How Small Can We Go? Calibrating Activation-Cache Compression for SAE Training

Abraham Yeung
Stanford University
ayeung16@stanford.edu

Abstract

To train a sparse autoencoder (SAE), you need a massive cache of model activations. We look into the effects that compression has on an SAE in two ways: we keep the original fp32-trained SAE and pass the compressed cache into the same SAE to see how many feature activations stay the same (the frozen-SAE test), and we retrain an SAE on the compressed cache and see how many of the original features still exist (the retraining test). We found that in the retraining test, there is no clear scale for retained features. On a GPT-2 cache, rounding everything to fp16 keeps 60.6% of features, but changing one value in a million by the smallest step fp32 allows (a change eight orders of magnitude smaller) keeps only 70.5%. Meanwhile, changing the random seed keeps 23.0%. For the frozen-SAE test, the percentage of features from the codecs that match the original increases in order of how precise the codecs are, but plain L_2 distance can tell us the same result. We also see the same frozen reading correspond to different percentages retained at different sites. So, what is the correct measurement? We build one by calibrating: retraining on identical bytes sets the upper bound of reasonable variance, and a reseed sets the floor. We then measure the effect of each compressed codec against its own site's floor. With TopK SAEs on GPT-2 small and Pythia-1.4B at research-scale budgets, naïve per-token int4 loses more features than a reseed everywhere we tested, as do PCA and random projection on GPT-2. Standardizing each channel before quantizing brings int4 back above the floor in terms of retained features, at the same file size. A bf16 forward pass, which many pipelines use for speed, loses slightly more features than rotated 8-bit storage. Calibrating a site costs four training runs and two evaluation passes.

1 Introduction

One way of understanding the internals of LLMs is to train SAEs (sparse autoencoders) on model internal activations. However, these activations take up a lot of space to store. A token of text is two bytes, but its activation vector can be several kilobytes (1). This means that training a single SAE can take over 1 GiB/s of disk read to avoid starving the GPU (2). So, ideally we would be able to cache the activations and compress them. The primary library used by researchers stores these activations in fp32 (3), but we do not know what the effect of compression is on the features of the SAE. The closest prior work quantizes model weights rather than the cache (4); separately, Paulo & Belrose established the seed instability we build on (5) (Appendix H). An SAE learns a dictionary of sparse features, directions in activation space that fire on some tokens; two SAEs agree on a feature when the other SAE has a feature whose activations over the same tokens correlate with it at $r > 0.9$. We run both tests from the abstract: the frozen-SAE test (same SAE, compressed cache) and the retraining test (new SAE trained on the compressed cache).

Preprint.

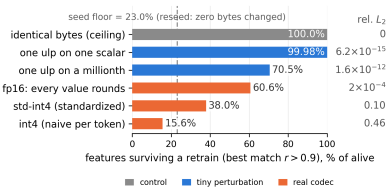

Figure 1: GPT-2 layer 6, seed 0: tiny perturbations already cost most of what fp16 costs, so the count needs a floor. Bars: alive features a retrained SAE recovers at best match $r > 0.9$; dashed line: seed floor; right column: relative L_2 distortion.

The retraining test turns out to be very noisy. We took a 20M-token cache and changed one number in it by the smallest amount possible in fp32 (1 ulp, a unit in the last place). When we retrained, 99.98% of the features survived. In contrast, when we changed a millionth of all the numbers by that same smallest step, only 70.5% of the features remained the same (Fig. 1). Yet when we rounded every number in the cache to fp16, a change 10^8 times larger, we retain 60.6% of the features. So the count responds to almost any change in the bytes and only weakly to the size of the change (§3.1). It cannot be read as a level, only compared against controls built the same way.

A measurement can be precise and still answer the wrong question (12; 13); the frozen-SAE test is one, and the retraining test needs calibration first. Our two contributions:

1. **The ranking of codecs is reliable; the exact percentages are not** (§3, §4). The ordering holds across sites, thresholds and budgets (up to 2 pp ties, seed noise; TopK SAEs only), but the percentage moves with every setting we varied, so no single number transfers.
2. **A same-process control lets us compare compression against other reasonable noise, and that is enough** (§2, §4). Naïve int4 alters the features more than a seed change in every experiment we ran, and per-channel standardization brings it back above reseeding at the same bytes per token. A bf16 forward pass loses 1.8 to 3.0 pp more agreement than rotated int8 storage at our $1\times$ budget (three seeds, same direction, two of three outside the 2 pp tie band).

2 Calibrating the retraining test

The retraining test needs calibrating before it can be used (§3.2 turns to the frozen-SAE test), because SAE training is unstable across seeds (5; 6): a naïve fp32-vs-codec comparison could mix the codec's effect with seed noise. So we calibrate against the seed change by using it as a unit of measurement, building the bounds directly: retraining on identical bytes gives the upper bound, and retraining with a different seed gives the lower one.

Glossary of terms used in the paper. A channel is one of the d coordinates of the activation vector. Next we have the codecs, our compressed storage formats. rot-int$N$ standardizes each channel (by subtracting the mean and dividing by std), applies a fixed random rotation and quantizes to N bits. std-int$N$ skips the rotation. int$N$ is naïve per-token quantization (meaning one absmax scale per token). int$N$-oM keeps M outlier channels in fp16. A codec's ratio is the percentage of retained features divided by its site's floor from reseeding. A codec that is above this reseeding floor is considered viable. If two single-seed runs come within 2 percentage points (pp) we consider them tied as seed noise alone shifts every run's number by up to 1.1 pp (Table 1). Numbers in the text are seed 0, except for the reseeding floors which are three-pair means and the denominator of each ratio. Tables in the appendix add three-seed means written $\pm$ half the min–max range. Our $1\times$ budget is 33M token-visits (tokens seen in training, counting repeats).

Design. We use one fp32 master cache per site from a single forward pass on a model. Every codec store we make is a transcode of this cache, shard for shard. The loader then shuffles based on just the seed, so two SAEs trained with the same seed consume identical tokens in the same order and differ only in bytes. We trained each SAE with Adam at learning rate 3×10^{-4} for $8{,}000$ steps at batch $4{,}096$. A site is one model, layer and read-off point; agreement between two SAEs is the fraction of the reference SAE's alive features whose best match in the other reaches Pearson $r > 0.9$

on held-out fp32 tokens; we train TopK SAEs (the JumpReLU panel, SAEs with learned thresholds, is Table 5).

Ceiling. Retraining with the exact same seed and bytes reproduces the run exactly: $r = 1.0000$, a 100% agreement on Apple M5 (MPS) and on an A100 (Appendix D). Any same-seed difference therefore comes from the codec alone.

Floor. To find the lower bound, how much seed noise alone costs, we retrained an SAE with a different seed. On GPT-2 small, we only retain $23.0 \pm 0.1\%$ of features over three seed pairs at layer 6 (20M OpenWebText tokens; TopK SAEs, where the k=32 largest of d_{sae}=12288 latents fire per token). On Pythia-1.4B (10M Pile tokens, bf16 forward, d_{sae}=16384), we retain $15.7 \pm 0.1\%$ at layer 8 and $18.4 \pm 0.1\%$ at layer 16. These floors depend on the site, so every codec must be compared against its own site's floor.

When we reseeded only the initialization, we got an indistinguishable $23.3 \pm 0.8\%$ (Appendix D). Under Paulo & Belrose's criterion, our floor is close to their published value (their site differs so it is not exact) (5).

Which features we are considering. Every number about agreement is over alive features, those that fire on more than 0.1% of the evaluation tokens in the reference SAE, which ends up being around 40% of the dictionary at each site. Rarer features do not activate enough for a correlation to be significant. Increasing the threshold raises agreement and the reseeding floor and lowers rot-int4's ratio monotonically across the five cut values we swept (Table 8); ours is the loosest of the five, so a ratio should be quoted with its cut.

