EASY-KUANT: DIAGNOSING AND MITIGATING KV-CACHE QUANTIZATION COLLAPSE WITH SELECTIVE LAYER PROTECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Serving large language models increasingly relies on quantizing the key–value (KV) cache to hardware-native low-bit formats such as NVFP4 and FP8, widely reported to be near-lossless at four bits. On the Qwen2.5 family and its derivatives the same one-line switch is catastrophic: in vLLM, Qwen2.5-7B-Instruct falls from 90.2% to 11.0% on GSM8K under NVFP4 and to 2.2% under FP8, and vision-language checkpoints fall to 0%. Rather than tuning against cache reconstruction error, as prior methods do, we ask *where* the error becomes a failure. Replaying each layer's attention from clean activations shows that cache error is flat across depth while attention-output error varies by an order of magnitude, enters almost entirely through keys, and concentrates in one to four layers; two-directional interventions show that these layers are sufficient both to cause and to repair the collapse. The diagnosis becomes Easy-Kuant: a few forward passes yield one statistic per layer, a fixed threshold turns it into the list of layers to keep in BF16, and stock vLLM applies the list through an existing argument—no kernel, no search, no gradients. On the native serving path Easy-Kuant returns NVFP4 to 90–96% of BF16 accuracy on the 3B and 7B checkpoints and FP8 to 98–100%, transfers unchanged to long-context and multimodal checkpoints, matches software baselines that cannot run in vLLM, and costs $0.94\times$ the per-token KV processing of full NVFP4 at 85% of its cache capacity. An optional quantization-aware stage, trained on the model's own rollouts and guided by the same layer list, adds up to 3.7 further points where a gap remains.

1 INTRODUCTION

The key–value (KV) cache of a Transformer decoder (Vaswani et al., 2017) is the dominant dynamic memory cost of serving: it sets how many requests an engine can hold (Kwon et al., 2023), and decoding is bound by the bytes it moves per step (Dao et al., 2022). Accelerators therefore expose narrow formats for the cache itself—FP8, and NVFP4, which pairs E2M1 data with an E4M3 scale per 16 elements (NVIDIA, 2026)—and engines such as vLLM enable them with a single argument. A substantial literature reports that the KV cache can be taken to roughly four bits at near-lossless quality (Hooper et al., 2024; Liu et al., 2024c; Kang et al., 2024; Li et al., 2025). That literature also agrees on where the residual damage comes from, a few activation channels far above the rest (Sun et al., 2024), and repairs it with finer granularity (Liu et al., 2024c), non-uniform codes and outlier routing (Hooper et al., 2024), low-rank correction (Kang et al., 2024), layer-wise precision search (Li et al., 2025), or rotation and scale migration (Ashkboos et al., 2024; Liu et al., 2025; Xiao et al., 2023). All of these are tuned against the reconstruction error of the cached tensor, and all replace the engine's native cache path with a method-specific layout, kernel or search.

On one widely deployed family of checkpoints, none of this is what happens. With weights, prompts and decoding untouched, switching the cache of Qwen2.5-7B-Instruct (Qwen Team, 2025a) to NVFP4 in vLLM drops GSM8K accuracy from 90.2% to 11.0%, and to 2.2% under FP8; Qwen2.5-1.5B-Instruct drops from 73.7% to 0.2%. Models trained from these checkpoints inherit the failure—Qwen2.5-VL-3B (Bai et al., 2025) falls to 0.0% on MMBench and Qwen2.5-Omni-7B (Xu et al., 2025) to 0.2% on GSM8K

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

(Table 1)—while Qwen3-VL-8B (Qwen Team, 2025b) loses nothing. The quantity the field optimizes does not see this coming: replaying every layer from clean activations, the reconstruction error of the quantized cache is nearly constant across depth (0.007–0.011), while the error the same perturbation induces at the attention output ranges from 0.031 to 0.679 and enters almost entirely through keys. Measured at the output, the failure is localized to one to four layers per model, and interventions in both directions show the localization is causal: quantizing *only* those layers drives 7B to 0.0%, protecting *only* them returns 88.3% against 89.8% in BF16, and protecting an equal number of random layers, or the values instead of the keys of the selected layers, returns 0.0%.

Table 1: **The collapse is family-wide.** Accuracy (%) with the KV cache in BF16 and in NVFP4, nothing else changed.

Model	Task	BF16	NVFP4
Qwen2.5-1.5B-Inst	GSM8K	73.7	0.2
Qwen2.5-3B-Inst	GSM8K	85.9	1.1
Qwen2.5-7B-Inst	GSM8K	90.2	11.0
Qwen2.5-VL-3B	MMBench	83.0	0.0
Qwen2.5-VL-7B	MathVista	31.7	2.9
Qwen2.5-Omni-3B	GSM8K	81.9	1.4
Qwen2.5-Omni-7B	MMBench	87.0	26.7

The diagnosis becomes a method almost mechanically. Easy-Kuant runs a few calibration prompts through the BF16 model, records the largest post-RoPE key magnitude per layer, protects every layer above a fixed multiple of the per-model median, and quantizes the rest. The output is a list of integers that stock vLLM already accepts through `kv_cache_dtype_skip_layers`, so deployment adds no quantizer, kernel or cache layout; an optional stage fine-tunes the checkpoint on its own rollouts with the deployment quantizer in the loop where a gap remains. On the native serving path the repair returns NVFP4 to 90–96% of BF16 GSM8K accuracy on the 3B and 7B checkpoints and FP8 to 98–100% on all three, with the same picture on MATH-500, MMLU-STEM, long-context retrieval and vision and omni-modal checkpoints; against KIVI, KVQuant, GEAR and KVTuner it is within noise on 7B and trails the finer-grained schemes only on the 2-KV-head 1.5B model in long generation. Preparation costs 4 300× less than layer-wise precision search and per-token KV processing 0.94× that of full NVFP4.

In summary, our key contributions are as follows:

- **Causal localization of KV-cache quantization collapse:** The failure of NVFP4 and FP8 caches on the Qwen2.5 family is produced by the keys of one to four layers whose channel-structural outliers dominate the shared block scale; measured at the attention output rather than in the cache, these layers are sufficient both to cause and to repair the collapse.
- **Easy-Kuant:** A training-free, search-free repair that follows directly from the diagnosis—a few forward passes, one threshold, and a list of layers that stock vLLM already accepts—with no kernel, 4 300× less preparation compute than precision search, and per-token KV processing cheaper than full NVFP4. An optional on-policy quantization-aware stage, guided by the same layer list, extends the repair where the training-free gap is largest.
- **Native-path evidence and mechanism ablations:** In vLLM, Easy-Kuant returns NVFP4 to 90–96% and FP8 to 98–100% of BF16 accuracy across text, math, long-context and multimodal checkpoints at 85% of full-NVFP4 cache capacity; ablations show that the accuracy–memory trade-off is a step rather than a slope, and that range-redistributing alternatives fail because the outlier channels lack mantissa precision, not range.

2 Diagnosing the Collapse

Table 1 shows a failure that the standard account of low-bit KV damage does not predict: switching the cache format costs the Qwen2.5 family not a few points but essentially everything, and it does so for every descendant of these checkpoints—fine-tuned for instructions, mathematics, images or audio—while Qwen3-VL-8B is untouched. The failure is binary rather than graded, which points to a specific breaking point rather than to accumulated rounding noise, and it is inherited through further training, which points to a representation the checkpoints share. The question is therefore not how large the quantization error is, but *where* in the network it is turned into a failure, and whether that location is causal.

Throughout, FP8 stores E4M3 values with a per-tensor scale and NVFP4 stores E2M1 values with an E4M3 scale per block of 16 elements (NVIDIA, 2026). Deployment accuracy is measured in native vLLM (Kwon et al., 2023); layer-level measurements use a simulation harness that fake-quantizes

the post-RoPE keys and values inside attention (two variants, agreeing on every conclusion; Appendix A.1). Errors are normalized mean squared errors (NMSE) against the BF16 tensor.

2.1 The Damage Is Produced by a Few Layers, Through Their Keys

The obvious place to look is the cache itself, and it is uninformative: the reconstruction error of the quantized keys and values is 0.007–0.011 in every layer of every model, which is what every KV quantizer is tuned against. We therefore measure the damage where it acts: replaying one layer's attention from its clean BF16 inputs with only that layer's cache quantized, and discarding the result so that nothing propagates, isolates each layer's own contribution to the error of the attention output (Table 2). Seen this way the failure has a shape. Output error varies by an order of magnitude across the layers of one model and is concentrated in one to four of them—the worst layer of 7B, layer 27, is six times worse than the layer mean—and it enters through the keys: quantizing K alone reproduces the joint damage (0.675 against 0.679 on 1.5B, 0.444 against 0.444 on 7B), while quantizing V alone is two orders of magnitude smaller. The first-order expansion explains the asymmetry: for one head with attention weights a and $J_a = \mathrm{diag}(a) - aa^\top$,

Table 2: **Where the damage is produced.** Attention-output NMSE when a single layer's keys, values or both are quantized to NVFP4; the cache NMSE itself is 0.007–0.011 at every layer.

