A KL Lens on Quantization: Fast, Forward-Only Sensitivity for Mixed-Precision SSM–Transformer Models

Anonymous CVPR submission

Paper ID 21

Abstract

Deploying Large Language Models (LLMs) on edge devices faces severe computational and memory constraints, limiting real-time processing and on-device intelligence. Hybrid architectures combining Structured State Space Models (SSMs) with transformer-based LLMs offer a balance of efficiency and performance. Aggressive quantization can drastically cut model size and speed up inference, but its uneven effects on different components require careful management. In this work, we propose a lightweight, backpropagation-free, surrogate-based sensitivity analysis framework to identify hybrid SSM–Transformer components most susceptible to quantization-induced degradation. Relying solely on forward-pass metrics, our method avoids expensive gradient computations and retraining, making it suitable for situations where access to in-domain data is limited due to proprietary restrictions or privacy constraints. We also provide a formal analysis showing that the Kullback–Leibler (KL) divergence metric better captures quantization sensitivity for Language modeling tasks than widely adopted alternatives such as mean squared error (MSE) and signal-to-quantization-noise ratio (SQNR). Through extensive experiments on SSM and hybrid architectures, our ablation studies confirm that KL-based rankings align with observed performance drops and outperform alternative metrics. This framework enables the practical deployment of advanced hybrid models on resource-constrained edge devices with minimal accuracy loss. We further validate our approach with real-world on-device profiling on Intel Lunar Lake hardware, demonstrating that KL-guided mixed-precision achieves near-FP16 perplexity with model sizes and throughput competitive with Uniform INT4 on both CPU and GPU execution modes.

1. Introduction

Large Language Models (LLMs) based on the Transformer architecture [17] have achieved remarkable success across NLP tasks, but their deployment on edge devices is constrained by high memory and computation demands. Recently, State Space Models (SSMs) [7] have emerged as efficient alternatives for long sequence modeling, offering $\mathcal{O}(n)$ sequence processing and favorable memory scaling. SSM-based architectures can maintain strong accuracy on sequence tasks while being more hardware-friendly, which makes them attractive for resource-constrained platforms. In parallel, *hybrid models* [4, 6] combining SSM and Transformer components have been proposed to capture the best of both worlds. These hybrids achieve competitive performance with improved efficiency, exemplified by models like Zamba [6] and Hymba [4] that interleave SSM and attention modules.

Despite their efficiency, even SSM-based hybrids at scale can contain hundreds of millions to billions of parameters, remaining impractical for direct deployment on mobile and IoT devices. Quantization [10, 13] is a proven approach to compress models by reducing the numerical precision of weights and activations. 4-bit uniform quantization of Transformers has become standard in industry for inference speed-ups, and recent studies demonstrate that even lower-bit weight quantization has become feasible for LLMs given proper handling of outlier features [2, 20, 22]. However, applying uniform quantization to all parts of a hybrid SSM model can lead to disproportionate accuracy degradation in certain layers. Prior work [15] notes that SSMs exhibit different quantization behavior than Transformers: for instance, the state transition in selective SSMs has highly sensitive internal activations, and SSM outputs contain large magnitude outliers not seen in Transformer outputs. These differences call for a more *nuanced*, layer-wise approach to quantization.

In this paper, we demonstrate that quantizing SSM–Transformer models for edge deployment is both necessary and tractable. While prior mixed-precision techniques [5, 14] have shown success in CNNs and Transformers by using gradient-based methods or signal-to-quantization-noise ratio (SQNR) for sensitivity analysis, these approaches fall short in the context of language modeling. In particu-

lar, we find that SQNR—though effective for convolutional architectures—fails to capture the true sensitivity of components in language models. Our key contribution is a novel, gradient-free sensitivity analysis method tailored for SSM–Transformer architectures. It operates entirely via forward-pass signals and reveals which layers truly require higher precision. This insight enables a post-training quantization pipeline that achieves substantial compression with minimal accuracy degradation, addressing a gap where existing mixed-precision strategies break down.

The remainder of the paper is organized as follows. Section 2 reviews prior work on quantization for LLMs, SSMs, and hybrid architectures. Section 3 presents our forward-pass sensitivity analysis, including per-layer metrics, mixed-precision assignment, and Kendall's τ ranking. Section 4.1 formally shows that KL divergence is a tighter proxy for perplexity than SQNR. Section 5.1 reports ablations on the HYMBA hybrid model at the layer and parameter level. Section 4.2 validates layer-level correlations across Mamba and hybrid architectures, confirming that $\mathrm{KL}_{\mathrm{student}\to\mathrm{teacher}}$ outperforms SQNR and other metrics. Finally, Section 7 demonstrates end-to-end deployment on Intel Lunar Lake hardware, achieving near-FP16 perplexity with latency and throughput competitive with Uniform INT4 on CPU and GPU.

2. Related Work

Prior work on efficient sequence modeling can be viewed along two largely orthogonal axes: **model architecture** and **quantization strategy**. Along the first axis, recent systems include Transformer-based large language models (LLMs), state-space models (SSMs), and hybrids that combine the two. Along the second axis, quantization ranges from **homogeneous** schemes that use a single bit-width throughout the model to **mixed-precision** schemes that assign different precisions to different components based on sensitivity. Together, these two axes provide a useful way to understand how prior work trades off efficiency, accuracy, and hardware friendliness.

