoothQuant}, \text{AWQ}, \text{GPTQ}\}$, and each candidate method uses exactly the same hyperparameters as its standalone baseline. Therefore, the performance differences between CALM and the corresponding single-method baselines come from layer-wise method assignment rather than method-specific hyperparameter tuning.

Activation quantization. For activation quantization, we use symmetric INT8 quantization at the input interface of quantized linear layers. Activation scales are computed dynamically at runtime using the current activation tensor, following common W4A8 inference practice. Weights are stored in packed INT4 format, while activations are represented with INT8 values during quantized linear computation. Normalization layers, embedding layers, residual additions, output heads, and KV cache tensors are kept in FP16/BF16, which is consistent with standard low-bit LLM inference implementations. All methods follow the same activation quantization rule to avoid confounding effects from different runtime formats.

Runtime format and kernels. After layer-wise selection, CALM exports a static heterogeneous quantized model in the same W4A8 runtime format as the single-method baselines. Although dif-

ferent layers may be selected from different PTQ methods, all selected layers are represented as packed INT4 weights with INT8 activation interfaces. Offline rescaling factors introduced by AWQ or SmoothQuant are folded into the corresponding weights or adjacent affine parameters before inference. Thus, CALM does not require runtime routing, input-dependent method switching, additional controller modules, or method-specific dispatch. All latency and throughput measurements are conducted on a single NVIDIA A800 GPU using packed INT4 weight kernels with INT8 activation interfaces. We use a batch size of 1 and a sequence length of 2048 for latency evaluation. Before measurement, we run 20 warm-up iterations and then average over 100 measured iterations. We report latency in milliseconds per token and generation throughput in tokens per second.

Randomness and variance. We fix the random seed to 42 for calibration sampling, activation collection, and layer-wise method selection. Given the fixed calibration set and deterministic greedy decoding, downstream evaluation is deterministic for a given quantized model. Therefore, the main source of variance comes from the random choice of calibration samples rather than decoding stochasticity. Unless otherwise specified, the main tables report results from a single quantized model generated with this fixed seed. To reduce accidental variance, all compared methods use the same calibration subset and the same evaluation prompts under the same random seed.