	Layer	Worst layer			
Model	mean	idx	K/V	K only	V only
Qwen2.5-1.5B-Inst	.077	0	.679	.675	.007
Qwen2.5-7B-Inst	.069	27	.444	.444	.001

$$\delta o = V^\top J_a \, E^K q/\sqrt{d} \; + \; (E^V)^\top a \; + \; O(\|E\|^2), \tag{1}$$

so a key error E^K passes through the query direction, the softmax Jacobian and the value contrasts and can be enlarged or cancelled depending on its direction, whereas a value error E^V is a convex combination of per-position errors. Cache NMSE sees none of these factors, which is why it cannot rank layers; the expansion says how the two tensors enter, not that keys must dominate, so the ranking above is measured rather than derived (Appendix C).

2.2 What Makes Those Layers Different

The damaged layers share one visible feature. Their post-RoPE keys contain a handful of channels far above all others—the top eight channels of a head carry essentially all of the outlier mass—and these channels are fixed: the outliers appear at every token rather than at a few positions, and attention-sink positions are *smaller* than average. Under a block-shared scale this is the worst possible structure: the block maximum sets the quantization step for its fifteen neighbours, and any neighbour below $1/24$ of the maximum is rounded to zero (Proposition C.1), so a channel that dwarfs its neighbours erases or crushes them at every token. It also gives the damaged layers a signature that can be read from a single forward pass: the largest absolute post-RoPE key of each layer, whose depth profile is flat with a few spikes at exactly the layers Table 2 identifies (Figure 3, §4.5).

2.3 The Layers Are Causal

A statistic that co-varies with the damage does not make a layer responsible for an end-to-end failure, so we intervene in both directions on the layers the signature picks out (layer 0 on 3B; layers 0, 3 and 27 on 7B). If *only* these layers are quantized, with every other layer left in BF16, 7B collapses to 0.0%, exactly as under all-layer quantization; if *only* these layers are protected, with everything else quantized, it recovers 88.3% against a BF16 reference of 89.8% (Figure 1). Protecting an equal number of random layers recovers nothing, protecting the *values* of the selected layers recovers nothing, and protecting their keys alone recovers 82.8%. Trading memory for accuracy one layer at a time sharpens this into a step (Figure 2): on 7B, protecting layer 27 alone leaves 0.5%, adding layer 0 returns 98% of BF16, and the remaining 26 layers together add one point. Two layers' keys thus produce the collapse and suffice to repair it. On the 2-KV-head 1.5B model the same curve rises gradually instead, which is where a layer-level view meets its granularity limit.

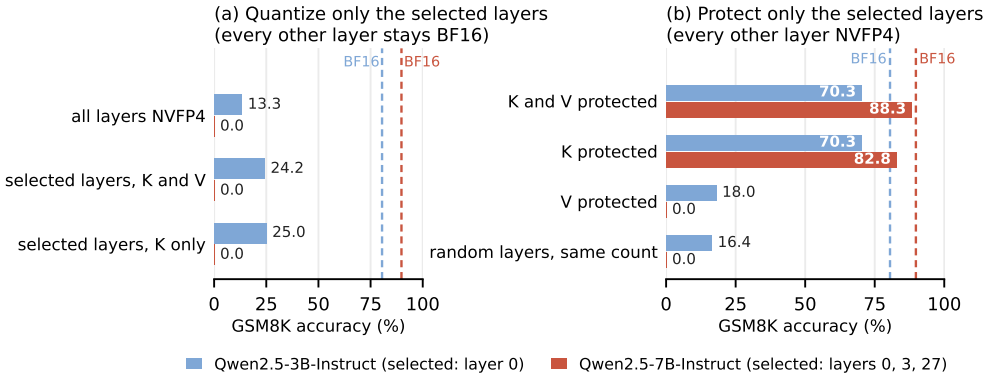


Figure 1: **The probe-selected layers are sufficient to cause and to repair the collapse.** *Left*: quantizing only those layers, with every other layer in BF16, already reproduces most of the all-layer failure. *Right*: protecting only those layers, with everything else in NVFP4, recovers most of the BF16 reference (dashed lines), whereas protecting their values or an equal number of random layers recovers nothing.

2.4 Why a Local Error Becomes a Total Failure

A local output error at one layer does not by itself explain a fall to 0%. It becomes one because the cache is written by the model that reads it. Holding the token sequence fixed, the keys a quantized model *newly computes* at a late position have drifted from their BF16 counterparts by an NMSE of 0.45–0.62, fifty times the error of writing a key to the cache, and the output distribution has moved by 6–7 nats of KL; protecting the selected layers brings both back to the noise floor (Appendix B.3). The corrupted output re-enters the residual stream, corrupts the next keys, and compounds. Consistently, on MMLU-STEM, which has no decode loop, bare NVFP4 falls only to chance level and protection returns every model to within a few points of BF16, including the 1.5B model whose GSM8K gap is 16 points (Table 3). The layers can be found from one forward pass and neutralized by keeping them out of the format, which is what the next section does.

3 Easy-Kuant

A few layers produce the failure, they can be recognized from the magnitude of their keys, and keeping them out of the low-bit format undoes it. Easy-Kuant does exactly this with a forward-only probe, one threshold and a configuration argument the serving engine already exposes (Algorithm 1); nothing is searched, trained or fitted.

3.1 Finding the Layers: One Statistic and One Threshold

What a block-shared scale cannot tolerate is a channel far above its neighbours, and such channels are fixed and present at every token (§2.2), so a handful of calibration prompts $\mathcal{D}_{\text{cal}}$ is enough to expose them. We run the BF16 model once and record, for every layer, the largest absolute post-RoPE key over all tokens, heads and channels,

$$a_l = \max_{x \in \mathcal{D}_{\text{cal}}} \max_{t,h,d} \left| K_l(x)_{t,h,d} \right|. \tag{2}$$

This is a cheap proxy for how coarse a layer's worst quantization step will be, not a sensitivity measure, and deliberately a *key* statistic (Table 2); its usefulness rests on §2.3 and §4. Because absolute key magnitudes differ across models, the decision is relative: a layer is protected when its statistic stands out from the model's own typical layer,

$$\mathcal{O} = \{\, l : a_l > \tau \cdot \text{median}_{l'}\, a_{l'} \,\}, \tag{3}$$

with one threshold τ shared by every model; the empty set is a legitimate outcome. Because the accuracy–memory curve of a collapsing model is a step followed by a plateau (§2.3), τ is a decision

Algorithm 1 Easy-Kuant: probe and precision policy

Require: BF16 model with layers $l = 0, \ldots, L-1$; calibration prompts $\mathcal{D}_{\text{cal}}$; threshold τ; low-bit cache format Q

1: $a_l \leftarrow 0$ for all layers
2: **for** each prompt $x \in \mathcal{D}_{\text{cal}}$ **do**
3: Run one BF16 forward pass and collect the post-RoPE keys $K_l(x)$ of every layer
4: $a_l \leftarrow \max\left(a_l, \max_{t,h,d} |K_l(x)_{t,h,d}|\right)$
5: **end for**
6: $\mathcal{O} \leftarrow \{l : a_l > \tau \cdot \text{median}_l\, a_l\}$
7: **return** $\mathcal{O}$; serve with the K/V of layers in $\mathcal{O}$ cached in BF16 and all other layers cached in Q

boundary rather than an optimum, and any threshold that clears the step gives the same result; the configuration used here and the layers it selects are reported in §4.

3.2 Applying the Policy: A Configuration Change

Because the policy is a list of layer indices, deployment needs no quantizer, kernel or cache layout of its own; stock vLLM (Kwon et al., 2023) already accepts the list:

```
LLM(model, kv_cache_dtype="nvfp4", kv_cache_dtype_skip_layers=[ℓ1, ℓ2, ...])
```

The selected layers keep both K and V in BF16, because that is what the engine's control expresses, although only the keys need it. With $p = |\mathcal{O}|/L$ and b_q bits per element for the low-bit format (4.5 for NVFP4 with its block scales), the cache costs roughly $(1-p)\, b_q + 16p$ bits per element, 5.7 when three of 28 layers are protected; exempting a layer removes quantization work rather than adding any (§4.4).

3.3 Why Store the Outliers Rather Than Remove Them

The natural alternative to paying sixteen bits for a few layers is to shrink the offending channels, and every zero-cost variant we tried fails for one reason. Migrating scale from keys into queries (Xiao et al., 2023; Lin et al., 2025) is harmful under a block-shared scale: the shrunken channel hands its quantization step to its block neighbours and the query side re-amplifies the error. Protecting attention-sink tokens does not help, because the sinks are not where the mass is. Reordering channels so that the outliers share as few blocks as possible slightly *worsens* accuracy, because a 128-dimensional head has only eight blocks and the outlier channels already share one. Keeping the selected layers in FP8 collapses, as all-layer FP8 collapses 7B (Table 3) despite its far wider exponent range, because E4M3's three mantissa bits leave a 6.25% relative rounding error whatever the scale. What the outlier channels lacked in every variant was mantissa precision rather than dynamic range (Appendix B.7), which is why the method spends bits rather than rearranging them.

3.4 Optional On-Policy Quantization-Aware Tuning

When training is affordable, an optional stage lets the model adapt to the precision it will be served at, without external data: the BF16 checkpoint generates its own continuations for a small set of prompts (on-policy self-rollouts), and the model is fine-tuned on them with a response-only cross-entropy while a straight-through fake quantizer (Bengio et al., 2013; Liu et al., 2024b) applies the deployment format to the post-RoPE K/V tensors. The probe-selected layers may be exempt from the fake quantizer or included, and deployment always protects the probe-selected set. The objective and the checkpoints used in this paper are specified in Appendix B.5 (Eq. (5), Table 12); §4.5 shows that the stage pays off where the training-free repair leaves a gap.