Sanity checks. We ran a few checks after every pipeline change: an identity check (a cache against itself returns exactly 1.0), a misalignment check (one token of misalignment collapses the correlation to 0.15–0.38) and a shuffle control on best-match luck (0.06 pp or less) (Appendix D).

3 What each test can and cannot tell us

Our two tests give very different answers for the same codec. Run the frozen-SAE test on fp16 activations and each feature's activations correlate with its own activations on fp32 at mean $r = 0.99997$, every feature above 0.9. Run the retraining test on those same fp16 bytes and only 60.6% of features are retained. Neither number can be taken at face value.

3.1 The retraining test

We first examine the effect of adding isotropic noise to the fp32 cache, at exact relative L_2 magnitudes from 10^{-2} down to 10^{-7}. Each level is stored in fp32 (nothing is quantized), passed its own identity check, and got a matched SAE retrained on the same tokens in the same order (Table 14 and Fig. 2, Appendix E). Across the four noise levels at or below fp16's distortion, agreement barely moves: 60.6% at fp16's own distortion level (by coincidence the same number fp16 gives), rising only to 66.0% three decades below, while the frozen SAE reads at least 0.999985 throughout; zero perturbation returns exactly 100%.

Dense noise cannot go below fp32's own precision, so for the remaining experiments we changed only a few values instead of all of them. When we changed 15,335 scalars (a millionth of the cache's 15 billion values) by one ulp each, 70.5% of features survived. When we moved a single scalar in the whole corpus by this amount, 99.98% survived. The only difference between these two experiments is how many values moved, and that difference alone costs 29.5 pp. Most of the features lost at $r > 0.9$ have shifted rather than disappeared: lowering the threshold to $r > 0.7$ raises agreement from 70.5 to 89.8% (Table 16).

Three checks (Appendix E) confirm that these numbers are real. Re-running the two headline perturbations with fresh randomness moves them by at most 1.1 pp. The full drop is already present at 77 touched values out of 15 billion ($1 < n < 77$ remains open), and every perturbation retains features far above the reseeding floor.

With near losslessness in values, retraining barely changes features (5.4 pp over three decades, against ±1.1 pp of seed noise). However, over the range real codecs occupy it can go from 60.6 to 15.6% (Fig. 2). Any slope of agreement against perturbation size depends on the range it is fitted over (Table 15), so we use the retraining test only comparatively, against the reseeding floor.

3.2 The frozen-SAE test

The frozen-SAE test asks the other question: with the same SAE, how big is the difference in feature activations between the original and the compressed activations? As expected, the codecs that are most precise replicate the feature activations best. However, this is functionally equivalent to measuring the L_2 distance from the original activations. The frozen reading orders codecs as retraining does: Spearman $\rho = 0.982$ over all 15 GPT-2 conditions and 1.000 at both Pythia sites (Table 21, Appendix G). But the plain L_2 distance between the caches, which needs no SAE at all, ranks at $\rho = 0.979$, and mean cosine ties the frozen SAE at 0.982 (Table 21). At $n = 15$ the three are indistinguishable (one exception in Appendix G).

But the order of the codecs does not tell us how replicated the results are. Using the frozen-SAE test's thresholded count (features whose frozen correlation exceeds 0.9), the fp16 cache gets 100.0% of features intact and int6 and int4-o16 keep 99.9% (Table 1). Retraining on those same bytes, however, returns 60.6, 38.0 and 37.6% respectively, and the same frozen reading of 0.9992 corresponds to keeping 54% of features at one site and 38% at another (Fig. 5, Table 4). No L_2 distance to the original cache can be calibrated around the perturbation that matters most, though, because a seed change changes no bytes, so every distance metric would return 0, yet only 23.0% of features survive a different seed.

We found this pattern of viable codecs passing and naïve int4 failing to be not SAE-specific: linear probes and a causal steering test (difference-in-means vectors injected into the residual stream), each scored against its own control, got the same results with two exceptions (PCA-256 keeps probe accuracy on one of two tasks while its directions changed; int4-o16 fails one probe floor; Appendix G).

4 Calibrated verdicts

Table 1 (Appendix A) has the full thirteen-codec GPT-2 result and Fig. 3 (Appendix F) plots the five codecs run at every site against each site's reseeding floor. Four conclusions follow, then the bf16 forward pass.

For viable codecs, most of the features lost would have been lost under reseeding anyway. Naïve int4 loses far more. Under rot-int8, only 0.13% of features that survive seed change (seed-stable) are lost in all three seeds. 0.90% are lost in rot-int4 while 24.2% are lost in naïve int4 (Table 19). Each viable codec retains 88.8–95.7% of the seed-stable features on GPT-2 and 82–89% on Pythia. Seed-unstable features are lost to a greater degree: 27–54% are retained on GPT-2 and 19–38% are retained on Pythia. This gap stays regardless of our threshold for stability (Table 20). Reconstruction error sees none of this: the fraction of variance unexplained (FVU) stays flat, 0.1355 at fp16 through 0.1361 at int4-o16, while agreement falls from 60.6 to 37.6% (Table 1).

Per-token quantization fails on outlier channels and standardizing each channel fixes this. A few channels have values much larger than the rest, so one scale per token means that we waste a lot of levels to reach the largest values. Naïve per-token int4 falls below the reseeding floor everywhere we tested: 15.6% against 23.0 on GPT-2, worse at the larger model and the deeper layer, 0.18× the floor at Pythia layer 8 and 0.05× at layer 16, keeping only 13.7 and 5.1% of even the stable features (Tables 2, 3). Standardizing each channel before quantizing repairs it at identical file size: 15.6 to 38.0% (±0.9 over three seeds), 1.65× the floor. Our rotated codec standardizes and then applies one fixed random rotation, so we try not rotating to separate the two effects. We find the rotation adds nothing we can detect (37.7 ± 0.3 against 38.0 ± 0.9% over three seeds, a tie, Table 18). Rotation is useful in inference quantization (10), but here standardization alone is enough (per-channel outlier handling is standard in LLM quantization (15; 16; 17); what is new is its measured cost in retained features). The standardized codecs tie or beat the outlier-channel (-oM) codecs at equal or fewer bytes, with one exception the other way: int4-o16 keeps more seed-stable features than rot-int4 (91.2 against 88.8%, Table 1). With zstd on top, std-int4 stores at 303.8 B/token against fp16's 1,419 (Table 11).

On a second model, we get the same ranking but the exact percentages are different. Pythia-1.4B is a model in a different family with $11\times$ the parameters, with a different dataset and tokenizer. We see the same codec order in terms of consistent features (the only inversions are among tied codecs) and the same 4-bit gap between standardized and naïve quantization (Fig. 3). Naïve int4 performs worse than reseeding at every site and threshold (Table 13). Changing both a codec and seed does not perform worse than the seed change: fp32 seed 0 compared to a rot-int8 or rot-int4 trained at seed 1 stays at the lower bound (24.5 and 23.5% respectively, against 23.0). Naïve int4 at seed 1 goes below at 14.2%. The exact percentages, however, changed with everything we varied (definition of stability, matching metric, budget, sparsity, firing-rate cut, threshold). The rot codecs' ratios span 1.06 to $3.55\times$ (Tables 8, 13).

The reseeding floor increases with training and this leads to codec viability performing differently. When we scale training to $3\times$ and $10\times$, the reseeding floor rises from 23.0 to 42.3%. More optimization makes seeds agree with each other, so every codec agreement ratio above 1 compresses toward 1 (fp16 from 2.63 to $1.20\times$, rot-int4 from 1.63 to $1.09\times$). However, naïve int4's agreement stays flat while the reseeding floor climbs, so it performs worse relatively, from 0.68 to $0.39\times$ the floor. More training does not recover the information the codec lost. These budget tiers ($1\times$, $3\times$, $10\times$) re-read one 20M-token cache, so part of the rising floor may also be convergence on a fixed corpus (Appendix F).