Early quantization work focused mainly on *homogeneous quantization*, especially for Transformer models. Uniform low-bit quantization is useful because it maps well to integer hardware and simplifies deployment. Early work in vision showed that carefully calibrated 8-bit quantization can preserve accuracy [10], and later work adapted similar ideas to LLMs using outlier-aware calibration and improved rounding methods [13]. However, large Transformers show strong layerwise variation in sensitivity, with some activations and weights exhibiting strong outlier behavior. As a result, forcing all layers to use the same precision can lead to noticeable degradation unless extra techniques such as per-layer clipping are used [21]. This motivates going beyond one-size-fits-all precision assignment.

Mixed-precision quantization addresses this issue by allocating bit-widths non-uniformly across the network. The main idea is to keep sensitive parts at higher precision while quantizing more robust parts more aggressively. HAWQ is an early example, using Hessian-based sensitivity estimates to assign layerwise precisions and showing strong compression–accuracy trade-offs in CNNs [5]. Similar ideas were later extended to LLMs. For example, Pandey et al. [14] propose a post-training method that uses a calibration set to allocate 4/8-bit weights across LLM layers, outperforming uniform int8 baselines on GLUE [18] and SQuAD [16]. Related work also explores low-bit fine-tuning [3], adaptive post-training rounding [12], and hardware-aware bit allocation [19]. Overall, these works show that mixed precision often gives a better accuracy–efficiency trade-off than homogeneous quantization for LLMs.

Quantization research has also started to extend beyond Transformers to *state-space models*. SSMs offer a different efficiency profile, with linear-time sequence processing and constant-memory recurrent state updates, making them attractive for long-context and edge settings. Mamba shows that modern SSMs can achieve competitive sequence modeling performance at lower compute cost than Transformers on several long-range tasks [7]. Quantization studies for these models are still limited, but existing results already suggest different sensitivity patterns from Transformers. Pierro and Abreu [15] provide one of the first systematic studies of post-training quantization for Mamba-style recurrent LLMs, showing that uniform int8 quantization can retain strong performance when sensitive state-transition components are treated carefully, in some cases with quantization-aware training. At the same time, pushing all components to int4 leads to much larger degradation, suggesting that homogeneous quantization is mainly viable for SSMs at moderate precision.

These observations naturally motivate *hybrid architectures* that combine SSM and Transformer components, where both architectural heterogeneity and precision heterogeneity become important. Recent models such as Hymba and Zamba interleave recurrent state-space computation with attention and feed-forward blocks to capture the efficiency advantages of SSMs while retaining Transformer flexibility [4, 6]. Such models are especially well suited to mixed-precision deployment because their submodules play different computational roles. In practice, recurrent state-transition matrices may need higher precision to preserve long-range dynamics, while feed-forward and attention-related components can often be quantized more aggressively. This makes hybrid models a natural setting for sensitivity-guided precision allocation.

Overall, prior work points to two consistent themes. First, quantization sensitivity is highly non-uniform, mak-

ing homogeneous precision increasingly suboptimal as models become larger and more diverse. Second, this non-uniformity becomes even stronger in hybrid SSM–Transformer architectures, where recurrent and attention-based components have different numerical requirements. Our work builds on this intersection and studies how sensitivity-guided precision assignment can better exploit the structure of hybrid sequence models.

3. Methodology

We propose a lightweight sensitivity analysis method that evaluates each layer or module of a model for quantization robustness *without requiring backpropagation*. In contrast, our method uses forward-pass computations only, making it scalable to very large models and avoiding any need for retraining or fine-tuning during the analysis phase.

Our core idea is to measure how much quantization of a single layer affects the model's output or performance metric. Given a trained model, we perform a series of modified inference passes over a representative dataset:

Algorithm 1 KL-PPL CORRELATION VIA KENDALL τ

Require: Model $\mathcal{M}$ with L layers; quantization function QUANTIZE$(\cdot)$; evaluation dataset $\mathcal{D}$
Ensure: Kendall τ and p-value between KL and PPL rankings

```
Initialize lists: KL ← [], PPL ← []
for ℓ = 1 to L do
    Teacher ← M
    Student ← QUANTIZE(M, ℓ)
    logits_T, PPL_T ← EVALUATE(Teacher, D)
    logits_S, PPL_S ← EVALUATE(Student, D)
    KL_ℓ ← COMPUTEKL(logits_T, logits_S)
    Append KL_ℓ to KL; append PPL_S to PPL
end for
π_KL ← ARGSORT(KL, ↓)
π_PPL ← ARGSORT(PPL, ↓)
(τ, p) ← KENDALLTAU(π_KL, π_PPL)
return (τ, p)
```

Table 1. Sort direction for each metric (↑: ascending, ↓: descending).

Metric	Order
Perplexity	↓
SQNR (dB)	↑
$\mathrm{KL}_{\text{teacher}\to\text{student}}$	↓
$\mathrm{KL}_{\text{student}\to\text{teacher}}$	↓
Δ Cross Entropy	↓

This requires no gradients and is easily parallelizable.

3.1. Efficient Mixed-Precision Assignment

Given sensitivity scores, we assign a low bit-width to most layers and a higher bit-width to the top-k sensitive ones. This yields significant compression with minimal accuracy loss. Given model output and input logits, we perform a comprehensive analysis to discover a correlation between perplexity and metrics discussed in Sec. 4.1.