4 Experiments

Setup. Text models are Qwen2.5-{1.5B,3B,7B}-Instruct (Qwen Team, 2025a); multimodal models are Qwen2.5-VL-{3B,7B} and Qwen2.5-Omni-{3B,7B} (Bai et al., 2025; Xu et al., 2025), and

Table 3: **Native vLLM deployment** of Qwen2.5-Instruct models (accuracy %). +Easy-Kuant protects the probe-selected layers ("Skip"); +QAT additionally uses the optional QAT checkpoint.

		GSM8K						MATH-500				MMLU-STEM			
Model	Skip	BF16	NVFP4	+Easy-Kuant	+QAT	FP8	+Easy-Kuant	BF16	NVFP4	+Easy-Kuant	+QAT	BF16	NVFP4	+Easy-Kuant	+QAT
1.5B-Inst	0,1	73.7	0.2	57.8	**61.5**	15.1	72.6	53.8	0.0	**39.4**	35.2	54.7	27.1	**52.4**	50.4
3B-Inst	0	85.9	1.1	77.7	**78.8**	81.9	85.8	61.8	0.0	**52.6**	51.4	62.2	27.7	58.6	**59.6**
7B-Inst	0,3,27	90.2	11.0	86.4	**86.5**	2.2	88.3	65.4	3.0	**60.0**	59.2	70.7	30.2	64.6	**65.3**

Table 4: **Multimodal transfer** (accuracy %) with the probe and threshold unchanged; *ours* is NVFP4+Easy-Kuant. Qwen3-VL-8B does not collapse.

	GSM8K			MMBench			MathVista			RealWorldQA		
Model	BF16	NVFP4	ours	BF16	NVFP4	ours	BF16	NVFP4	ours	BF16	NVFP4	ours
VL-3B	81.7	0.0	74.2	83.0	0.0	83.2	46.1	0.1	44.0	63.7	0.3	64.7
VL-7B	49.6	0.2	50.1	86.7	19.2	86.3	31.7	2.9	32.7	69.3	5.0	69.3
Omni-3B	81.9	1.4	75.4	83.3	72.0	81.4	58.7	29.3	55.2	65.0	54.8	62.5
Omni-7B	87.0	0.2	86.2	87.0	26.7	86.5	55.6	8.0	53.9	68.9	28.0	69.4
Qwen3-VL-8B	91.9	90.8	91.1	89.4	88.3	88.5	54.4	53.9	–	73.1	70.3	71.4

Qwen3-VL-8B (Qwen Team, 2025b), which does not collapse. Benchmarks are GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021b; Lightman et al., 2024), MMLU-STEM scored by option log-likelihood (Hendrycks et al., 2021a), passkey retrieval at 4k–16k (Kamradt, 2023) with LongBench-E (Bai et al., 2024), and MMBench, MathVista-mini and RealWorldQA (Liu et al., 2024a; Lu et al., 2024; xAI, 2024). KIVI-4, KVQuant-4, GEAR-4 and KVTuner (Liu et al., 2024c; Hooper et al., 2024; Kang et al., 2024; Li et al., 2025) are compared in a shared quantization-simulation harness, since they are not available as vLLM cache backends. The probe uses the same eight calibration prompts (694 tokens) and $\tau{=}4$ everywhere, takes 1.8 s on 7B, and selects $\{0\}$ for 3B, $\{0, 1\}$ for 1.5B and $\{0, 3, 27\}$ for 7B; calibrating 7B on contexts an order of magnitude longer changes one peripheral layer ($\{0, 1, 27\}$) and never the two load-bearing ones (Appendix A).

4.1 Accuracy on the Native Serving Path

In native vLLM (Table 3), bare NVFP4 leaves every model at 0–11% on GSM8K and 0–3% on MATH-500; protecting the probe-selected layers returns 7B to 86.4% (96% of BF16) and 3B to 90% of BF16, with the same ordering on MATH-500 (92% and 85%) and MMLU-STEM (91–96% on all three models). FP8 collapses too where the outliers are most extreme (7B falls to 2.2%) and the same list repairs it to 98–100% of BF16 on every model; on 3B, where FP8 does not collapse, protection is harmless. The 2-KV-head 1.5B model retains the largest gap (57.8% against 73.7% on GSM8K) and is where the optional QAT stage matters, raising GSM8K to 61.5%; on 7B QAT changes nothing.

4.2 Multimodal and Long-Context Transfer

Nothing in the diagnosis is specific to text, and the cleanest recoveries are on the vision and omni-modal checkpoints. With the identical probe and threshold, Qwen2.5-VL and Qwen2.5-Omni (Table 4) collapse if anything more completely (VL-3B scores at most 0.1% on GSM8K, MMBench and MathVista alike) and protection returns 13 of 16 cells to within 2.5 points of BF16, the exceptions being the two 3B models on GSM8K and Omni-3B on MathVista; Qwen3-VL-8B does not collapse and is left unchanged. On long contexts, where a KV format is meant to pay off, bare NVFP4 fails most completely (Table 5): passkey retrieval falls from 1.00 to 0.00 on 1.5B and 7B and HotpotQA F1 below 0.04 on every model, while protection returns retrieval to 1.00, 0.99 and 0.88 on 7B, 3B and 1.5B and HotpotQA to within noise of BF16.

Table 5: **Long context**: passkey retrieval (NIAH, 4k–16k) and LongBench-E HotpotQA F1.

Model	Task	BF16	NVFP4	+Easy-Kuant
Qwen2.5-1.5B-Inst	NIAH	1.000	0.000	0.875
	HotpotQA	0.181	0.007	0.239
Qwen2.5-3B-Inst	NIAH	1.000	0.375	0.986
	HotpotQA	0.313	0.036	0.326
Qwen2.5-7B-Inst	NIAH	1.000	0.000	**1.000**
	HotpotQA	0.543	0.003	0.512

Table 6: **Serving** (Qwen2.5-7B-Instruct, vLLM): cache capacity and throughput relative to BF16 when the cache always fits (A) and under a tight KV pool (B).

KV cache	Capacity	A	B
BF16	1.00	1.00	1.00
FP8	2.00	0.94	1.09
FP8+Easy-Kuant	1.87	0.95	1.10
NVFP4	3.24	0.91	1.22
NVFP4+Easy-Kuant	2.76	0.91	1.12

Table 7: **Comparison with KV-quantization baselines** (accuracy %). Because the baselines are available as research implementations rather than vLLM cache backends, all methods are compared in a shared quantization-simulation harness.

Model	Task	BF16	NVFP4	KIVI-4	KVQ-4	GEAR-4	KVTuner	Easy-Kuant	+QAT
1.5B-Inst	GSM8K	67.3	0.0	65.1	65.2	**66.1**	59.3	45.2	61.2
	MATH-500	29.4	0.0	29.8	30.0	28.6	25.8	16.6	**30.4**
	MMLU-STEM	53.4	24.1	**54.0**	53.4	53.5	52.5	51.1	51.7
3B-Inst	GSM8K	82.5	9.1	81.4	80.4	**82.3**	80.8	76.0	78.1
	MATH-500	58.4	2.2	**58.4**	57.0	55.6	56.8	50.0	51.4
	MMLU-STEM	57.4	32.5	57.9	56.6	56.1	57.1	**58.1**	56.5
7B-Inst	GSM8K	87.2	0.5	86.6	85.6	86.8	84.6	86.4	**88.3**
	MATH-500	62.0	0.0	62.0	60.0	60.4	62.4	60.2	**63.6**
	MMLU-STEM	70.3	27.6	70.3	69.8	**70.9**	69.2	69.2	68.1

4.3 Comparison with Software Baselines

Table 7 places the policy against four software baselines in one harness. On 7B Easy-Kuant is within 0.4 points of KIVI and GEAR and above KVTuner on GSM8K, within about 2 points of the best baseline on MATH-500 and within 1–2 points on MMLU-STEM. The one systematic shortfall is the 2-KV-head 1.5B model in long generation, where whole-layer protection trails the finer-grained schemes by 20 points on GSM8K and 13 on MATH-500 while matching them on MMLU-STEM: with two KV heads the damage is spread over more layers (Figure 2) and a layer is a coarse unit. We claim no uniform quality advantage; the case for the method is that this quality is reached on the engine's native path with a list of integers, whereas each baseline is a research implementation whose layout, kernel or search vLLM does not execute.

4.4 Preparation and Serving Cost

Table 8 reruns each baseline's offline stage under its default configuration and microbenchmarks per-token KV processing in one harness. The probe's peak memory is that of inference, whereas Fisher-based calibration backpropagates and needs three times as much; the probe takes 1.8 s against 7 783 s for layer-wise precision search (4 300×); and protection is the only entry that makes serving *cheaper* than the unrepaired baseline (0.920 against 0.974 µs per token), while sparse-outlier maintenance costs 2.87× and per-step low-rank correction 14.98×. Under memory pressure (Table 6), protected NVFP4 holds 2.76× as many tokens as BF16 against 3.24× for full NVFP4 and delivers 1.12× against 1.22× throughput; when the cache always fits both run at 0.91× BF16, and a load sweep reaches 1.70× (Table 13).