The effect of a bf16 forward pass. Now, we examine the effect of something that is not a codec. If we computed the cache with a bf16 forward pass as Gemma Scope (2) and most production pipelines do for speed, there is no compression and yet the frozen SAE trained on fp32 has a correlation of $r = 0.99819$. At our $1\times$ budget, the bf16 forward costs 1.8 to 3.0 pp more retrained agreement than rotated 8-bit storage at all three seeds (49.7–51.2 against 51.6–53.8%, Table 10; two of the three gaps exceed the tie band), but the forward pass itself runs $2.21\times$ faster. Now the scope is important as our budget is 33M token-visits. Production runs train each SAE on 4–16B tokens (2) and at $3\times$ our budget the gap falls inside the tie band (one seed, Appendix F).

Takeaways. If your forward pass is bf16, store the cache as bf16: that is bit-lossless, and the only remaining question is whether to compress to below 16 bits. However, storing an fp32 forward as bf16 is not free (55.6 against fp16's 60.6%, Table 1). Measured against 16-bit storage rather than fp32, rotated 4-bit is $4.0\times$ smaller before entropy coding, not the $7.9\times$ an fp32 baseline suggests. Standardize each channel before quantizing to 4 bits. Once that is done, 4 bits loses less agreement than a seed change at the budgets we measured, and the rotation on top of it adds nothing we can detect (Table 18), so skip it. Do not reduce dimension by PCA or random projection (7.8 and 2.8% on GPT-2, far below the floor, Table 1), and do not certify a training codec with a frozen SAE or with reconstruction error: they rank codecs without saying what you will keep. If you only run pre-trained SAEs over the cache, the frozen reading is the right test (fp16, int6 and int4-o16 all read at least 99.9% on it). Calibrating a site costs four training runs and two evaluation passes (Appendix C), and if you already plan multi-seed training (5) the floor's runs are essentially free.

5 Scope and limitations

The issue with storing caches at 4 bits applies only to activation caches, but the overall argument here is that it is important to carefully calibrate these changes. Retraining also tends to be more noisy than at first thought.

Limitations. Two models, residual-stream sites only, research-scale budgets, unnormalized activations. On JumpReLU the viable codecs sit within a few points of the floor (Table 5), too close for a graded verdict, so every graded claim is TopK only. The Pythia master is itself a bf16 forward, so its ceiling is trivial and its reference is already perturbed. All SAEs are evaluated on fp32 activations. The panel omits fp8 and block-scaled formats and compares codecs at equal token counts rather than equal bytes. Loader rates are from one laptop with CPU decode (Table 12), well below production disk rates. In short: the retraining test needs its own floor and ceiling; with them, standardized 4-bit caching is viable at our budgets and naïve per-token int4 is not.

References

[1] Z. He et al. Llama Scope: Extracting Millions of Features from Llama-3.1-8B with Sparse Autoencoders. arXiv:2410.20526, 2024.

[2] T. Lieberum et al. Gemma Scope: Open Sparse Autoencoders Everywhere All At Once on Gemma 2. arXiv:2408.05147, 2024.

[3] J. Bloom, C. Tigges, A. Duong, and D. Chanin. SAELens (software). `https://github.com/decoderesearch/SAELens`. Default activation dtype `float32`: `sae_lens/config.py`, accessed 2026-08-09.

[4] E. Duan. Perplexity Can Miss SAE Feature Damage Under Quantization. arXiv:2606.03002, 2026.

[5] G. Paulo and N. Belrose. Sparse Autoencoders Trained on the Same Data Learn Different Features. arXiv:2501.16615, 2025.

[6] G. Gerasimov, T. Rusalev, N. Balagansky, D. Laptev, V. Kurochkin, and D. Gavrilov. Unstable Features, Reproducible Subspaces: Understanding Seed Dependence in Sparse Autoencoders. arXiv:2606.12138, 2026.

[7] D. Chanin and A. Garriga-Alonso. Sparse but Wrong: Incorrect L0 Leads to Incorrect Features in Sparse Autoencoders. arXiv:2508.16560, 2025.

[8] K. Ayonrinde, M. T. Pearce, and L. Sharkey. Interpretability as Compression: Reconsidering SAE Explanations of Neural Activations with MDL-SAEs. arXiv:2410.11179, 2024.

[9] C. Summers and M. J. Dinneen. Nondeterminism and Instability in Neural Network Optimization. In ICML, PMLR 139:9913–9922, 2021. arXiv:2103.04514.

[10] S. Ashkboos et al. QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs. NeurIPS, 2024.

[11] J. Chen et al. ActNN: Reducing Training Memory Footprint via 2-Bit Activation Compressed Training. ICML, 2021.

[12] B. Li, T. Yu, K. J. L. Koa, and K.-W. Huang. The Proxy Presumption: From Semantic Embeddings to Valid Social Measures. arXiv:2605.07409, 2026.

[13] G. König, M. Pawelczyk, U. von Luxburg, and S. Bordt. Validity Threats for Foundation Model Research. arXiv:2606.05029, 2026.

[14] Z. Xiao, S. Zhang, V. Lai, and Q. V. Liao. Evaluating Evaluation Metrics: A Framework for Analyzing NLG Evaluation Metrics using Measurement Theory. EMNLP, 2023.

[15] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale. NeurIPS, 2022. arXiv:2208.07339.

[16] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. ICML, 2023. arXiv:2211.10438.

[17] Z. Liu, J. Yuan, H. Jin, S. Zhong, Z. Xu, V. Braverman, B. Chen, and X. Hu. KIVI: A Tuning-Free Asymmetric 2bit Quantization for KV Cache. ICML, 2024. arXiv:2402.02750.

A Full result tables

All appendix table cells are generated programmatically from the released evaluation artifacts (`scripts/appendix_tables.py`); none is hand-transcribed. Caption prose is checked against the same artifacts by `verify_paper_numbers.py`, a lint rather than a proof.

Table 1: Full GPT-2 codec panel (conventions as in §2: $\pm$ rows are three-seed means, bare cells seed 0, so fp16 appears as both 59.3 ± 1.1 and 60.6). FVU is flat across every viable codec (0.1355 to 0.1361) while agreement falls from 60.6 to 37.6%: reconstruction quality and feature identity are different things.

codec	B/tok	survived %	FVU	mean r	stable %	Hung. cos	frozen %
fp16	1536	59.3 ± 1.1	0.1355	0.866	95.7	0.817	100.0
bf16 storage	1536	55.6	0.1355	0.853	93.3	0.799	100.0
rot-int8	772	52.7 ± 1.1	0.1356	0.845	94.2	0.786	100.0
int8	772	46.6	0.1356	0.821	91.5	0.751	100.0
int8-o8	780	50.8	0.1355	0.837	93.7	0.773	100.0
rot-int6	580	45.9 ± 0.7	0.1355	0.823	92.1	0.752	100.0
int6	580	38.0	0.1359	0.790	89.8	0.696	99.9
int6-o8	590	44.2 ± 0.4	0.1357	0.813	91.6	0.740	100.0
rot-int4	388	37.7 ± 0.3	0.1359	0.789	88.8	0.697	100.0
int4-o16	412	37.6	0.1361	0.789	91.2	0.692	99.9
int4	388	16.0 ± 0.3	0.1968	0.664	58.9	0.461	36.7
pca256	512	7.8	0.3640	0.571	34.8	0.351	37.2
rp256	512	2.8	0.8626	0.485	16.3	0.265	4.6
seed floor	—	23.0 ± 0.1	0.1355	0.72	(def.)	0.577	—

Table 2: Pythia-1.4B layer 8, same construction as Table 1. Codec runs are seed 0 only, so gaps under ± 1.1 pp are unresolved; the bracketed intervals resample features, not seeds. fp16 is bit-lossless under the bf16 master, so its row is a same-seed ceiling.

codec	survived %	$\times$ floor	stable %
fp16 (bit-identical ceiling)	100.0	6.35 [6.00, 6.71]	100.0
rot-int8	43.0	2.73 [2.58, 2.87]	88.0
int8	32.9	2.09 [1.98, 2.20]	84.9
rot-int4	25.8	1.64 [1.55, 1.73]	81.6
int4	2.8	0.18 [0.15, 0.20]	13.7
seed floor	15.7 ± 0.1	1.00$\times$	(def.)