3.2. Overview of Kendall's τ Ranking

Kendall's τ [9] is a non-parametric statistical measure that quantifies the ordinal correlation between two rankings. Specifically, given two rankings R_1 and R_2 of n elements, Kendall's τ is defined as:

$$\tau(R_1, R_2) = \frac{|\mathcal{C}| - |\mathcal{D}|}{\frac{1}{2}n(n-1)},$$

where $\mathcal{C}$ and $\mathcal{D}$ represent the sets of concordantly and discordantly ranked pairs, respectively. The coefficient ranges from -1 (complete disagreement) to $+1$ (complete agreement), with 0 indicating no correlation. Unlike Pearson's correlation, Kendall's τ relies solely on rank order, making it particularly suitable for assessing ranking-based sensitivity metrics.

4. Sensitivity Scoring in SSMs

4.1. Analytical Comparison of D_{KL} and SQNR as Sensitivity Metrics for Mixed-Precision Language Models

Recent works [14, 21] establish the *signal-to-quantization-noise ratio* (SQNR) as a reliable sensitivity measure for CNNs and transformer-based models. SQNR quantifies the distortion introduced by quantizing model logits compared to the original (teacher) logits, expressed in decibels (dB):

$$\mathrm{SQNR} = 10\log_{10}\frac{\mathbb{E}[\,\ell_{\text{orig}}^2\,]}{\mathbb{E}[(\ell_{\text{orig}} - \ell_{\text{quant}})^2]}.$$

Higher SQNR values signify that quantization retains the original activations more accurately.

However, for autoregressive language models, we demonstrate that the *Kullback–Leibler divergence* [11] (D_{KL}) generalizes better than SQNR. KL divergence measures distributional shifts due to quantization in both directions to account for potential asymmetries:

$$\mathrm{KL}(P\|Q) = \mathbb{E}_P\left[\log P - \log Q\right],$$

where $P = \mathrm{softmax}(\ell_{\text{orig}})$ and $Q = \mathrm{softmax}(\ell_{\text{quant}})$. Higher KL values indicate greater divergence between the original and quantized distributions.

While SQNR captures signal fidelity at the logit level, it does not reliably correspond to downstream performance in language tasks. In contrast, KL divergence provides a tighter link to the change in perplexity (), making it more suitable for sensitivity scoring in autoregressive settings.

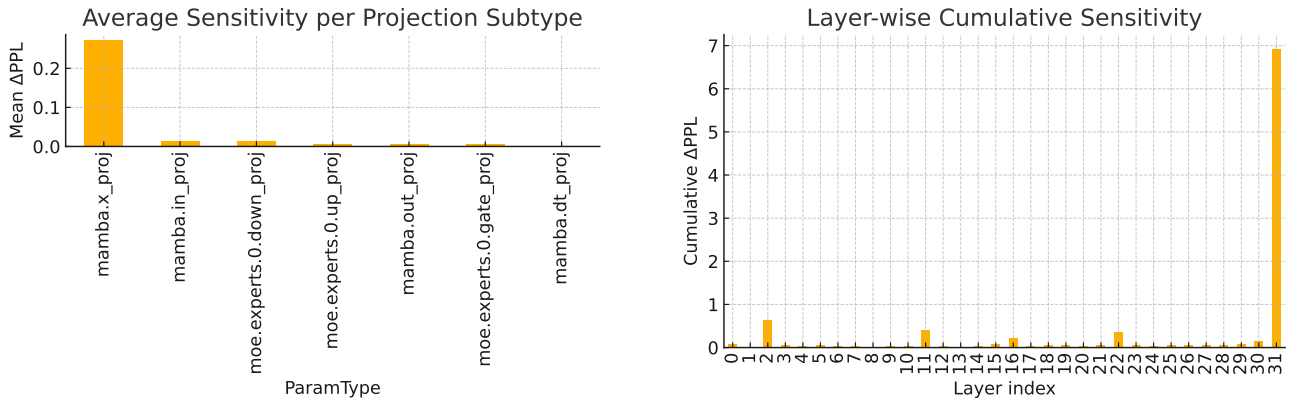

Figure 1. Quantization-sensitivity characteristics of Hymba. **(a)** Average ablation sensitivity $\overline{\Delta}\mathrm{PPL}$ for each projection subtype, with `mamba.x_proj` dominating the profile. **(b)** Layer-wise cumulative sensitivity $\Sigma\Delta\mathrm{PPL}$, showing a sharp peak at block 31.

Notation. Let x be a context and $y \in V$ a token. Throughout,

$$p(y \mid x): \text{ full-precision } \textit{teacher} \text{ model},$$
$$q_\theta(y \mid x): \text{ quantized model}.$$

Expectations $\mathbb{E}_p[\cdot]$ are taken over $(x, y) \sim p$. As all expectations $\mathbb{E}_p[\,\cdot\,]$ are taken with respect to the teacher, the results are *relative* to that uncompressed model.[1]

[Cross-entropy split]

$$(q_\theta, p) = -\mathbb{E}_p[\log q_\theta(y \mid x)] \tag{1}$$
$$= H(p) + D_{\mathrm{KL}}(p \,\|\, q_\theta). \tag{2}$$

[PPL factorization]

$$(q_\theta) = \exp\big((q_\theta, p)\big) \tag{3}$$
$$= (p)\, \exp\big(D_{\mathrm{KL}}(p \,\|\, q_\theta)\big), \tag{4}$$

where $(p) = \exp(H(p))$.