4.5 Ablations

Where the repair sits between NVFP4 and BF16. Figure 2 asks how much of the way from full NVFP4 (4.5 bits per element) towards BF16 keys (10.25 bits) one has to go, protecting layers in the order the probe ranks them. On 7B, almost none: layer 27 alone buys nothing (0.5%), adding layer 0 (+0.2 bit) buys 98% of BF16 accuracy, and the remaining 26 layers (+5.3 bits) add one point, so

Table 8: **Preparation and runtime cost** (Qwen2.5-7B-Instruct): each method's offline stage under its default configuration, and per-token KV processing vs. naive NVFP4.

Method	Gradients	Search/fit	Offline GPU s	Offline peak GB	Runtime µs/token	vs. native
KIVI-4 (Liu et al., 2024c)	no	no	0	-	1.160	1.19×
GEAR-4 (Kang et al., 2024)	no	no	0	-	14.591	14.98×
KVQuant-4 (Hooper et al., 2024)	yes	yes (k-means)	17.0	49.3	2.792	2.87×
KVTuner-4 (Li et al., 2025)	no	yes (NSGA-II ×100)	7 783	15.5	1.007	1.03×
naive NVFP4 (no repair)	no	no	0	-	0.974	1.00×
Easy-Kuant (probe)	no	no	**1.8**	**15.3**	**0.920**	**0.94×**

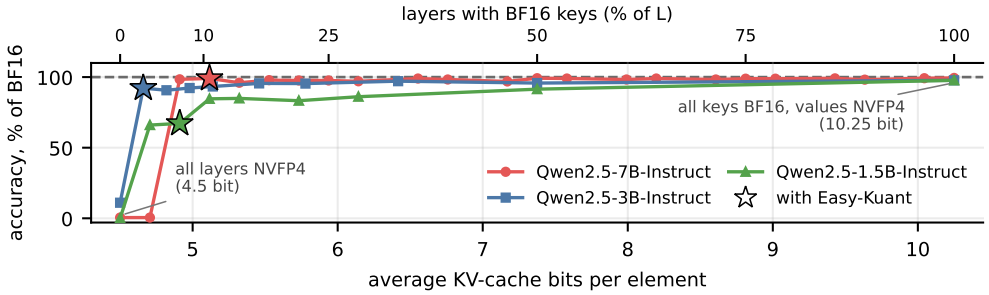


Figure 2: **Accuracy–memory trade-off between full NVFP4 and BF16 keys** (GSM8K). Each point keeps the keys of the k top-ranked layers in BF16 and everything else in NVFP4, placed at the resulting KV-cache bits per element; stars mark the operating point Easy-Kuant selects.

the curve is a step followed by a plateau and the $\tau{=}4$ rule sits right after the step, at 5.1 bits; the half point that remains with every key in BF16 is the cost of quantizing the values. Recovery depends on covering the responsible layers, not on the probe as such: every set in the sweep that contains layers 0 and 27 recovers and every set that misses layer 0 does not (Table 11). On 3B a single layer buys 92% and the other 35 add five points. On the 2-KV-head 1.5B model there is no step: the curve rises from 66% with two layers to 85% with three and reaches 98% only with every key protected, so the residual gap of §4.3 is a granularity limit of whole-layer protection rather than a mis-selected set.

What the probe reads, and how much the threshold matters. Figure 3 shows the statistic behind the rule, the largest absolute post-RoPE key of each layer on the calibration prompts. The profile is flat with a few spikes, at the layers of Table 2 and Figure 1: on 7B layers 27, 0 and 3 stand at 25×, 10× and 4.4× the median with the next layer at 3.8×; on 3B layer 0 stands at 6.8× with the next at 3.0×; on 1.5B layers 0 and 1 stand at 22× and 7× with the next at 3.9×. So $\tau{=}4$ falls in a gap of the profile on every model, and because the accuracy curve is a step its exact value matters little: on 7B any τ between 3.8 and 10.3 selects $\{0, 3, 27\}$ or $\{0, 27\}$, which are within a point of each other, a lower τ adds plateau layers at a memory cost only, and only a τ above 10.3, which drops layer 0, collapses; on 3B any τ between 3.0 and 6.8 selects the same single layer. The 2-KV-head 1.5B model is again the exception: lowering τ to 3 adds layer 15 and lifts accuracy from 67% to 85% of BF16, so there the rule trades accuracy for memory rather than sitting on a plateau.

When is quantization-aware tuning worth it? The QAT checkpoints of Table 3 (Appendix B.5) add +3.7, +1.1 and +0.1 GSM8K points over the training-free repair on 1.5B, 3B and 7B, and the gain tracks the gap that repair leaves (16, 8 and 4 points); they stay within two points on MMLU-STEM and are neutral to slightly negative on MATH-500, which the rollouts do not target. QAT is thus a targeted supplement for the small models, not the source of the repair, and a tuned checkpoint served *without* protection still collapses.

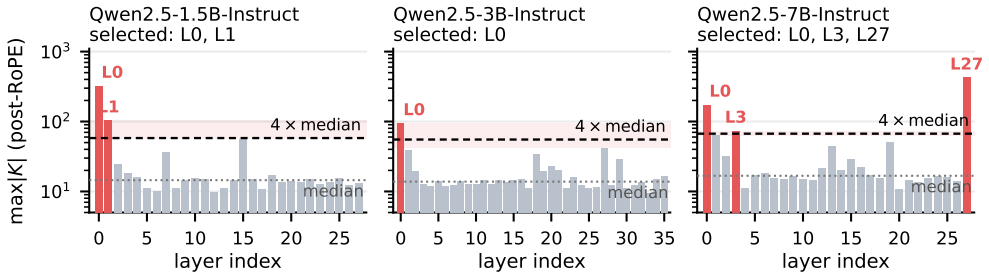


Figure 3: **What the probe reads.** Largest absolute post-RoPE key per layer on the calibration prompts (log scale); layers above four times the per-model median (dashed) are the ones Easy-Kuant protects, and every threshold inside the shaded band selects the same set.

5 Related Work

KV-cache quantization. KVQuant (Hooper et al., 2024) quantizes pre-RoPE keys per channel with non-uniform codes and a sparse outlier path; KIVI (Liu et al., 2024c) quantizes keys per channel and values per token in small groups; GEAR (Kang et al., 2024) adds a low-rank correction of the residual; KVTuner (Li et al., 2025) searches layer-wise precision assignments with a multi-objective optimizer; PM-KVQ (Liu et al., 2026) and KVarN (Muller et al., 2026) target long chains of thought and error accumulation. All are tuned against the reconstruction error of the cached tensor, and each needs a layout, kernel or search of its own, so they run beside rather than inside an engine's native low-bit path. We claim neither the first observation of layer sensitivity nor the first mixed-precision cache; we add the causal identification of the layers that turn a lossy format into a total failure, their key-side origin, and a policy that stays inside an existing low-bit path.

Outliers, formats and quantization-aware training. Outlier features (Dettmers et al., 2022) and massive activations (Sun et al., 2024) are the known obstacle to low-bit inference and motivate our statistic; weight and activation quantizers handle them by migrating scale (Xiao et al., 2023; Lin et al., 2024; 2025) or rotating the basis (Ashkboos et al., 2024; Liu et al., 2025; Su et al., 2025), and §3.3 shows why redistributing range does not repair a block-scaled cache format. FP8 and NVFP4 caches (NVIDIA, 2026) are served natively by vLLM (Kwon et al., 2023) with FlashInfer kernels (Ye et al., 2025), and vLLM's per-layer exemption is the only control Easy-Kuant needs. LLM-QAT, RoSTE and ReQAT (Liu et al., 2024b; Wei et al., 2025; Lee et al., 2026) apply straight-through training (Bengio et al., 2013) to weights and activations; our optional stage applies the same estimator to the post-RoPE cache on the model's own rollouts.

6 Conclusion and Limitations

Catastrophic KV-quantization failure on the Qwen2.5 family is neither diffuse damage nor an architectural fate. It is produced by the keys of one to four layers whose outlier channels dominate the shared block scales of a low-bit format, and it went unlocalized because cache reconstruction error, the quantity KV quantizers are tuned against, does not see it. Measuring at the output and intervening in both directions identifies a layer set that is sufficient both to cause and to repair the collapse, and Easy-Kuant turns that set into a list of integers that production engines already accept: a few forward passes, one threshold, no kernel, search or gradients. On the native serving path the list returns NVFP4 to 90–96% of BF16 accuracy on the 3B and 7B checkpoints and FP8 to 98–100% on all three, transfers unchanged to long-context and multimodal checkpoints, and costs $0.94\times$ the per-token KV processing of full NVFP4 at 85% of its cache capacity; an optional on-policy quantization-aware stage adds up to 3.7 points where a gap remains.

The layer is the unit that engines expose today and it suffices for the 3B and 7B checkpoints; for the 2-KV-head 1.5B model, whose accuracy–memory curve rises gradually, a head- or channel-level version of the same policy would close the remaining gap once engines expose it. The mechanism is established on one widely deployed family, and since the probe costs a few forward passes, the same measurement can precede any low-bit cache deployment on other families and formats.

REPRODUCIBILITY STATEMENT

Every table states its engine (native vLLM or simulation harness), sample count, quantizer and generation budget, and no row pools two of them. The probe (Algorithm 1) has one hyperparameter, τ=4, and eight fixed calibration prompts; the selected layer sets are listed in Table 3. Appendix A specifies both NVFP4 simulators, the evaluation protocols, the QAT recipe and the cost measurement; native and simulated accuracies are never compared with each other. Code, calibration prompts and per-sample outputs will be released.