Table 3: Pythia-1.4B layer 16. Identical construction, interval convention and bit-identical fp16 ceiling reading to Table 2. Source: `runs_m4_L16/eval`.

codec	survived %	$\times$ floor	stable %
fp16 (bit-identical ceiling)	100.0	5.44 [5.17, 5.70]	100.0
rot-int8	38.4	2.09 [1.99, 2.18]	89.3
int8	27.7	1.51 [1.44, 1.57]	83.4
rot-int4	26.1	1.42 [1.35, 1.48]	83.0
int4	0.8	0.05 [0.03, 0.06]	5.1
seed floor	18.4 ± 0.1	1.00$\times$	(def.)

Table 4: Frozen-SAE mean r beside retrained agreement at three sites. Within a site the frozen SAE ranks the codecs as retraining does; across sites the same reading carries no level (rot-int8 reads within 0.00005 everywhere and retrains to 2.09–$2.73\times$ the local floor). Seed 0 throughout.

codec	GPT-2 r	retr.%	L8 r	retr.%	L16 r	retr.%
fp16	0.99997	60.6	1.00000	100.0	1.00000	100.0
rot-int8	0.99919	53.8	0.99922	43.0	0.99924	38.4
int8	0.99632	46.6	0.99488	32.9	0.98916	27.7
rot-int4	0.98536	37.7	0.98678	25.8	0.98696	26.1
int4	0.84005	15.6	0.56333	2.8	0.15633	0.8
seed floor	—	23.0	—	15.7	—	18.4

Table 5: JumpReLU panel (GPT-2, calibrated $\lambda = 8.68$). The viable codecs land within a few points of the floor (fp16 26.4, rot-int4 26.1, rot-int8 25.3 against 23.7 to 24.2), too close together for a graded verdict; these runs end with 34% of latents dead, and the naïve int4 run converged to $L_0 = 38.2$ against fp32's 18.6, so it is excluded as confounded.

codec	survived %	mean r
fp16	26.4	0.716
rot-int8	25.3	0.708
rot-int4	26.1	0.708
int4	13.7	0.623
seed floor	23.7–24.2	0.70

Table 6: Budget scaling on a fixed stable set ($N = 774$ features at every tier, binomial SE about 1.1 pp). fp16 is flat, the rotated codecs improve. fp16's count is 741 of 774 at all three tiers by coincidence: the three pairs are distinct runs whose all-alive agreement is 60.6, 49.2 and 50.6%. Budget was scaled by re-reading one 20M-token cache (up to 16 passes over the same tokens), so this isolates optimization from data diversity but not from convergence on a fixed corpus.

budget	stab. thresh	fp16 %	rot-int8 %	rot-int4 %	gap pp
$1\times$ (33M visits)	0.900	95.7	94.2	88.8	7.0
$3\times$ (98M)	0.936	95.7	94.7	93.5	2.2
$10\times$ (328M)	0.954	95.7	96.3	94.2	1.6

Table 7: Linear probes: accuracy change against the fp32-trained probe, and learned-direction cosine, each scored against a seven-split resample floor. PCA-256 keeps accuracy on sst2 but not on ag_news (its drop there is outside the floor) while its directions collapse to about 0.5 on both; int4-o16 fails the sst2 direction floor (0.778 against a floor minimum of 0.788) on a test the retraining test passes.

task	codec	Δacc	direction cos
sst2	fp16	+0.0011	1.000
sst2	rot-int8	+0.0000	1.000
sst2	rot-int4	+0.0011	0.949
sst2	int4-o16	-0.0080	0.778
sst2	int4	-0.0608	0.489
sst2	pca256	-0.0046	0.500
ag_news	fp16	-0.0001	1.000
ag_news	rot-int8	+0.0000	1.000
ag_news	rot-int4	+0.0004	0.968
ag_news	int4-o16	-0.0017	0.937
ag_news	int4	-0.0554	0.524
ag_news	pca256	-0.0162	0.547

Table 8: The alive cut, swept. Tightening it raises agreement and the floor together and lowers rot-int4's ratio monotonically (1.63 to 1.13× on GPT-2); our 0.001 cut is the loosest of the five. rot-int4 only, so this bounds a ratio, not an ordering; the floor here averages two seed-0 pairs, hence 23.1 against the three-pair 23.0.

	> 0.001	> 0.002	> 0.005	> 0.01	> 0.02
gpt2_l6					
n alive	4860	2927	1421	654	201
floor %	23.1	27.0	33.7	40.1	47.5
rot-int4 ×floor	1.63	1.50	1.38	1.28	1.13
pythia_l8					
n alive	6573	4053	1523	506	100
floor %	15.8	16.1	20.9	24.7	36.0
rot-int4 ×floor	1.63	1.62	1.43	1.34	1.14
pythia_l16					
n alive	7088	4510	1525	353	73
floor %	18.5	19.2	25.7	46.2	58.2
rot-int4 ×floor	1.41	1.37	1.30	1.22	1.06

Table 9: Sparsity moves the floor without touching a byte: over a 1.95× swing in the floor, rot-int8's ratio spreads 1.72–3.23× while its survival among seed-stable features spreads only 92.8–97.1%.

sparsity	seed floor %	rot-int8 %	×floor	stable-only %	n_{alive}	stable frac %
$k = 16$	28.6 ± 0.8	49.1	1.72×	92.8	2,912	21.0
$k = 32$	23.0 ± 0.1	53.8	2.34×	94.2	4,860	15.9
$k = 64$	14.7 ± 0.2	47.4	3.23×	97.1	8,269	10.1

Table 10: A treatment that is not a storage codec: computing the cache with a bf16 forward (stored in fp32) beside the storage codecs. Over three seeds the bf16 forward retains 49.7–51.2% against rot-int8's 51.6–53.8, disjoint but with $n = 3$ (a sign test cannot go below $p = 0.25$; within-seed deltas -2.7, -1.8, -3.0 pp). At 3× budget the comparison reads 44.4 against 45.3% (one seed each), inside the tie band. It runs 2.21× faster.

treatment	B/tok	rel-L2	frozen r	survived %	×floor	stable %	Hung. cos	FVU
fp16 storage	1536	0.00020	0.99997	60.6	2.63	95.7	0.817	0.1355
rot-int8 storage	772	0.00551	0.99919	53.8	2.34	94.2	0.786	0.1356
int8-o8 storage	780	0.00928	0.99868	50.8	2.21	93.7	0.773	0.1355
bf16 forward	**3072**	**0.00980**	**0.99819**	**51.2**	**2.22**	**94.2**	**0.775**	**0.1356**
int8 storage	772	0.02660	0.99632	46.6	2.02	91.5	0.751	0.1356
seed floor	—	—	—	**23.0**	**1.00**	(def.)	—	—
bf16 fwd, 3 seeds	—	—	—	**49.7–51.2**	—	—	—	—
rot-int8, 3 seeds	—	—	—	51.6–53.8	—	—	—	—

Table 11: Lossless zstd-3 over the quantized payloads, measured on one shard per codec. Rotated codecs barely compress further; naïve scaling leaves headroom that zstd recovers. Standardization without the rotation lands with the rotated codec (303.8 against 307.5 B/token).

codec	raw B/tok	zstd B/tok	ratio
fp16	1536	1419.0	1.08×
int8	772	510.9	1.51×
rot-int8	772	709.1	1.09×
int6	580	431.9	1.34×
rot-int6	580	557.0	1.04×
int4	388	146.4	2.65×
rot-int4	388	307.5	1.26×
std-int4	388	303.8	1.28×
int4-o16	412	309.1	1.33×

Table 12: Serial loader throughput per store (prefetch $= 0$, warm page cache, one Apple M5). The $1\times$ training run consumes $10{,}017$ tokens/s, so even the slowest store (int4-o16, 55k tokens/s) outruns training; the standardize-and-rotate decode path costs real time (rot-int8 reads at 64% of int8's rate). One-machine numbers. In a paired idle-machine measurement (two interleaved runs each), std-int4 decodes at 88 MB/s against rot-int4's 89; our std-int4 keeps the rotation matmul with an identity matrix, so this does not isolate the rotation's own cost.

codec	B/tok read	disk MB/s	decode MB/s	loader tok/s
fp32	3072	3094	10416	383k
fp16	1536	3085	4748	445k
int8	772	2743	754	319k
rot-int8	772	2917	242	205k
int6	580	2796	208	228k
rot-int6	580	2998	113	147k
int4	388	2677	185	270k
rot-int4	388	2543	86	162k
int4-o16	412	3037	25	55k
pca256	512	2603	402	335k

Table 13: Ratio to floor as the counting threshold moves. Levels change with the threshold; the ordering and the below-floor verdict for naïve int4 do not.