[Teacher-relative PPL bound] If $D_{\mathrm{KL}}(p \,\|\, q_\theta) \le \varepsilon$ then

$$(q_\theta) \le (p)\, e^{\varepsilon}.$$

[SQNR Is Not Monotonic in] There exist logit pairs $\big(z^{(1)}, \hat{z}^{(1)}\big)$ and $\big(z^{(2)}, \hat{z}^{(2)}\big)$ such that

$$\mathrm{SQNR}\big(z^{(1)}, \hat{z}^{(1)}\big) < \mathrm{SQNR}\big(z^{(2)}, \hat{z}^{(2)}\big), \tag{5}$$
$$^{(1)} <^{(2)} . \tag{6}$$

Constant-shift example: Let $\hat{z} = z + c\mathbf{1}$ with any $c \neq 0$.

[1] If we use the test-set distribution for p, the formulas stay the same. The only change is that $H(p)$ now depends on that specific test data.

Since $(z) = (\hat{z})$, the models share the same , but

$$\mathrm{SQNR}(z, \hat{z}) = 10 \log_{10} \frac{\|z\|_2^2}{\|z - \hat{z}\|_2^2} \tag{7}$$
$$= 10 \log_{10} \frac{\|z\|_2^2}{c^2\, \|\mathbf{1}\|_2^2} \tag{8}$$
$$\xrightarrow{|c| \to \infty} -\infty. \tag{9}$$

So SQNR can be made arbitrarily small while is unchanged.

4.2. Metric Asymmetry

We observe a strong asymmetry between the two KL directions used to compare student and teacher distributions. While $KL_{\text{student}\to\text{teacher}}$ aligns well with the perplexity-based sensitivity ranking, the reverse direction $KL_{\text{teacher}\to\text{student}}$ shows a weak negative correlation ($\tau \approx -0.14$).

This difference arises because the two directions penalize different types of errors. $KL_{\text{student}\to\text{teacher}}$ strongly penalizes probability mass that the student assigns to tokens the teacher considers unlikely—precisely the type of distortion introduced by quantization. In contrast, $KL_{\text{teacher}\to\text{student}}$ focuses on whether the student underestimates the teacher's high-probability tokens, which tends to remain relatively stable under quantization. As a result, the reverse KL is less sensitive to the perturbations that drive perplexity degradation.

Practical Implication. As shown above, the *KL divergence gap* $\Delta D_{\mathrm{KL}}(p \,\|\, q_\theta)$ provides an upper bound proxy for perplexity, making it the preferred layer-sensitivity metric for language modelling tasks. Hence, in mixed-precision quantization of language models, precision allocation should be driven by ΔD_{KL}.

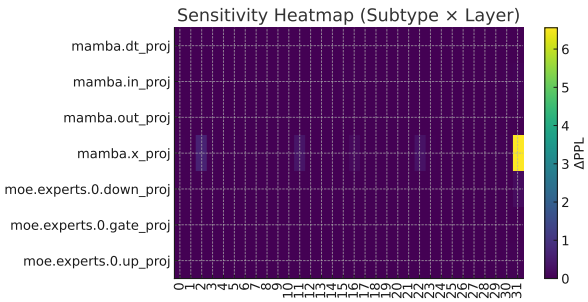

Figure 2. Sensitivity heat-map (ΔPerplexity; rows: projection subtype, columns: layer). Late-stage `x_proj` peaks coincide with moderate MoE projection peaks, underscoring their coupled importance for accurate mixed-precision deployment.

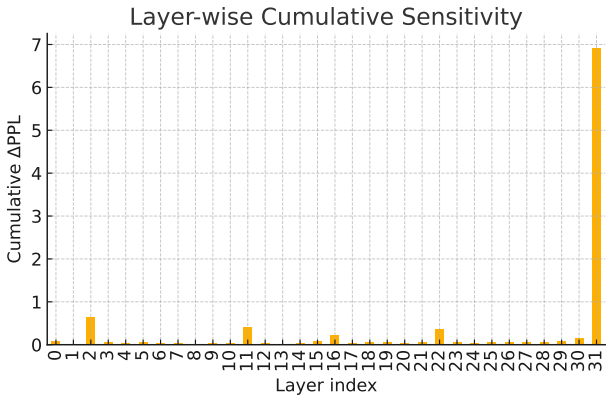

Figure 3. Layer-wise cumulative sensitivity ΣΔPPL, showing a sharp peak at block 31.

5. Ablations Studies on Hymba Hybrid Model

5.1. Layer- and Parameter-Level Quantization Sensitivity in HYMBA

We present a systematic, tensor-level ablation study of the 32-layer HYMBA model [4]. Although similar analyses are common for CNNs and pure-Transformer models, we are not aware of a prior, publicly documented examination of quantization sensitivity in hybrid SSM–Transformer architectures; our findings therefore offer a reference point for subsequent research on this emerging model class. All experiments are conducted under a **uniform 4-bit (INT4) quantization scheme**, a setting that is now widely supported on commercial mobile and server-class accelerators.

Subtype imbalance. Among the seven projection submodules, the cross-projection `mamba.x_proj` exhibits the highest average sensitivity ($\overline{\Delta} = 0.27$), exceeding all other subtypes by more than an order of magnitude ($\overline{\Delta} \leq 0.014$). Conversely, `mamba.dt_proj` is nearly insensitive ($\overline{\Delta} = 3.6 \times 10^{-4}$), suggesting that extremely low-precision representations are feasible (Figure 2).