AI USE STATEMENT

In this work, we used generative AI tools only to edit the manuscript for readability: improving grammar, wording and concision of text written by the authors, and checking LaTeX formatting. We have not used generative AI tools to generate data, develop the method or its theoretical analysis, formulate or prove mathematical claims, design experiments, implement methods, or analyse and interpret results. Every AI-assisted edit was reviewed by the authors against the original text, and all numbers and claims were checked against the experiment logs. We take responsibility for the final content of this work, including text produced with the aid of generative AI.

REFERENCES

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

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

Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, 2022.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets and Benchmarks Track*, 2021b.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems*, 2024.

Gregory Kamradt. Needle in a haystack – pressure testing LLMs. https://github.com/gkamradt/LLMTest_NeedleInAHaystack, 2023.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. GEAR: An efficient KV cache compression recipe for near-lossless generative inference of LLM. *arXiv preprint arXiv:2403.05527*, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, 2023.

Janghwan Lee, Sihwa Lee, Jinseok Kim, Yongjik Kim, Jieun Lim, Jinwook Oh, and Jungwook Choi. ReQAT: Achieving full-precision reasoning accuracy with 4-bit floating-point quantization-aware training. *International Conference on Machine Learning (ICML)*, 2026.

Xing Li, Zeyu Xing, Yiming Li, Linping Qu, Hui-Ling Zhen, Wulong Liu, Yiwu Yao, Sinno Jialin Pan, and Mingxuan Yuan. KVTuner: Sensitivity-aware layer-wise mixed-precision KV cache quantization for efficient and nearly lossless LLM inference. In *International Conference on Machine Learning (ICML)*, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. AWQ: Activation-aware weight quantization for LLM compression and acceleration. In *Proceedings of Machine Learning and Systems (MLSys)*, 2024.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. QServe: W4A8KV4 quantization and system co-design for efficient LLM serving. In *Proceedings of Machine Learning and Systems (MLSys)*, 2025.

Tengxuan Liu, Shiyao Li, Jiayi Yang, Tianchen Zhao, Feng Zhou, Xiaohui Song, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. PM-KVQ: Progressive mixed-precision KV cache quantization for long-CoT LLMs. In *International Conference on Learning Representations (ICLR)*, 2026.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. MMBench: Is your multi-modal model an all-around player? In *European Conference on Computer Vision (ECCV)*, 2024a.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. LLM-QAT: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics (ACL)*, 2024b.

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

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *International Conference on Machine Learning (ICML)*, 2024c.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*, 2024.

Lorenz K. Muller, Philippe Bich, Chiara Boretti, Hyun-Min Chang, Jiawei Zhuang, and Lukas Cavigelli. KVarN: Variance-normalized KV-cache quantization mitigates error accumulation in reasoning tasks. *arXiv preprint arXiv:2606.03458*, 2026. URL https://arxiv.org/abs/2606.03458.

NVIDIA. NVFP4: Transformer engine 2.16.0 documentation. https://docs.nvidia.com/deeplearning/transformer-engine-releases/release-2.16/user-guide/features/low_precision_training/nvfp4/nvfp4.html, 2026. Accessed September 7, 2026.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025a.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025b.

Zunhai Su, Zhe Chen, Wang Shen, Hanyu Wei, Linge Li, Huangqi Yu, and Kehong Yuan. RotateKV: Accurate and robust 2-bit kv cache quantization for llms via outlier-aware adaptive rotations. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2025.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. In *Conference on Language Modeling (COLM)*, 2024.

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

Quan Wei, Chung-Yiu Yau, Hoi-To Wai, Yang Katie Zhao, Dongyeop Kang, Youngsuk Park, and Mingyi Hong. RoSTE: An efficient quantization-aware supervised fine-tuning approach for large language models. In *International Conference on Machine Learning (ICML)*, 2025.

xAI. RealWorldQA: Grok-1.5 vision preview. https://x.ai/news/grok-1.5v, 2024.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning (ICML)*, 2023.

Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai, Keqin Chen, Jialin Wang, Yang Fan, Kai Dang, et al. Qwen2.5-Omni technical report. *arXiv preprint arXiv:2503.20215*, 2025.

Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. FlashInfer: Efficient and customizable attention engine for LLM inference serving. In *Proceedings of Machine Learning and Systems (MLSys)*, 2025.

A Experimental Details

A.1 Quantizers

The native path uses vLLM's NVFP4 and FP8 KV caches as implemented by FlashInfer on Blackwell; we do not modify them. The simulation harness fake-quantizes post-RoPE keys and values inside the attention call and has two NVFP4 variants. The *block-only* simulator uses $s_B = \mathrm{E4M3}(\max_{i\in B}|x_i|/6)$ and $\widehat{x}_i = s_B\, Q_{\mathrm{E2M1}}(x_i/s_B)$ for each block B of 16 elements. The *hierarchical* simulator adds the documented global scale,

$$s_{\mathrm{global}} = \frac{\max_i |x_i|}{448\cdot 6}, \qquad \bar{s}_B = \mathrm{E4M3}\!\left(\frac{\max_{i\in B}|x_i|}{6\, s_{\mathrm{global}}}\right), \qquad \widehat{x}_i = s_{\mathrm{global}}\, \bar{s}_B\, Q_{\mathrm{E2M1}}\!\left(\frac{x_i}{s_{\mathrm{global}}\bar{s}_B}\right), \tag{4}$$

with intermediate scales in FP32 and tiny scales clamped. The global maximum is taken over the tensor supplied to each call, so in a full-sequence hook a later token can change the scale applied to earlier entries. Neither simulator has been bit-matched to the serving kernel; what we establish is that every mechanism conclusion holds under both (Appendix B.1). Benchmark runs in the simulator use an SDPA attention path; the eager path of `transformers` 5.3 degrades Qwen2 generation on its own and is used only for the per-layer replays, whose clean reference is computed on the same path.

A.2 Evaluation Protocols

Native deployment. vLLM with FlashInfer 0.6 on one B300, greedy decoding, full test sets: GSM8K (n=1319, 8-shot chat prompt with a `\boxed{}` instruction, at most 640 new tokens), MATH-500 (n=500, 1024 new tokens, symbolic equivalence scoring), and MMLU-STEM (n=3153, few-shot completion format, scored by the log-likelihood of the four option letters, so no decoding is involved). Quantized caches are enabled with `kv_cache_dtype` and protection with `kv_cache_dtype_skip_layers`; nothing else changes between rows.

Simulation harness. The prior KV-quantization methods we compare against are available as research implementations rather than as vLLM cache backends, so a like-for-like comparison cannot be run on the native path. We therefore built a single quantization-simulation harness in which every method, including ours, is expressed as a fake-quantizer applied to the post-RoPE keys and values; all methods see the same prompts, scorers, generation budgets and batch composition (length-sorted batches of 16), so that the only difference between two columns of Table 7 is the quantization scheme. KIVI-4, KVQuant-4 and GEAR-4 use their repositories' default 4-bit configurations. KVTuner is a layer-wise precision preset released per architecture; we use the official 36-layer preset for Qwen2.5-3B and Qwen2.5-VL-3B and the official 28-layer preset for the 7B models, and for the two architectures without an official preset (Qwen2.5-Omni-3B and Qwen3-VL-8B) we transfer the 28-layer Qwen2.5-7B preset, which is why those cells carry a † in Table 15. Multimodal benchmarks: MMBench-DEV-EN restricted to the 1164 questions that carry an image (64 new tokens), MathVista-mini (1000; 128 new tokens for the 3B and Omni models, 512 for VL-7B and Qwen3-VL, whose answers are otherwise truncated), RealWorldQA (765; 64 new tokens) and GSM8K through the language tower. Long context: passkey retrieval at 4k, 8k and 16k with the needle at three depths and eight seeds (72 items), and the five LongBench-E subsets of Table 16 with their official prompts and metrics, 64 new tokens, no chat template, batch 4.

Mechanism measurements. Per-layer replays use 128 packed 2048-token sequences, half mathematical question–answer text and half SlimPajama, disjoint from the calibration prompts, with 64 query positions sampled per sequence; each layer's attention is recomputed from its clean BF16 Q, K, V with the stated tensors quantized and the perturbed output discarded. The fixed-token propagation runs use 32 (48 for the block-only arm) such sequences in a whole-sequence causal forward with the same hook, and measure the keys the quantized model computes at position 2047 against the clean keys at the same position. The two-directional interventions of Figure 1 use free generation on GSM8K (n=128, SDPA hook, block-only simulator) and are the only accuracy measurement in the paper that is not on a full test set; the full-test budget sweep of Figure 2 reproduces their conclusion.

Table 9: Agreement between the block-only (B) and hierarchical (H) NVFP4 simulators on the same clean activations: Pearson correlation of the layerwise output NMSE, the worst-layer NMSE, and the four worst layers of each.