	$r > 0.7$	$r > 0.8$	$r > 0.85$	$r > 0.9$	$r > 0.95$
GPT-2 L6					
floor (survival)	59.4	42.4	32.9	23.0	10.9
fp16	1.45×	1.83×	2.15×	2.63×	3.78×
rot-int8	1.39×	1.71×	1.97×	2.34×	3.19×
rot-int4	1.22×	1.39×	1.50×	1.63×	1.90×
int4	0.83×	0.79×	0.75×	0.68×	0.61×
Pythia L8					
floor (survival)	56.3	36.5	26.3	15.7	6.0
fp16	1.78×	2.74×	3.80×	6.35×	16.80×
rot-int8	1.43×	1.86×	2.19×	2.73×	3.55×
rot-int4	1.20×	1.37×	1.49×	1.64×	1.72×
int4	0.43×	0.33×	0.25×	0.18×	0.07×
Pythia L16					
floor (survival)	56.9	38.9	28.9	18.4	7.7
fp16	1.76×	2.57×	3.46×	5.44×	12.98×
rot-int8	1.34×	1.60×	1.81×	2.09×	2.50×
rot-int4	1.14×	1.25×	1.36×	1.42×	1.55×
int4	0.17×	0.10×	0.08×	0.05×	0.02×

Rows are a four-codec slice of a sweep covering 13 codecs at GPT-2 and these four at each Pythia site. Ordering is invariant across it at both Pythia sites. On GPT-2 the only inversions are among rot-int4, int4-o16 and int6 (the last two not shown here), whose largest inverting gap is 0.45 pp, inside the tie band. Naïve int4 is below floor at every threshold at every site.

Table 14: The noise levels and sparse changes (seed 0): isotropic noise at an exact relative L_2 and one-ulp changes to a fixed count of values, all stored in fp32 on both sides. The count response is a step: everything from $n = 77$ to 15,335 is flat. The fp16-scale noise and the millionth-of-values change were re-run twice with fresh randomness (60.6/60.4/60.8 and 70.5/69.7/71.6%); the table prints the original runs.

perturbation	rel-L2	frozen r	survived %	pp/decade
zero (bit-exact)	0	1.000000	**100.0**	—
noise 10^{-2}	1.00e-02	0.998551	51.2	+7.82
noise 10^{-3}	1.00e-03	0.999847	59.0	+2.38
fp16 (codec)	2.03e-04	0.999969	60.6	+0.07
noise 10^{-4}	1.00e-04	0.999985	60.6	+1.19
noise 10^{-5}	1.00e-05	0.999999	61.8	+1.69
noise 10^{-6}	1.00e-06	1.000000	63.5	+2.50
noise 10^{-7}	1.01e-07	1.000000	66.0	—
1 ulp on 10^{-6} of entries ($\sim$15k)	1.59e-12	1.000000	70.5	—
1 ulp on 1,526 scalars	1.70e-13	1.000000	68.6	—
1 ulp on 153 scalars	9.57e-15	1.000000	69.3	—
1 ulp on 77 scalars	5.61e-15	1.000000	69.9	—
1 ulp on **one** scalar	6.24e-15	1.000000	99.98	—
seed floor	—	—	23.0	—

Table 15: The noise levels and codecs on the seed-stable stratum ($n = 774$) beside all alive features. The response is flatter still on the stable stratum (0.35 pp per decade against the codecs' 2.6), and the ratio of the two rates depends on the window, so we quote no single asymmetry figure.

arm	rel-L2	all alive %	seed-stable %
noise 10^{-7}	1.01e-07	66.0	96.4
noise 10^{-6}	1.00e-06	63.5	97.0
noise 10^{-5}	1.00e-05	61.8	96.1
noise 10^{-4}	1.00e-04	60.6	95.3
noise 10^{-3}	1.00e-03	59.0	95.9
noise 10^{-2}	1.00e-02	51.2	93.4
fp16	2.03e-04	60.6	95.7
rot-int8	5.51e-03	53.8	94.2
rot-int4	9.99e-02	37.7	88.8
naïve int4	4.55e-01	15.6	58.9

Table 16: How far the two smallest perturbations sit above the seed floor at each counting threshold. The margin is positive everywhere (11–43 pp dense, 12–47 sparse) but its size is threshold-dependent.

criterion	10^{-7} %	floor %	margin (pp)	below ceiling (pp)
10^{-7} **dense**				
$r > 0.50$	95.6	83.8	+11.8	4.4
$r > 0.70$	88.3	59.4	+28.9	11.7
$r > 0.85$	76.2	32.9	+43.3	23.8
$r > 0.90$ (ours)	66.0	23.0	+43.0	34.0
$r > 0.95$	46.3	10.9	+35.4	53.7
$r > 0.99$	12.2	0.8	+11.4	87.8
sparse, 1 ulp on 10^{-6}				
$r > 0.50$	96.1	83.8	+12.3	3.9
$r > 0.70$	89.8	59.4	+30.4	10.2
$r > 0.85$	79.3	32.9	+46.4	20.7
$r > 0.90$ (ours)	70.5	23.0	+47.4	29.5
$r > 0.95$	52.1	10.9	+41.3	47.9
$r > 0.99$	15.1	0.8	+14.2	84.9

Table 17: The causal test: a difference-in-means steering vector injected into the residual stream, scored against the variation between two disjoint halves of the same data (a floor that runs $2\times$ a like-for-like referent, conservative only for a fail). Naïve int4 fails every resample split on both tasks; PCA-256 passes on average but fails 58 of 400 splits on ag_news; task-matched prompts and a strength sweep leave every verdict unchanged.

codec	direction cos	causal cos	effect/floor
ag_news (floor 0.9995 / 0.9998)			
fp16	1.0000	1.0000	0.002
rot-int8	1.0000	1.0000	0.007
rot-int4	1.0000	1.0000	0.053
pca256	0.9998	0.9999	0.276
int4	0.7901	0.8294	13.713 **FAIL**
sst2 (floor 0.9851 / 0.9877)			
fp16	1.0000	1.0000	0.001
rot-int8	1.0000	1.0000	0.003
rot-int4	0.9999	1.0000	0.056
pca256	0.9991	0.9998	0.245
int4	0.8763	0.9702	2.354 **FAIL**

Table 18: Which step repairs 4-bit. std-int4 is byte-identical to rot-int4 with the rotation set to the identity, so the two differ by exactly the rotation: standardization alone gives $38.0 \pm 0.9\%$, adding the rotation $37.7 \pm 0.3\%$, inside the tie band. All three arms are three-seed means over same-seed pairs. The ablation keeps the rotation matmul, so it isolates the rotation's effect, not its decode cost.

arm	B/tok	rel-L_2	frozen r	survived %	$\times$floor
naïve int4	388	0.4553	0.84005	16.0 ± 0.3	$0.69\times$
std-int4 (standardize only)	388	0.1033	0.98466	38.0 ± 0.9	$1.65\times$
rot-int4 (standardize + rotate)	388	0.0999	0.98536	37.7 ± 0.3	$1.64\times$