Localized layer hotspots. Aggregating ΔPerplexity across sub-modules reveals that block 31 alone accounts for over 70% of the aggregate sensitivity budget ($\Sigma\Delta = 6.9$), with smaller peaks in blocks 2, 11, 16, and 22 (Figure 3). Most remaining blocks tolerate aggressive quantization.

Coupled `x_proj`–MoE behaviour. The heat-map in Figure 2 shows that late-stage `x_proj` sensitivities co-occur with moderate MoE `up/down` sensitivities in the same block, indicating that quantization noise in mixing pathways can be amplified by expert routing.

5.2. Ranking Correlation Protocol

We evaluate how effectively each sensitivity metric predicts the quantization impact at a per-layer granularity, defined by the increase in perplexity (↑) when that layer alone is quantized to `int8`. Following prior practice [14], we establish the ground-truth ranking based on Δ and measure the alignment with proxy metrics—namely, SQNR, KL divergences (teacher-to-student and student-to-teacher), and ΔCE—using Kendall's τ coefficient (see Sec. 3.2). Metrics with higher τ values better reflect the true sensitivity of layers.

6. Experiments & Results

6.1. Layer-Level Correlation Analysis

Tab. 2 summarizes the Kendall's τ results across several hybrid SSM models of varying scales. Notably, the student-to-teacher KL divergence consistently achieves the highest correlation, averaging a Kendall's τ of **0.79**, surpassing the widely used SQNR metric (average $\tau = 0.76$). Statistical significance tests (paired one-sided t-tests) further validate that KL divergence significantly outperforms SQNR ($p < 10^{-6}$). This improved correlation is explained by KL divergence operating directly in probability space, aligning closely with the downstream metric of interest (perplexity), whereas SQNR measures only logit-level fidelity and can overlook distributional shifts (Sec. 4.1).

6.2. Ranking-Correlation Algorithm

We detail the Kendall's τ ranking-correlation algorithm used to measure the alignment between our sensitivity metrics and the observed perplexity () change.

Ranking Protocol. We first sort layers by the observed Δ as the ground truth ranking. We then compare this ranking against rankings obtained from each sensitivity metric.

Table 2. Kendall's τ values for each metric across models.

Metric	Mamba-130M	Mamba-380M	Mamba-1.4B	Mamba2-130M	Hymba	Zamba	Avg.
SQNR (dB)	0.6967	0.7314	0.7941	0.4898	0.7124	0.8419	0.7111
$KL_{teacher \to student}$	−0.1980	−0.2316	−0.2607	−0.0272	−0.1617	0.1142	−0.1275
$KL_{student \to teacher}$	**0.8419**	**0.7936**	**0.8327**	**0.8078**	**0.6646**	**0.8060**	**0.7911**
ΔCE	−0.1400	−0.1662	−0.0679	0.2364	−0.1260	−0.1234	−0.0645

Table 3. p-values for each metric across models.

Metric	Mamba-130M	Mamba-380M	Mamba-1.4B	Mamba2-130M	Hymba	Zamba
SQNR (dB)	4.97×10^{-24}	1.5×10^{-51}	1.91×10^{-60}	6.869×10^{-7}	5.87×10^{-57}	4.57×10^{-52}
$KL_{teacher \to student}$	4.06×10^{-3}	1.729×10^{-6}	7.30×10^{-8}	7.827×10^{-1}	3.07×10^{-4}	3.95×10^{-2}
$KL_{student \to teacher}$	2.51×10^{-34}	2.29×10^{-60}	2.84×10^{-66}	2.637×10^{-16}	8.31×10^{-50}	7.1×10^{-48}
ΔCE	4.21×10^{-2}	5.97×10^{-4}	1.61×10^{-1}	1.656×10^{-2}	4.92×10^{-3}	2.61×10^{-2}

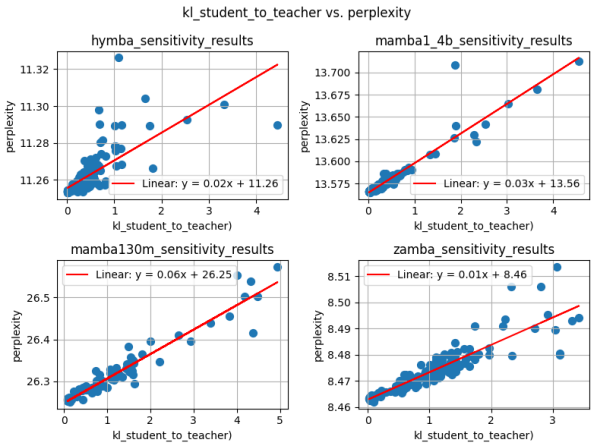

Figure 4. KL-Divergence vs. Perplexity correlation.