Model	Pearson r	Max NMSE B	Max NMSE H	Worst four B	Worst four H
1.5B	0.947	0.679	0.376	0,15,1,18	15,0,1,14
7B	0.970	0.444	0.332	27,0,17,3	0,27,3,1

A.3 Probe and Tuning Settings

The probe (Algorithm 1) uses eight mathematical calibration prompts (694 tokens), accumulates the per-layer maximum absolute post-RoPE key over all tokens, heads and channels, and protects every layer above $4\times$ the per-model median; the same τ and prompts are used for every model, benchmark and modality, and no accuracy is ever used to choose a set. Table 3 lists the selected sets.

The optional tuning stage generates about one thousand continuations with the BF16 checkpoint from mathematical prompts (DAPO), keeps only the response tokens in the loss, and fine-tunes all weights with a straight-through fake quantizer on the post-RoPE K/V of the exposed layers, for 600 steps at learning rate $3{\cdot}10^{-5}$ (Table 12). Deployment always protects the probe-selected set of the tuned checkpoint.

A.4 Cost Measurement

Offline costs in Table 8 are measured on one B300 after loading and warm-up, with device synchronization at both boundaries. The probe runs its eight prompts once; KVQuant's stage runs its Fisher-information pass over 16 sequences of 2048 tokens (forward and backward) followed by per-tensor k-means; KVTuner's stage runs one trial of its NSGA-II search (a 20-question 4-shot GSM8K evaluation) and multiplies by the published default of 100 trials. Because `scikit-learn` was unavailable, the k-means step used an equivalent one-dimensional Lloyd implementation on subsampled data, so its CPU time is a lower bound. Runtime is the per-token cost of quantizing and dequantizing the prefill KV of Qwen2.5-7B at batch 8 in one harness, relative to stock NVFP4; KIVI's decode path is excluded because its residual window leaves single tokens unquantized. Serving throughput (Table 6) uses vLLM on the same device in two regimes: batch 16 with `gpu_util`=0.85, so that the cache always fits, and batch 384 with 2048+512 tokens and a 17 GB pool (`gpu_util`=0.13). Allocated bytes per cached token, including scales, padding and metadata, are the authoritative memory figure; the b_{eff} expression of §3.2 is an approximation.

B Additional Results

B.1 Agreement Between the Two NVFP4 Simulators

Table 9 evaluates the block-only and hierarchical simulators on the same clean activations. Both concentrate the damage in a few layers, their layerwise output NMSEs correlate at Pearson r=0.94–0.97, and adding the global scale lowers the worst-layer damage (from 0.679 to 0.376 on 1.5B and from 0.444 to 0.332 on 7B) without changing which layers are worst; the fixed-token propagation measurements of Table 10 likewise agree under both. The global scale is therefore not a repair on its own.

B.2 Outlier Interventions Within a Layer

Within a damaged layer the error can come from the outlier's own rounding or from the block scale it inflates. For each block we fix the top-k absolute entries ($k \in \{1, 2, 4\}$) and either *restore* them to BF16 while leaving the block scale and all other dequantized values untouched, or *isolate* them and recompute the scale of the remaining entries before quantizing. Writing D_K for the K-only output NMSE and D_{int} for the intervened one, recovery is $(D_K - D_{\text{int}})/D_K$. Three regimes appear: recomputing the scale recovers 57–76% of the error on 7B layer 3 (scale pollution dominates), restoring the outlier's value alone recovers about half on 1.5B layer 1 and 7B layer 17 (the outlier's

Table 10: Fixed-token propagation under both simulators (B/H). Key drift is the NMSE between the keys a quantized model newly computes at a late position and the clean BF16 keys; KL is the output-distribution KL to BF16 (nats).

	Key drift (B/H)		Output KL (B/H)	
Model	All quantized	Protect K/V	All quantized	Protect K/V
1.5B	0.622 / 0.394	0.018 / 0.018	6.98 / 3.56	0.12 / 0.12
7B	0.453 / 0.413	0.013 / 0.016	5.85 / 5.63	0.12 / 0.12

Table 11: Protection-budget sweep behind Figure 2 (GSM8K). Entries are k: accuracy (%), where the k layers with the largest post-RoPE $\max|K|$ keep their keys in BF16 and everything else is NVFP4; boxed entries are the sets chosen by the $\tau{=}4$ rule.

Model (L; BF16)	k: accuracy							
1.5B-Inst (28; 67.3)	0: 0.0	1: 44.4	$\boxed{\text{2: 45.2}}$	3: 56.9	4: 57.1	6: 56.0	8: 57.9	14: 61.5
	28: 65.7							
3B-Inst (36; 82.5)	0: 9.1	$\boxed{\text{1: 76.0}}$	2: 74.8	3: 76.1	4: 77.0	6: 78.8	8: 78.7	12: 80.1
	18: 79.0	36: 80.6						
7B-Inst (28; 87.2)	0: 0.5	1: 0.5	2: 85.8	$\boxed{\text{3: 86.4}}$	4: 83.7	5: 85.3	6: 85.1	7: 85.1
	8: 84.6	10: 86.3	11: 85.7	13: 84.5	14: 86.5	15: 86.3	17: 85.7	18: 86.3
	20: 85.7	21: 86.2	22: 86.2	24: 86.4	25: 85.6	27: 86.5	28: 86.7	

own precision matters), and on 1.5B layer 0 and 7B layer 0—the two most sensitive layers—neither recovers much, so the damage there is not a single-outlier effect. This is why the repair protects the whole key tensor of a layer rather than a sparse set of entries.

B.3 Error Propagation During Decoding

This experiment asks whether the damage of a quantized layer stays inside that layer or feeds back through the cache. The confound is that a quantized model generates different text, so we remove generation from the picture: the BF16 model and the quantized model are given the *same* fixed 2048-token sequences, and we compare the keys that each model computes for the last position—keys that have not yet been quantized, so any difference between them is inherited from earlier layers and earlier positions. If the damage stayed local, these freshly computed keys would agree up to the error of a single cache write (NMSE about 0.009). Instead they differ by 0.45–0.62 on the 7B and 1.5B models (Table 10), and the two models' output distributions differ by 6–7 nats of KL. Protecting only the probe-selected layers brings the drift down to 0.013–0.018 and the KL to 0.12, i.e. to the level of the write error itself. The reading is direct: a corrupted attention output enters the residual stream, changes the keys the model writes for later layers and positions, and those keys are quantized again, so the error compounds along the sequence; removing the few source layers breaks the loop. The numbers are reported under both NVFP4 simulators (B/H) and agree. Two caveats keep this a qualitative result: the write error and the drift are normalized by different reference tensors, so they cannot be added into an exact error budget, and the hierarchical simulator's global scale is not token-local (Appendix A.1).

B.4 Protection Budget

Table 11 gives the budget sweep behind Figure 2: the k layers with the largest probe statistic (Figure 3) keep their keys in BF16 and everything else is NVFP4. Recovery is a property of covering the responsible layers, not of the probe as such: any selection that contains layers 0 and 27 of 7B recovers, and any selection that misses layer 0 does not; the probe is the cheapest reliable way of finding them.

Table 12: QAT checkpoints behind the +QAT columns (GSM8K, accuracy %); "exposure" states which layers the training forward pass fake-quantized.

Model	Exposure	Own BF16	NVFP4	NVFP4+Easy-Kuant	Untrained NVFP4+Easy-Kuant
1.5B-Inst	all layers	67.0	2.8	**61.2**	45.2
3B-Inst	all layers	81.2	60.0	**78.1**	76.0
7B-Inst	protected exempt	90.0	0.3	**88.3**	86.4

B.5 Optional Quantization-Aware Tuning

With $p_l \in \{0, 1\}$ indicating whether layer l is exempt during training, the fake quantizer applied to each post-RoPE tensor $X_l \in \{K_l, V_l\}$ is

$$\widetilde{X}_l = \begin{cases} X_l, & p_l = 1, \\ X_l + \mathrm{sg}\left(\mathrm{DQ}(\mathrm{Q}(X_l)) - X_l\right), & p_l = 0, \end{cases} \tag{5}$$

where sg is stop-gradient, so the forward pass sees the deployed cache and the backward pass is the straight-through estimator. "All layers" sets p_l=0 everywhere; "protected layers exempt" sets p_l=1 on the probe-selected set.

QAT checkpoints. Table 12 lists the checkpoints behind the +QAT columns of the main text: each is fine-tuned for 600 steps at learning rate $3{\cdot}10^{-5}$ on about one thousand of its own BF16 rollouts, with the fake quantizer of Eq. (5) applied to the layers stated in the "exposure" column. Two facts about deployment are worth recording. The probe run on the tuned checkpoint may select a superset of the layers it was trained with ($\{0, 1, 15\}$ vs. $\{0, 1\}$ on 1.5B, $\{0, 3, 27\}$ vs. $\{0, 27\}$ on 7B), and deploying the superset is better (61.2% vs. 59.3%, 88.3% vs. 88.1%), so we rerun the probe rather than pin the training set. And a QAT checkpoint deployed *without* protection still collapses in vLLM (6.8% and 19.6% GSM8K on 1.5B and 7B; Table 14), with 3B (55.6%) the only partial exception: tuning does not replace the layer list.

B.6 Serving Throughput Under Load

Table 13 sweeps batch size and response length for the 7B QAT checkpoint served as NVFP4 with layers $\{0, 27\}$ protected, against BF16, in the tight-pool regime. Throughput depends on memory pressure and scheduling and is not monotone in load: the speedup grows from $1.04\times$ at batch 32 to $1.70\times$ at batch 384 and falls to $1.41\times$ at 512, and grows with response length from $0.98\times$ at 128 output tokens to $1.64\times$ at 2048. These are workload-specific gains from holding more tokens, not a kernel speedup, and this configuration protects two layers where the accuracy tables protect three.