Table 19: Is codec damage a shared bias or seed-like noise? On the fp32 consensus set ($n = 774$), losses are enriched above independence for every codec, so a shared component exists, but under rot-int8 only 0.13% of consensus features are lost in all three seeds against 24.2% under naïve int4. Recomputed from released pair files; no training.

codec	lost/seed %	P(lost s1 | lost s0) %	indep. %	lost all 3 %	indep. %
rot-int8	6.1	17.8	6.3	0.13	0.023
rot-int4	11.1	25.3	12.0	0.90	0.134
int4	41.0	75.5	42.9	24.16	6.876

Table 20: Survival among seed-stable features under four definitions of stability: activation-space matching at three thresholds, and weight-space Hungarian matching on encoder+decoder cosine. The gradient (viable codecs high, naïve int4 low) holds under every definition, while the level for the viable codecs moves from 62.9 to 98.8%, so a stratified percentage is only interpretable with its definition attached. The last column is survival among the unstable features at $r > 0.9$. On Pythia the viable codecs keep 18.6 to 38.3% of the unstable features (`runs_m4_L8`, `runs_m4_L16` stability files). GPT-2 L6, seed 0. Sources: `stability_fp32_s0_t0p*.json`, `stability_hungarian.json`.

codec	$r > 0.8$ %	$r > 0.9$ %	$r > 0.95$ %	Hungarian %	unstable, $r > 0.9$ %
fp16	90.1	95.7	97.3	80.7	54.0
rot-int8	86.0	94.2	98.8	76.4	46.2
rot-int4	73.8	88.8	95.5	62.9	28.0
int4	41.3	58.9	73.6	26.9	7.4

Table 21: Spearman rank agreement of three cheap readings with retrained agreement over every released condition per site. The 15 GPT-2 conditions are the 13 storage codecs of Table 1 plus the bf16 forward and the std-int4 ablation. None of the three readings needs a trained SAE, and none ranks better than the others.

site	n	frozen r ρ	rel-L_2 ρ	cosine ρ
GPT-2 L6	15	0.982	0.979	0.982
Pythia L8	5	1.000	1.000	1.000
Pythia L16	5	1.000	1.000	1.000

Table 22: Predicting per-feature survival from fp32 statistics (5-fold CV AUC). Mean activation is the strongest single predictor, unlike the peak activation reported for weight quantization.

codec	CV AUC	mean act	peak act	rate
rot-int6	0.836	0.758	0.713	0.552
rot-int4	0.844	0.755	0.711	0.570
int4-o16	0.831	0.749	0.711	0.564
int6	0.838	0.747	0.709	0.573

B Matched-pair activating contexts

Side-by-side top-activating contexts for sampled matched pairs (fp32 vs rot-int4, and the fp32 cross-seed control) are released with the artifact (`docs/appendix-semantic-pairs-*.md`). Illustrative excerpts: a survived pair ($r = 0.98$) fires on identical "if you want to" contexts on both sides with 90% top-10 position overlap; a damaged pair ($r = 0.47$) matches a precise "rise/increase of" feature to a coarser "of / fall of" feature: related, visibly degraded. The seed-pair control shows the same qualitative signature: codec damage is seed-shaped. Bands (survived $r > 0.9$, damaged $r < 0.5$) follow (4). Survived pairs share 64–85% of their top-50 activating contexts while damaged pairs share 20–27%, and the two bands separate in all seven comparisons we measured.

C The protocol as a recipe

To calibrate any treatment T of SAE training data on a new stack:

1. **Fix the substrate.** One master cache per site from a single model forward; derive every treated dataset from it shard-for-shard; derive the loader's shuffle from the seed alone. Two runs at one seed then consume identical tokens in identical order and differ only in the treatment.
2. **Measure the ceiling** (1 extra run). Retrain at the same seed on the same bytes. Bit-reproducible stacks return 1.0 and any same-seed deviation is pure treatment effect; otherwise the measured ceiling is the upper bound every later number is read against.
3. **Measure the floor** (2 extra runs). Retrain at three or more seeds on the untreated data and report the mean with the min–max range. This is the unit everything else is quoted in. Two further runs that reseed only the initialization bound how much the data order contributes; on our site the two floors are indistinguishable.
4. **Run the identity and misalignment checks** (2 eval passes). Compare a dataset with itself through the entire pipeline (it must return exactly 1), and then against itself misaligned by one token: it must collapse. The first decodes one store twice, so it catches nondeterminism but not a fault symmetric across both sides; the second is what catches a metric too blunt to see drift, and a pipeline that agrees regardless. Rerun both after every pipeline change.
5. **Measure the saturation point** (1 extra run). Retrain on the untreated data perturbed by one ulp on a millionth of its scalars, a departure from bit-identity eight orders of magnitude below fp16. On our site that alone costs 29.5 pp of agreement. Without this measurement a practitioner reads fp16's 39.4 pp gap as codec damage when three quarters of it is the price of retraining on anything but the identical bytes. Read it as a reference point that other numbers are placed against, not as a term to subtract out, since drift and codec damage need not add: our own locality control moves it 2.6 pp at matched count, and we ran it once at one site.
6. **Stratify, then re-stratify.** Report T separately on features reproduced across the floor's seeds, and re-derive that split under a second, independent stability definition (we use weight-space Hungarian matching against activation-space matching). Levels move between definitions even when rankings do not, so quote the definition and threshold with every percentage.
7. **Report the ratio as a local unit.** Absolute agreement does not transfer across sites, and the ratio itself moves with budget, sparsity and the firing-rate threshold (§4); a number is interpretable only with all of them attached. What travels is the ordering within one test; across tests even that can reverse (§4).

Steps 2–5 are the whole cost of the calibration: four training runs and two evaluation passes, independent of how many treatments are then compared against them.

D Pipeline validation in full

Running a cache against itself through the whole pipeline returns exactly 1.0: a mean r of 1.0000000 over 500k tokens, on raw channels at all three sites and through a frozen SAE at GPT-2 and Pythia layer 8. We then misalign one side by a single token, and the correlation collapses to 0.15–0.38 across the sites and codecs we ran. We rerun both controls after every change we make to the

pipeline. The identity check (our null) decodes the same store twice, so it catches nondeterminism but not issues that affect both sides equally. On an A100 the same-seed ceiling also returns exactly 100%, the floor pair reads 22.8% against our 23.2 for that pair, and rot-int4 reads 37.1 against 37.7% (`runs_cuda_modal/`). This is why we also run the misalignment check (our canary): if our pipeline were broken in a way that returns high agreement no matter what, it would still pass the identity check, but it would fail the misalignment one. We also bound one further source of inflation. Since each feature searches 12,288 candidates for its best match, some matches could just be luck. When we shuffle the candidates, every match dies (nothing comes near 0.9); a stricter shuffle that preserves co-activation patterns inflates agreement by at most 0.06 pp; and 96–99.9% of matches are one-to-one for the viable codecs and the seed pair (90.9% for naïve int4, 87.1% for PCA-256; released artifacts).

There are also nuances about what the floor means. Our seed impacts both the initialization and the data order, which is a wider change than the same-data floor of (5). When we reseeded only the initialization and kept data order fixed, we got $23.3 \pm 0.8\%$ across three pairs, indistinguishable from the full floor. Using Paulo & Belrose's weight-space metric we get 30.6% above cosine 0.7, near their published $\sim$30% (their site and criterion differ). Other work already implicitly compares against this floor when correcting for seed instability (5) and sparsity misselection (7). The floor moves with sparsity and budget (§4), so it is worth measuring at the configuration you actually use.

Codec details. int$N$ quantizes each token vector symmetrically to N bits with one fp32 absolute-maximum scale per token. int$N$-oM first removes the M channels with the largest mean absolute activation on a fitting sample and stores them in fp16. std-int$N$ standardizes each channel with a mean and standard deviation fitted on a sample of the cache, then quantizes per token as int$N$ does; rot-int$N$ additionally applies one fixed orthogonal matrix drawn by QR-factorizing a seeded Gaussian matrix before quantizing. Decoding inverts every step, so the SAE always trains on fp32 reconstructions. Pythia uses the same TopK k=32; correlations use 500k held-out tokens at every site.