In this paper, we demonstrate that quantizing SSM–Transformer models for edge deployment is both necessary and tractable. While prior mixed-precision techniques [5, 14] have shown success in CNNs and Transformers using gradient-based sensitivity analysis or signal-to-quantization-noise ratio (SQNR), these approaches fall short in the context of language modelling. In particular, SQNR—though effective for convolutional architectures—fails to capture the true sensitivity of components in autoregressive LLMs, as we formally demonstrate in Sec. 4.1. Our key contribution is a novel, gradient-free sensitivity analysis method tailored for SSM–Transformer architectures that operates entirely via forward-pass signals. We validate this approach end-to-end with real-world profiling on **Intel Lunar Lake** hardware [8], covering CPU and GPU execution across two Mamba model scales, and show that KL-guided mixed-precision achieves near-FP16/INT8 perplexity while matching or exceeding INT4 throughput—a combination that homogeneous quantization cannot provide.

In this paper, we demonstrate that quantizing SSM–Transformer models for edge deployment is both necessary and tractable. While prior mixed-precision techniques [5, 14] have shown success in CNNs and Transformers using gradient-based sensitivity analysis or signal-to-quantization-noise ratio (SQNR), these approaches are generally less aligned with the linear projection layers within the SSM block than KL divergence.

In particular, SQNR—though effective for convolutional architectures– often lags behind KL-divergence in capturing the true sensitivity of components in autoregressive LLMs, as we formally demonstrate in Sec. 4.1. Our key contribution is a novel, gradient-free sensitivity analysis method tailored for SSM–Transformer architectures that operates entirely via forward-pass signals. We validate this approach end-to-end with real-world profiling on **Intel Lunar Lake** hardware [8], covering CPU and GPU execution across two Mamba model scales, and show that KL-guided mixed-precision achieves near-FP16/INT8 perplexity while matching or exceeding INT4 throughput, a combination that homogeneous quantization cannot provide.

7. On-Device Profiling of Mixed-Precision Quantization

7.1. Experimental Setup

We profile models end-to-end on an Intel **Lunar Lake** platform [8], which integrates a CPU, dedicated GPU, and Neural Processing Unit (NPU) on the same die, making it representative of next-generation AI-capable edge clients. Models are converted to OpenVINO IR via the XAMBA framework [1] and benchmarked with the OpenVINO `benchmark_app` (latency mode, 100 iterations). We evaluate **Mamba-130M** and **Mamba-1.4B** [7] on CPU, and **Mamba-2 130M** [7] on GPU, comparing against **FP16**,

Uniform INT8, and **Uniform INT4** baselines. We report model size (MB), latency (ms), throughput (FPS), and perplexity (WikiText-103).

Mixed-precision configurations. Each configuration p_k ($k = 1, \ldots, 10$) corresponds to a distinct KL threshold κ_k, above which layers are retained at FP16 and below which they are compressed to INT4 (CPU) or INT8 (GPU). Formally, given per-layer KL sensitivity scores $\{s_\ell\}_{\ell=1}^{L}$, configuration $\mathbf{p}k$ assigns:

$$b_\ell = \begin{cases} \text{FP16} & \text{if } s_\ell \geq \kappa_k, \\ \text{INT4/INT8} & \text{otherwise,} \end{cases} \tag{10}$$

where $\kappa_1 > \kappa_2 > \cdots > \kappa_{10}$. **p01** applies the most conservative threshold, quantizing only the least sensitive layers; **p10** applies the most aggressive threshold, approaching near-uniform quantization. The configurations thus trace a smooth compression–accuracy curve controlled entirely by the KL sensitivity scores.

7.2. CPU Profiling: Mamba-130M and Mamba-1.4B

Figures 5 to 7 show perplexity, latency, and throughput plotted against model size for both CPU models. KL-guided quantization progressively compresses Mamba-130M from 493 MB to 84 MB (5.9×) and Mamba-1.4B from 5.2 GB to 723 MB (7.2×), matching Uniform INT4 at the most aggressive setting. Perplexity remains near the FP16 baseline through **p06** for Mamba-130M and **p09** for Mamba-1.4B; beyond these thresholds, a sharp sensitivity cliff emerges—consistent with the ablation finding that a single high-sensitivity layer can dominate the entire quantization error budget (Sec. 5.1). **p05** represents the best operating point for both models, achieving INT8-level or better throughput (35.9 vs. 35.5 FPS; 5.6 vs. 5.1 FPS) with near-FP16 perplexity, and delivering the lowest observed latency for Mamba-1.4B (178 ms)—below both Uniform INT8 (196 ms) and INT4 (190 ms).

7.3. GPU Profiling: Mamba-2 130M

Figure 8 reports GPU results as two complementary views: perplexity vs. throughput (left) and perplexity vs. size.

Why GPU memory savings are negligible. Unlike the CPU models, KL-MP configurations cluster around ≈45 MB regardless of k, versus 81 MB for the FP16 baseline. This is a consequence of two pipeline constraints: (1) the XAMBA exporter applies `compress_to_fp16=True` by default, so the FP16 baseline is already half the FP32 size before any further quantization; and (2) the GPU pipeline applies INT8 rather than INT4, yielding only a 2× reduction over FP16 on the quantized projection layers, while the non-quantizable embedding table dominates the file size throughout. Consequently, efficiency gains on GPU manifest as **latency and throughput improvements** from faster INT8 arithmetic—not memory reduction.