B.7 Transformations That Do Not Repair the Collapse

§3.3 summarizes four zero-cost alternatives; this section records the measurements. *Scale migration*: dividing the outlier key channels by a factor folded into W_k and multiplying the matching query channels by the same factor leaves attention scores unchanged in exact arithmetic, but under a block-shared E2M1 scale the migrated channel's neighbours now set the step and the query side re-amplifies the error; a naive variant raised the perplexity ratio under NVFP4 from about $15\times$ to $210\times$, and a block-equalized variant that stops migrating once the channel is level with its neighbours was neutral. *Attention-sink protection*: keeping the first 4 or 16 cache tokens in BF16 does not change the collapse, consistent with the outlier mass being channel-structural (top-8 channels carry all of it; the largest 16 tokens carry 8.5%; sink positions are $0.76\times$ the rest). *Block-aware permutation*: reordering channels so that outlier channels share as few 16-element blocks as possible, folded into W_k/W_q and the RoPE frequency table, lowers the fraction of blocks whose maximum exceeds eight times their median from 0.133 to 0.079 but leaves attention-output error unchanged or slightly worse, because the eight outlier channels of a 128-dimensional head already fall into roughly one of its eight blocks. *Protected-key format*: keeping the selected layers' keys in FP8 E4M3 instead of BF16 collapses whether the scale is per tensor or per channel, consistent with all-layer FP8 collapsing 7B in Table 14. E4M3 has three mantissa bits, so a finer scale cannot reduce its relative rounding

Table 13: Throughput (output tokens/s) of the QAT checkpoint served as NVFP4+Easy-Kuant (layers $\{0, 27\}$ protected) against BF16 with a 17 GB KV pool.

Load sweep: 2048 input tokens, 512 output tokens			
Batch	BF16	Protected NVFP4	Speedup
32	6054	6291	1.04×
64	10438	11321	1.08×
128	11900	15079	1.27×
256	12974	16911	1.30×
384	10271	17480	1.70×
512	10184	14381	1.41×
Response sweep: batch 256, 1024 input tokens			
Output tokens	BF16	Protected NVFP4	Speedup
128	8453	8304	0.98×
256	27916	28010	1.00×
512	21620	27694	1.28×
1024	17670	25734	1.46×
2048	16405	26855	1.64×

Table 14: Complete native vLLM results (accuracy %). "QAT" rows load the optional QAT checkpoint and report *its own* BF16 accuracy; +Easy-Kuant protects the probe-selected layers.

		GSM8K					MATH-500					MMLU-STEM				
Model	Weights	BF16	NVFP4	+Easy-Kuant	FP8	+Easy-Kuant	BF16	NVFP4	+Easy-Kuant	FP8	+Easy-Kuant	BF16	NVFP4	+Easy-Kuant	FP8	+Easy-Kuant
1.5B-Inst	base	73.7	0.2	57.8	15.1	72.6	53.8	0.0	39.4	7.0	48.2	54.7	27.1	52.4	44.8	53.9
1.5B-Inst	QAT	70.7	6.8	61.5	46.9	70.7	45.0	2.6	35.2	20.2	44.2	54.1	35.8	50.4	45.0	52.9
3B-Inst	base	85.9	1.1	77.7	81.9	85.8	61.8	0.0	52.6	57.8	62.2	62.2	27.7	58.6	61.0	60.9
3B-Inst	QAT	83.4	55.6	78.8	81.7	83.3	56.0	36.4	51.4	57.2	56.0	61.6	46.7	59.6	59.3	61.1
7B-Inst	base	90.2	11.0	86.4	2.2	88.3	65.4	3.0	60.0	0.6	61.2	70.7	30.2	64.6	16.8	67.4
7B-Inst	QAT	90.5	19.6	86.5	1.7	88.2	68.4	4.4	59.2	0.6	61.0	70.8	32.7	65.3	21.7	67.6

error below the 6.25% bound of its normal range, and values that underflow are lost entirely; which of the two dominates here we have not isolated, and the bound alone does not predict the size of the collapse (Appendix C). Post-RoPE channel scaling also cannot be assumed to fold into pre-RoPE weights without checking that it commutes with RoPE. These outcomes apply to the tested implementations; they do not show that scaling or rotation must fail in general, only that redistributing dynamic range does not supply the mantissa precision the outlier channels need.

B.8 Complete Result Tables

Table 14 gives every native vLLM cell, including the FP8 columns for MATH-500 and MMLU-STEM and each QAT checkpoint's own BF16 accuracy; Table 15 gives the multimodal suite with all baselines; and Table 16 gives all five LongBench-E subsets. Bare NVFP4 scores 0.00–0.04 on every LongBench subset of every model, while the differences between protected NVFP4 and BF16 are within the noise of n=150–300 items in both directions (e.g. -3.1 HotpotQA F1 on 7B, $+5.8$ on 1.5B), with the exception of passage retrieval on 1.5B (0.09 against 0.22), the same 2-KV-head model that trails on long generation.

Table 15: Complete multimodal results (accuracy %). †: no official KVTuner preset exists for this architecture, so the 28-layer Qwen2.5-7B preset is transferred (Appendix A.2). Omni and Qwen3-VL QAT checkpoints were not trained.

Model	Task	BF16	NVFP4	KIVI-4	KVQ-4	GEAR-4	KVTuner	ours	ours+QAT
Qwen2.5-VL-3B	GSM8K	81.7	0.0	81.4	81.8	81.3	81.7	74.2	76.2
	MMBench	83.0	0.0	83.0	82.8	83.2	82.8	83.2	82.6
	MathVista	46.1	0.1	46.6	47.3	46.5	47.9	44.0	48.1
	RealWorldQA	63.7	0.3	63.5	64.2	64.3	64.1	64.7	63.5
Qwen2.5-VL-7B	GSM8K	49.6	0.2	49.8	50.4	49.4	49.6	50.1	45.8
	MMBench	86.7	19.2	86.9	86.3	86.9	86.8	86.3	86.7
	MathVista	31.7	2.9	30.8	31.3	32.7	32.0	32.7	26.0
	RealWorldQA	69.3	5.0	69.3	69.2	69.2	67.6	69.3	67.1
Qwen2.5-Omni-3B	GSM8K	81.9	1.4	81.6	80.7	81.4	79.4†	75.4	–
	MMBench	83.3	72.0	83.0	83.0	82.7	81.7†	81.4	–
	MathVista	58.7	29.3	58.9	57.3	58.6	56.6†	55.2	–
	RealWorldQA	65.0	54.8	65.6	65.1	65.5	62.4†	62.5	–
Qwen2.5-Omni-7B	GSM8K	87.0	0.2	87.2	87.5	87.3	87.2	86.2	–
	MMBench	87.0	26.7	86.9	86.8	86.8	85.8	86.5	–
	MathVista	55.6	8.0	56.8	56.7	56.1	54.9	53.9	–
	RealWorldQA	68.9	28.0	69.0	69.9	68.9	67.6	69.4	–
Qwen3-VL-8B	GSM8K	91.9	90.8	91.7	91.3	91.7	91.3†	91.1	–
	MMBench	89.4	88.3	89.2	89.1	89.4	88.1†	88.5	–
	MathVista	54.4	53.9	53.7	52.8	53.3	49.5†	–	–
	RealWorldQA	73.1	70.3	72.3	72.8	72.4	70.2†	71.4	–

Table 16: Complete long-context results: NIAH passkey retrieval (4k–16k) and the five LongBench-E subsets with their official metrics (Appendix A.2).

	NIAH			Passage retrieval			HotpotQA		
Model	BF16	NVFP4	Easy-Kuant	BF16	NVFP4	Easy-Kuant	BF16	NVFP4	Easy-Kuant
1.5B-Inst	1.000	0.000	0.875	0.222	0.000	0.092	0.181	0.007	0.239
3B-Inst	1.000	0.375	0.986	0.352	0.021	0.279	0.313	0.036	0.326
7B-Inst	1.000	0.000	1.000	0.937	0.031	0.887	0.543	0.003	0.512
	Qasper			2WikiMQA			MultiFieldQA		
1.5B-Inst	0.230	0.011	0.207	0.168	0.004	0.193	0.227	0.011	0.222
3B-Inst	0.300	0.033	0.312	0.294	0.014	0.250	0.255	0.041	0.360
7B-Inst	0.370	0.006	0.382	0.372	0.008	0.320	0.420	0.009	0.382

C Block-Scale Distortion and Local Attention Sensitivity

This appendix gives two elementary results that help interpret the measurements in §2.1: an exact lower bound on reconstruction error caused by the zero cell of a block-scaled quantizer, and a local expansion of attention-output error. The first result identifies a possible source of damage to ordinary components; the second describes how key and value errors reach the output. Neither result alone establishes which layers cause end-to-end failure or guarantees that key errors outweigh value errors. Those conclusions are evaluated by the replay and intervention experiments in §2.