E Robustness of the perturbation experiments

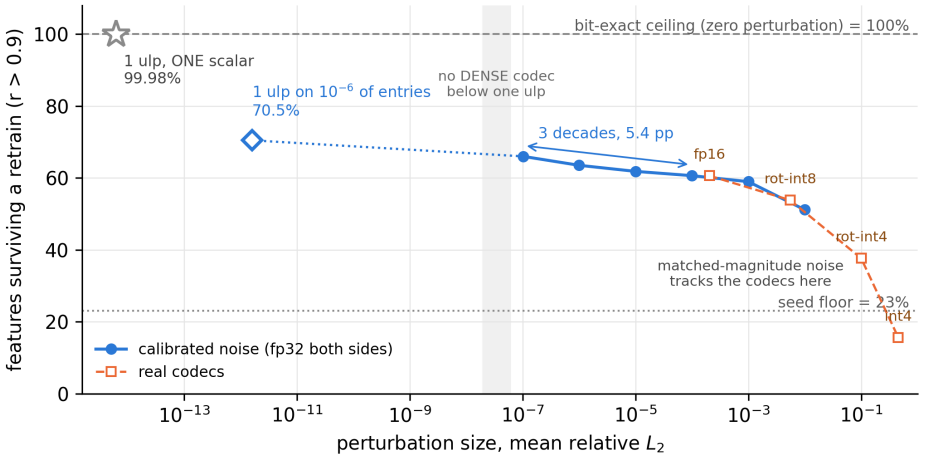

Figure 2: Retrained agreement against perturbation size. Blue: calibrated isotropic noise, fp32 on both sides; orange: real codecs; star and diamond: one ulp on one scalar and on a millionth of them; the shaded band is unreachable by any dense fp32 codec, since noise cannot go below fp32's own precision. Retraining barely responds to tiny perturbations but separates the real codecs clearly. GPT-2 L6, seed 0; the fp16-scale noise and the millionth-of-values change were each run three times and are plotted at their original run.

How solid are the headline numbers of §3.1? We regenerated the noise at the fp16-scale setting and the value placements at the millionth-of-values setting, twice each, and got 60.6 ± 0.2 and $70.6 \pm 1.0\%$ across the three repetitions, so neither number is an accident of one random draw.

We also ran an experiment that packs the same count of changes into one shard of the cache (pre-registered before it ran); it lands worse, not better (67.9%, outside the repetition spread), so where a perturbation sits matters even at fixed count. The three experiments at $n = 77$, 153 and 1,526 changed values all land between 68.6 and 70.5%, flat across two decades of count, and each was run once. The trained weights agree with these counts: the one-scalar change moves them by 0.056 relative Frobenius against a reseed's 1.409 (every alive latent of the one-scalar run still best-matches itself, so no permutation enters on that side). Throughout the paper, counts at $r > 0.9$ are the number a practitioner needs; the continuous mean-r readings are in Table 14. The above/below-floor verdict of §4 also survives a held-out-seed stratified floor: rot-int4 sits at 1.15–1.20× that floor at all three sites and naïve int4 at 0.06–0.71× (`stable_floor.json`). A codec plus a reseed lands at the floor for the viable codecs (fp32 seed 0 against rot-int8 and rot-int4 trained at seed 1: 24.5 and 23.5%) and below it for naïve int4 (14.2%), so the viable codecs add no drift beyond a reseed. Read against the level where retraining saturates (66–70%) rather than against 100, the standardization repair is partial: std-int4 sits about a third of the way up from the floor and rot-int8 about two thirds.

None of this is reseeding in disguise: both of the smallest perturbations sit far above the seed floor at every counting threshold, by 11–43 pp for the smallest dense noise and 12–47 for the millionth-of-values change (Table 16). Nor is it the weight-side story. One altered weight bit can destabilize training as much as a seed change (9), but one altered data scalar costs 0.02 pp and 15,335 of them cost 29.5, still far short of a reseed. On the data side, a perturbation has to be spread out to matter, not merely present.

One more objection deserves its own check: seed noise averages out across seeds, but a codec's damage might be the same features in every seed, a shared bias that multi-seed protocols cannot average away. Table 19 tests this on the fp32 consensus set. Losses are enriched above independence for every codec, so a shared component exists, but under rot-int8 it amounts to 0.13% of the consensus features lost in all three seeds, against 24.2% under naïve int4: a multi-seed protocol recovers almost everything under a viable codec and cannot under a failed one.

F Budget scaling in full

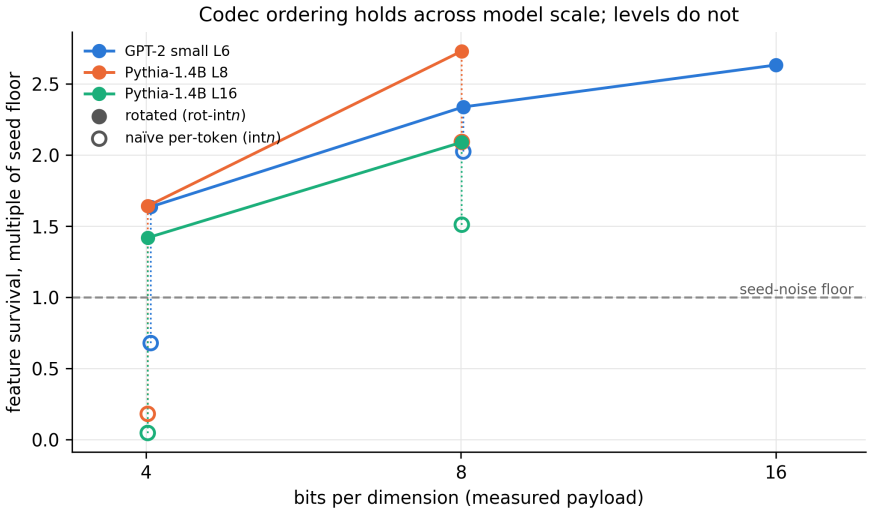

Figure 3: Agreement as a multiple of each site's own seed floor, against measured bits per dimension, 1× budget. Filled markers: rotated codecs; hollow: naïve per-token at the same rate. The dotted riser is the effect of standardization at identical bytes, and at 4 bits it spans the floor at every site. Read the ordering, not the levels: at 10× budget rot-int4's ratio is 1.09×. Pythia's fp16 is its own bf16 master, so it is omitted there.

When we scale training to 3× and 10× (98M and 328M token-visits, three fp32 seeds per tier), the floor rises from 23.0 to 33.3 to 42.3%. Every viable codec closes on the rising floor: fp16 reads 49.2 and 50.6% at 3× and 10× (1.47 and 1.20× the floor) and rot-int4 46.3% at 10× (1.09×). Held at

a fixed stable population, fp16 is flat while the rotated codecs improve (Fig. 4), so the statement we can defend is that the 16-to-4-bit gap does not widen with budget, not that it provably shrinks. Naïve int4's agreement stays flat (15.6/16.3/16.5%) while the floor climbs, so it falls from 0.68 to 0.49 to 0.39× the floor. Table 6 carries the per-tier numbers and two caveats: the stability threshold rises with budget, and budget was scaled by re-reading one cache, which isolates optimization from data diversity. At 3× the bf16 forward reads 44.4 against rot-int8's 45.3% (one seed each), a 0.91 pp gap inside the tie band. The bf16 forward's within-seed feature tests are effect-size evidence only, since features are dependent (Table 10), and at matched distortion that treatment is indistinguishable from a storage codec that stores four times fewer bytes (Table 10).