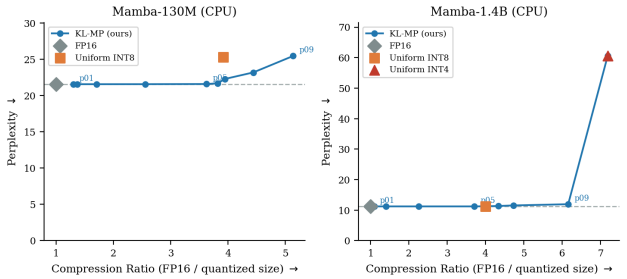

Figure 5. Perplexity vs. model size (CPU, Intel Lunar Lake). The x-axis is inverted so more compressed models appear to the right. KL-MP maintains near-FP16 accuracy across a wide compression range. The sharp cliff at **p10** for Mamba-130M (diverged) and **p10** for Mamba-1.4B (PPL = 60.6) reflects the quantization of a single high-sensitivity layer identified in our ablation (Sec. 5.1); Uniform INT4 crosses this boundary by construction.

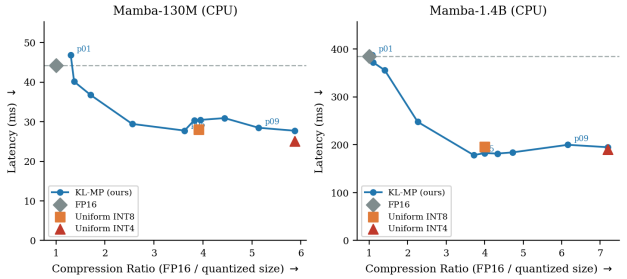

Figure 6. Inference latency vs. model size (CPU, Intel Lunar Lake). KL-MP **p05** achieves the lowest latency for Mamba-1.4B (178 ms), undercutting both Uniform INT8 and INT4, demonstrating that sensitivity-guided precision allocation reduces not only memory but also compute overhead.

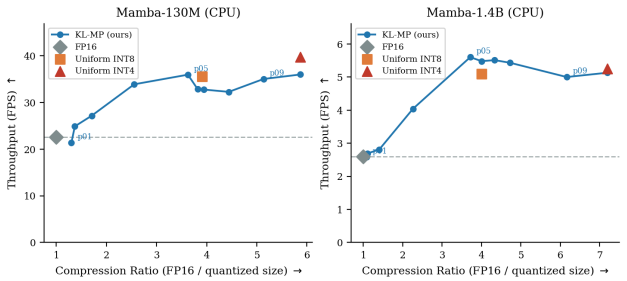

Figure 7. Throughput vs. model size (CPU, Intel Lunar Lake). KL-MP **p05** matches or exceeds INT4 throughput for both models while preserving near-FP16 perplexity, occupying a region of the accuracy–efficiency space inaccessible to homogeneous quantization.

Table 4. On-device results: best KL-MP point vs. baselines on Intel Lunar Lake. ↑ higher is better; ↓ lower is better. "–" * The 130M model exhibited substantially higher perplexity under uniform INT4 quantization. †FPS and latency are not reported, as INT8 execution is not supported for the certain relevant operations on Intel Lunar Lake

Model	Config	PPL ↓	Size	FPS ↑	Latency ↓
Mamba-130M (CPU)	FP16	21.59	493 MB	22.6	44 ms
	INT8	25.32	126 MB	35.5	28 ms
	INT4*	3e35	84 MB	39.7	25 ms
	KL-MP p05 (ours)	**21.61**	**136 MB**	**35.9**	**28 ms**
Mamba-1.4B (CPU)	FP16	11.22	5.2 GB	2.6	384 ms
	INT8	11.25	1.3 GB	5.1	196 ms
	INT4	60.55	723 MB	5.3	190 ms
	KL-MP p05	**11.22**	**1.4 GB**	**5.6**	**178 ms**
Mamba-2 130M (GPU)	FP16	46.45	81 MB	0.02	60006 ms
	INT8†	53.03	125 MB	–	–
	KL-MP p02	**46.46**	**45 MB**	**0.29**	**3417 ms**

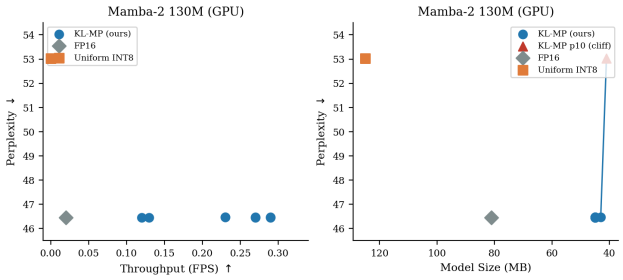

Figure 8. GPU profiling of Mamba-2 130M on Intel Lunar Lake. **(Left)** Perplexity vs. throughput: KL-MP configurations **p01**–**p09** achieve up to 14.5× throughput improvement over FP16 with negligible perplexity cost. **(Right)** Perplexity vs. model size: file sizes cluster near 45 MB due to FP16 export compression and INT8-only quantization; only **p10** (red triangle) crosses the precision cliff, collapsing to Uniform INT8 perplexity.

Latency and accuracy. The FP16 baseline incurs 60,006 ms due to iGPU memory-transfer overhead, while KL-MP reduces this by up to **17.6**× (**p02**: 3,417 ms, 0.29 FPS). Perplexity holds at 46.45–46.48 for **p01**–**p09**, matching the FP16 baseline; only **p10** crosses the precision cliff to 53.03—identical to Uniform INT8—confirming that the KL top-k selection correctly identifies the critical layer boundary.