C.1 Zero-Cell Distortion Under a Shared Block Scale

NVFP4 represents each value using an E2M1 codeword and scaling factors, including a shared scale for each block of 16 elements (NVIDIA, 2026). Let

$$\mathcal{C}_+ = \{0, \tfrac{1}{2}, 1, \tfrac{3}{2}, 2, 3, 4, 6\}, \qquad \mathcal{C} = \mathcal{C}_+ \cup (-\mathcal{C}_+),$$

and let $Q_{\mathcal{C}}$ denote deterministic nearest-codeword rounding. We first consider an idealized block $x \in \mathbb{R}^{16}$ with $A = \|x\|_\infty > 0$ and an exactly represented scale $s = A/6$, so that $\widehat{x}_i = s\, Q_{\mathcal{C}}(x_i/s)$. Strict inequalities below avoid dependence on the tie-breaking rule.

Proposition C.1 (Zero-cell distortion)**.** *For every component, $|x_i| < A/24$ implies $\widehat{x}_i = 0$. Consequently,*

$$\|\widehat{x} - x\|_2^2 \;\geq\; \sum_{i=1}^{16} x_i^2\, \mathbf{1}\{|x_i| < A/24\}. \tag{C.1}$$

Proof. The midpoint between the codewords 0 and $1/2$ is $1/4$. By symmetry, $|x_i/s| < 1/4$, equivalently $|x_i| < A/24$, rounds to zero and contributes exactly x_i^2 to the squared error. All remaining contributions are nonnegative. □

Corollary C.2 (Expected loss of ordinary components)**.** *Fix $A > 0$ and $0 < u \leq A$. Suppose a block contains an entry of magnitude A, all its entries have magnitude at most A, and an ordinary component has distribution $X \sim \mathrm{Unif}[-u, u]$. With the same fixed scale $s = A/6$, define $\widehat{X} = sQ_{\mathcal{C}}(X/s)$. Then*

$$D_{\mathrm{bulk}}(A; u) := \mathbb{E}[(\widehat{X} - X)^2] \;\geq\; \frac{\min\{A/24, u\}^3}{3u}.$$

If $A > 24u$, this component is rounded to zero throughout its support, and $D_{\mathrm{bulk}}(A; u) = \mathbb{E}[X^2] = u^2/3$.

Proof. Set $b = \min\{A/24, u\}$. The componentwise implication in Proposition C.1 gives

$$D_{\mathrm{bulk}}(A; u) \geq \frac{1}{2u} \int_{-b}^{b} x^2 \, \mathrm{d}x = \frac{b^3}{3u}.$$

The endpoints have zero probability. If $A > 24u$, the entire support lies strictly inside the zero cell, so $\widehat{X} = 0$ and the stated equality follows. □

Interpretation and scope. Under these assumptions, an ordinary component can lose all of its signal energy even though the block maximum is represented exactly. The normalized lower bound is

$$\frac{D_{\mathrm{bulk}}(A; u)}{\mathbb{E}[X^2]} \geq \min\left\{\frac{A}{24u}, 1\right\}^3,$$

which depends on the within-block contrast A/u, rather than on A alone. The uniform distribution is an illustrative model, not an assumption about all measured activations. The growing lower bound does not by itself establish monotonicity of the full expected error for arbitrary activation distributions. In particular, a measured contrast of 20 does not imply complete erasure under the sufficient condition $A > 24u$, and a measured median magnitude cannot be substituted for the support bound u without an additional assumption.

Effective scales in deployment. For a positive effective scale s_{eff}, define

$$\widehat{x}_i = s_{\text{eff}} Q_{\mathcal{C}}(x_i / s_{\text{eff}}).$$

The exact implication is then

$$|x_i| < s_{\text{eff}}/4 \quad \Longrightarrow \quad \widehat{x}_i = 0.$$

In the hierarchical NVFP4 representation, s_{eff} is the product of the global scale and the stored block scale. Rounding the block scale can shift this threshold relative to $A/24$; the latter constant assumes the ideal scale $A/6$. Thus, applying the zero-cell argument to a deployed implementation requires its actual scale and rounding convention.

Implications for probing and rescaling. A layer's maximum key magnitude is an inexpensive proxy for identifying potentially problematic blocks in the recurring outlier patterns observed in our experiments. It does not determine the distribution of block contrasts, the fraction of affected components, or the sensitivity of the attention output. The usefulness of this proxy and of the protection rule is therefore established empirically, not by Proposition C.1 alone. Likewise, channel rescaling changes both the quantization scales and the way errors enter the key–query product. Reducing a key channel's magnitude need not reduce output error, but the proposition does not rule out beneficial rescaling. The unsuccessful variants in §3.3 are observations about the tested transformations and settings, not an impossibility result.

Distinction from FP8. The zero-cell constant above is specific to the E2M1 codebook. For E4M3 with nearest rounding, a standard relative-error bound of $2^{-4} = 6.25\%$ applies to values whose scaled magnitudes lie in the normal representable range, without clipping. This is an upper bound, not a fixed error incurred by each component. It does not apply uniformly to subnormal or underflowing values; a nonzero value rounded to zero has relative error 100%. Consequently, this bound alone does not explain the FP8 collapses reported in §3.3 and Table 3.

C.2 From Cache Error to Attention-Output Error

Consider one query head and the n key–value positions visible to its query. Fix $q \in \mathbb{R}^d$, $K \in \mathbb{R}^{n \times d}$, and $V \in \mathbb{R}^{n \times d_v}$. Using column vectors for the query and output, write

$$z = Kq/\sqrt{d}, \qquad a = \operatorname{softmax}(z), \qquad o = V^\top a, \qquad J_a = \operatorname{diag}(a) - aa^\top.$$

For perturbations E^K, E^V of the same shapes as K, V, respectively, define

$$\widehat{o} = (V + E^V)^\top \operatorname{softmax}((K + E^K)q/\sqrt{d}), \qquad \delta o = \widehat{o} - o.$$

This isolates cache perturbations with a fixed query and fixed visible positions; it does not model changes to the query or later layers.

Proposition C.3 (Local attention sensitivity)**.** *For sufficiently small perturbations,*

$$\delta o = V^\top J_a \frac{E^K q}{\sqrt{d}} + (E^V)^\top a + R, \qquad \|R\|_2 \le C\,(\|E^K\|_F + \|E^V\|_F)^2, \tag{C.2}$$

where C is finite on a fixed bounded neighbourhood of (K, V) and may depend on that neighbourhood and on q. If only key j is perturbed by $\varepsilon_j \in \mathbb{R}^d$, with $E^V = 0$, then $\delta o = a_j(v_j - o)q^\top \varepsilon_j/\sqrt{d} + R$, where v_j is the transpose of row j of V.

Proof. The map $F(K, V) = V^\top \operatorname{softmax}(Kq/\sqrt{d})$ is smooth for fixed q. Its differential in direction (E^K, E^V) is $V^\top J_a E^K q/\sqrt{d} + (E^V)^\top a$. Taylor's theorem on a bounded neighbourhood with bounded second derivative gives the stated quadratic remainder bound. For a single perturbed key, $E^K q = e_j q^\top \varepsilon_j$, where e_j is the jth coordinate vector, and $V^\top J_a e_j = a_j(v_j - o)$. □

Key sensitivity is a property of the complete mapping. A key error is projected onto the query and then transformed by J_a and $V^\top$. The softmax Jacobian itself satisfies $\|J_a\|_2 \le 1/2$: it is symmetric, and its absolute row sums are $2a_j(1 - a_j) \le 1/2$. Here matrix $\|\cdot\|_2$ denotes the spectral norm. Hence the first-order key term obeys

$$\left\| V^\top J_a \frac{E^K q}{\sqrt{d}} \right\|_2 \le \frac{\|V\|_2 \|q\|_2}{2\sqrt{d}} \|E^K\|_F.$$

Large or small output effects therefore depend on the complete mapping, including the query direction and the value contrasts $v_j - o$, rather than on amplification by the softmax Jacobian alone. The expansion describes these dependencies; it does not guarantee that key errors dominate value errors in every model or layer.

Value averaging is an absolute-error statement. If only V is perturbed, the attention weights remain fixed and the output change is exactly

$$\delta o_V = (E^V)^\top a = \sum_{j=1}^{n} a_j e_j^V, \qquad \|\delta o_V\|_2 \le \sum_{j=1}^{n} a_j \|e_j^V\|_2 \le \max_j \|e_j^V\|_2,$$

where e_j^V is the transpose of row j of E^V. This convex-combination bound applies to one query before the output projection. It does not bound relative output error independently of the clean output magnitude, which can be small because value contributions cancel.

Relation to the measured output NMSE. Our experiments measure the attention output after W_O. In a column-vector convention, let $y = W_O[o_1; \ldots; o_H]$ denote the projected concatenation of the H head outputs. Then

$$\delta y = W_O[\delta o_1; \ldots; \delta o_H], \qquad \|\delta y\|_2 \le \|W_O\|_2 \left(\sum_{h=1}^{H} \|\delta o_h\|_2^2 \right)^{1/2}.$$

Both the output projection and the clean-output denominator matter for output NMSE. Cache NMSE summarizes squared reconstruction error relative to cache energy; it does not specify error directions, queries, attention weights, value contrasts, or the output projection. Equal cache NMSE therefore does not determine an ordering of output damage. The differences in Table 2, including key-dominated damage at the most sensitive layers, are empirical findings consistent with this sensitivity analysis. The quadratic remainder need not be small at the largest observed perturbations, so we use the expansion to interpret local dependencies, not to numerically predict collapse. End-to-end claims rely on direct replay and intervention rather than on Eq. (C.2) alone.