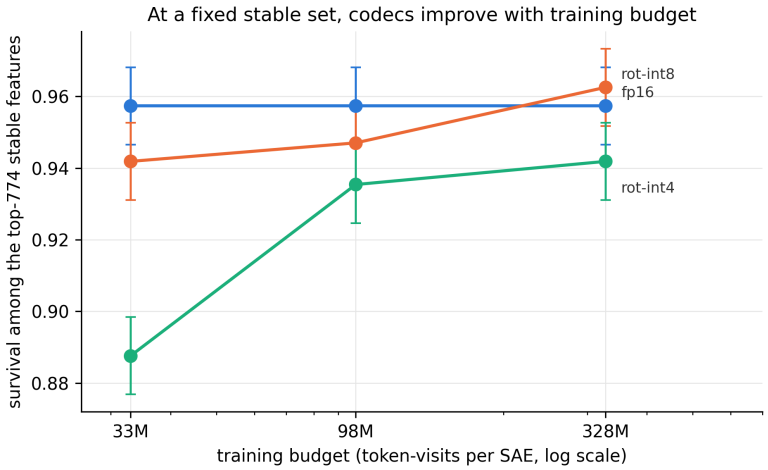

Figure 4: Agreement on a fixed-size stable set ($N = 774$, the 1× tier's stable-feature count, taking the top N by cross-seed stability at each budget) against training budget; error bars are binomial standard errors. fp16 is flat; the rotated codecs improve, so the gap does not widen.

G The other tests

Linear probes, scored against their own resample floor, pass every viable codec and fail naïve int4 (Table 7). The exceptions cut both ways: PCA-256 keeps probe accuracy on sst2 (on ag_news its accuracy drops outside the floor) while its learned directions collapse on both (0.50/0.55 cosine), which an accuracy-only check would not show, and int4-o16 fails one probe floor that the retraining test passes, so no single test is uniformly the strictest. A causal steering test, scored against the variation between two disjoint halves of the same data, agrees where it matters: naïve int4 fails both tasks and every viable codec passes (Table 17). Fig. 5 shows the cross-site picture for the frozen SAE itself. The frozen SAE does win in one place, across codec families and only on its continuous reading: PCA-256 (projection onto the top 256 principal components) is the smaller perturbation by both magnitude measures yet halves naïve int4's agreement (7.8 against 15.6%). Frozen mean r orders that pair correctly (0.810 against 0.840; int4's value is the GPT-2 column of Table 4); the thresholded reading inverts it (37.2 against 36.7%, the frozen column of Table 1).

One stratification runs the other way: splitting naïve int4's single-run loss by the fp32 SAE's feature statistics, the top peak-activation quartile reaches 1.04× that stratum's own floor while every other stratum stays below it (`stratify_singlerun.json`); every whole-SAE comparison in the paper is below the floor.

H Extended related work

Duan (4) measures SAE feature survival under weight quantization through a frozen SAE; we adopt its banding and show the frozen protocol cannot be calibrated for the data question. Paulo & Belrose (5) and follow-ups (6) establish seed instability; we turn it into the calibration a compression study needs. MDL-SAEs (8) also frame SAEs as lossy compression but price explanations; our

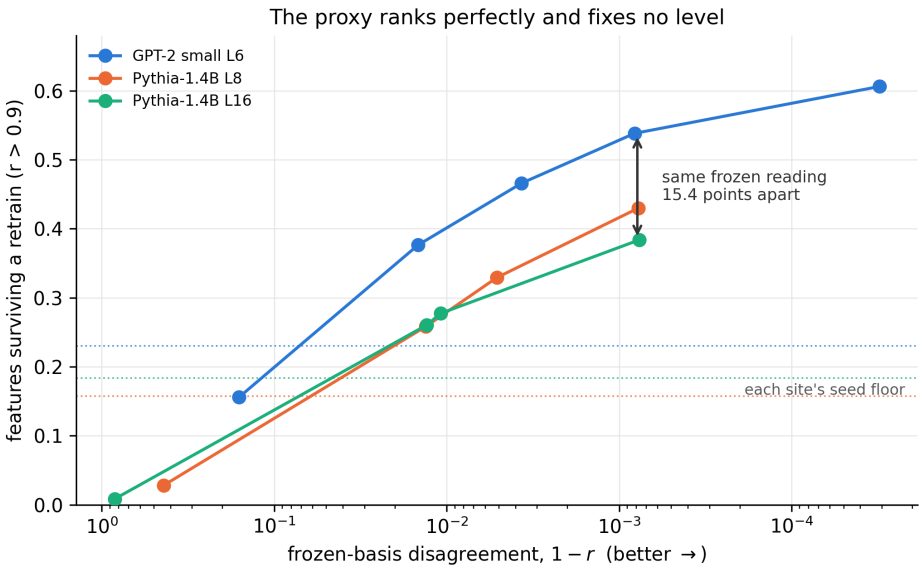

Figure 5: The frozen-SAE reading against the retraining it stands in for, at three sites; the x axis is frozen disagreement $1 - r$, log scale, improving rightward. Every series is monotone (the frozen SAE ranks codecs correctly), but the series are offset from each other (the same reading corresponds to different retained percentages at different sites); the viable codecs crowd into a span a linear axis would collapse to a point. Dotted lines are each site's seed floor. Pythia's fp16 point is omitted: the bf16 master makes it bit-lossless.

object is the training corpus. Rotation-based outlier suppression comes from inference quantization (10); activation compression for training (11) targets in-step memory, not a persistent dataset. The measurement-validity programme (12; 13; 14) supplies our framing. The outlier-channel problem we hit at 4 bits, and the per-channel scaling that fixes it, are well established in LLM activation and KV-cache quantization (15; 16; 17); we do not claim the mechanism, only its measured price for SAE training.

I Reproduction

Every number regenerates from scratch on one laptop via the restartable drivers: `run_m1.sh` (main grid), `run_m1_followup.sh` (rotation codecs), `run_m4.sh` (scale check), `run_arch.sh` (architecture panel), `run_m6.sh` (sparsity×seed sweep), `run_m2.sh` (probes), `run_controls_a.sh` and `run_controls_b.sh` (controls, including the forward-precision seeds and the four sub-fp16 noise levels), `run_noise_overlap.sh` (the two noise levels inside the codec range), `run_sparse_rung.sh` (the millionth-of-values change), `run_redraws.sh` (the repetitions and the initialization-only floor), `run_nbracket.sh` (the count-bracket experiments), `run_steering.sh` (the causal steering test), `run_budget_scaling.sh` and `run_int4_budget.sh` (budget tiers), `run_fwdprec.sh`, `run_ksweep_codec.sh`, `run_pythia_frozen.sh` and `run_seeds.sh`, plus `scripts/figures.py`, `scripts/appendix_tables.py`, `scripts/budget_fixedset.py` and `scripts/paired_feature_tests.py`; `scripts/modal_cuda_ceiling.py` reruns the CUDA replication on a cloud A100. A SAELens drop-in adapter (`acc.saelens_adapter`, zero added dependency) feeds SAELens's trainer directly from a compressed store. Compute, summed from the recorded per-run wall-clock in the released artifacts: 178.9 GPU-hours of SAE training across 104 recorded runs, on one Apple M5. Activation caching adds 2.8 hours that are recorded (both Pythia sites); the GPT-2 cache build is not in any released manifest, and transcoding, evaluation and probes are not separately timed. The Pythia misalignment-check values are recorded in `runs_m4.log` rather than in the eval JSON set, and the frozen-SAE identity check was run at GPT-2 and Pythia layer 8 only. Peak disk was of order 300 GB.

Use of AI assistance. We used an AI assistant (Claude, Anthropic) throughout this project for code implementation, for executing pre-registered experiment scripts, and for drafting and editing text, including internal adversarial reviews used during revision. No LLM is part of the method itself: the subject models (GPT-2 and Pythia-1.4B) are the objects of measurement, and no LLM-generated content enters any experimental result. Every number in the paper regenerates from the released artifacts via the included scripts, table cells are generated programmatically rather than transcribed, `verify_paper_numbers.py` checks every main-text number against those artifacts, and figures are generated by `scripts/figures.py` from the same JSONs. All citations were verified against the cited papers. We reviewed and take responsibility for all content.