8. Conclusion

In this work, we introduced a practical and scalable sensitivity analysis framework tailored specifically for hybrid SSM–Transformer models, enabling efficient mixed-precision quantization without requiring backpropagation or retraining. By systematically evaluating quantization sensitivity through forward-pass metrics, we identified critical layers where quantization-induced errors significantly impact model performance. Our results on language modeling tasks demonstrate that selectively maintaining higher precision in these sensitive layers effectively mitigates accuracy degradation, achieving up to 4× reduction in model size with minimal perplexity loss. Future directions include extending our analysis framework to broader hybrid architectures and exploring dynamic quantization strategies that adapt precision allocation during inference, further enhancing model efficiency on resource-constrained edge devices. Real-world profiling on Intel Lunar Lake confirms these findings end-to-end: KL-guided mixed-precision reduces Mamba-1.4B from 5.2 GB to 1.4 GB with no measurable perplexity loss, and cuts Mamba-2 130M GPU latency by up to 17.6×17.6 times 17.6× over the FP16 baseline.

References

[1] XAMBA: Enabling efficient state space models on resource-constrained neural processing units. Software repository, 2024. 6

[2] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022. 1

[3] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. *arXiv preprint arXiv:2305.14314*, 2023. 2

[4] Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, et al. Hymba: A hybrid-head architecture for small language models. *arXiv preprint arXiv:2411.13676*, 2024. 1, 2, 5

[5] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 293–302, 2019. 1, 2, 6

[6] Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. Zamba: A compact 7b ssm hybrid model. *arXiv preprint arXiv:2405.16712*, 2024. 1, 2

[7] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 1, 2, 6

[8] Intel Corporation. Intel core ultra 200v series processors (lunar lake) product brief. https://www.intel.com/content/www/us/en/products/docs/processors/core-ultra/core-ultra-200v-product-brief.html, 2024. Accessed: 2025. 6

[9] Maurice G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1–2):81–93, 1938. 3

[10] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018. 1, 2

[11] Solomon Kullback and Richard A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22 (1):79–86, 1951. 3

[12] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning (ICML)*, 2020. 2

[13] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021. 1, 2

[14] Nilesh Prasad Pandey, Markus Nagel, Mart van Baalen, Yin Huang, Chirag Patel, and Tijmen Blankevoort. A practical mixed precision algorithm for post-training quantization. *arXiv preprint arXiv:2302.05397*, 2023. 1, 2, 3, 5, 6

[15] Alessandro Pierro and Steven Abreu. Mamba-ptq: Outlier channels in recurrent large language models. *arXiv preprint arXiv:2407.12397*, 2024. 1, 2

[16] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*, 2016. 2

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 1

[18] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018. 2

[19] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: Hardware-aware automated quantization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[20] Guangxuan Xiao, Ji Lin, Mathieu Seznec, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. *arXiv preprint arXiv:2211.10438*, 2023. 1

[21] Yuewei Yang, Xiaoliang Dai, Jialiang Wang, Peizhao Zhang, and Hongbo Zhang. Efficient quantization strategies for latent diffusion models. *arXiv preprint arXiv:2312.05431*, 2023. 2, 3

[22] Zhewei Yao, Yuxiong He, et al. ZeroQuant: Efficient and affordable post-training quantization for large-scale transformers. *arXiv preprint arXiv:2206.01861*, 2022. 1

A. On-Device Profiling: Reproducibility Guide

All profiling scripts target Intel Lunar Lake via OpenVINO. The pipeline runs in three stages.

Step 1 | Convert and Quantize

```
python convert.py              # base models
python quantize_uniform.py     # INT8 / INT4
python quantize_mixed.py       # KL M-P (CPU)
python quantize_mixed_gpu.py   # KL M-P (GPU)
```

The sensitivity metric is controlled by a constant inside each script. All output filenames automatically inherit the selected tag (e.g., `mamba-130m-hf_kl_point05.xml`):

```
SENSITIVITY_METRIC = "kl_student_to_teacher"
METRIC_TAG = "kl"
```

The ten configurations **p01–p10** are produced by splitting the sensitivity-ranked layer list into equal segments. Configuration **p01** quantizes only the least sensitive layers, while **p10** approaches near-uniform quantization.

Step 2 | Benchmark

```
python benchmark.py      # CPU latency +
    throughput
python benchmark_gpu.py  # GPU latency +
    throughput
```

Each model is evaluated using the OpenVINO `benchmark_app` tool:

```
benchmark_app -m <model>.xml \
  -d CPU -hint latency \
  -t 60 -niter 50 \
  --inference_only TRUE
```

The flags `-t 60` and `-niter 50` ensure that each run executes for at least 60 seconds and at least 50 iterations, whichever takes longer.

Results are written to `log/benchmark_log/` as per-model text logs and a summary CSV file: `latency_throughput_{device}_{tag}_report.csv`.

```
python eval_perplexity.py      # CPU, WikiText-2
    PPL
python eval_perplexity_gpu.py  # GPU, WikiText-2
    PPL
```

Perplexity is computed on the WikiText-2 test set. Fake quantization is applied to replicate each configuration's weight precision during evaluation.

```
ds = load_dataset(
    "wikitext", "wikitext-2-raw-v1",
    split="test")

text = "\n\n".join(
    t for t in ds["text"] if t.strip())
```

Results are stored in `perplexity_results_{tag}.json` with the structure

```
{model: {point: perplexity}}
```

This format allows direct comparison across sensitivity metrics and model scales.