CHANNEL SELECTION FOR MIXTURE-OF-EXPERTS DECODING NEEDS NO CALIBRATION DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-experts (MoE) language models decode slowly because every generated token must read its experts' weights from memory. Yet for a given token, a small share of each expert's channels carries most of its output. Methods that skip the other channels must choose which to compute before reading them, and the strongest tune thresholds on calibration data. We show that these data are unnecessary: replacing the calibrated thresholds of the strongest such method, Prox, by a fixed number of channels per token (top-k) changes its divergence from the full model by at most a quarter, whereas changing how channels are ranked changes it more than sixfold. Thresholds also make the cost depend on the input: on Gemma 4, Prox's thresholds read fewer bytes on reasoning than on their calibration data and collapse into repetition, which top-k selection with the same score avoids at the same reads. We therefore build a reader that ranks channels from the weights alone with a compact ternary sketch (7.8% of the expert bytes), computes the top-ranked channels exactly and recovers the rest with a low-rank map of the weights. Across four MoE models and nine baselines at equal memory reads, it is closer to the full model than Prox from 18% of expert reads up on OLMoE and Gemma 4 and level on the Qwen models, reads the same bytes on every input, and speeds up the MoE block by up to 1.9× over optimized vLLM. Channel selection can thus be applied to a new checkpoint from its weights alone.

1 INTRODUCTION

Generating text with a large language model is slow mostly because of memory traffic: at the small batch sizes of interactive use, each decoding step spends its time reading weights rather than computing with them. Mixture-of-experts (MoE) models (Shazeer et al., 2017; Fedus et al., 2022) send each token to a few experts, but those routed experts still dominate the bytes read per token. Much of this reading is wasted. An expert is a gated MLP whose internal *channels* (one gate row, one up row and one down column each) contribute very unequally to a given token, and a small share of them carries most of the expert's output. Reading only those channels would make decoding faster, but the choice has a catch: it must be made before the channels are read.

Existing methods make this choice with a cheap score and a threshold. Some score channels by the gate or up projection itself (Lee et al., 2024; He et al., 2024; Cheon & Kang, 2025; Park et al., 2026), which already reads a third of the expert; some by the input's magnitudes (Liu et al., 2025a; Chen et al., 2026b; Zhang et al., 2025), which lose quality quickly when few channels are kept; the strongest, Prox (Liu et al., 2026), scores them with a resident INT4 copy of the gate and up weights. All of them, at their best, tune their thresholds or per-layer budgets on calibration data, because how many channels clear a threshold depends on the activations each layer sees. Calibration is treated as the price of quality. It also ties both quality and memory cost to how closely new inputs resemble the calibration data.

We find that this price is not needed. Replacing Prox's two calibrated thresholds by top-k selections, a fixed number of inputs and channels per token, changes the KL divergence of its next-token distribution from that of the unmodified (*stock*) model by -17 to $+24\%$ across four models, whereas changing how channels are scored changes it up to sixfold (Section 6.1). Quality follows the ranking:

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.

across selectors and budgets, the divergence falls with the share of the layer's output that the kept channels carry. Thresholds also carry a hidden risk. Their reads follow the input, and on Gemma 4 Prox's thresholds, tuned to read 17% of the expert bytes on chat, read 13% on GPQA-Diamond reasoning, where the model falls into repetitive loops and scores 11.6 against the full model's 80.3. At the same 13% of reads, a top-k selection with Prox's own score keeps 62.1: the collapse is the threshold's, not the score's. Calibration, in short, buys little quality and makes both cost and quality depend on the input.

If the ranking decides, the goal becomes the best ranking that can be computed cheaply from the weights alone. Our reader (Section 4) scores every candidate channel of a token by its expected contribution to the layer output, estimated from a ternary sketch of the gate and up weights that holds 7.8% of the expert bytes, so that one top-F with F fixed by the read budget serves every token and every layer. It computes the chosen channels exactly and recovers the omitted ones with a low-rank map of the down projections; the model, the router and its single weight layout are unchanged, and a new checkpoint needs only its weights. Compared with nine published methods on four MoE models at equal *reads*, the bytes of expert weights read per token as a share of what the stock model reads, priced in the 32-byte sectors in which GPU memory moves data, the reader is closer to the stock model than Prox from 18% of expert reads up on OLMoE and Gemma 4 and level on the Qwen models, with half of Prox's resident state; at 13% Prox's ranking remains competitive. Because a top-F reads the same bytes on every input, its memory cost is known in advance, and the MoE block runs up to 1.9× faster than optimized vLLM. Our contributions are:

- the finding that channel selection for frozen MoE decoding needs no calibration data: top-k in place of calibrated thresholds loses little fidelity, quality follows the ranking, and calibrated budgets drift on new inputs into repetition failures (Section 6.1);
- a reader that ranks channels from the weights alone, computes the chosen ones exactly and recovers the rest, at a read cost fixed in advance (Section 4);
- an evaluation against nine baselines on four MoE models at equal sector-priced reads, under each baseline's own protocol, with speed measured against optimized vLLM (Section 6).

2 Related work

Methods that skip weights during decoding differ in what they score and how they decide how much to keep.

Activation sparsity. Trained predictors of active ReLU neurons came first (Liu et al., 2023; Song et al., 2024; Akhauri et al., 2024; Shin et al., 2025). For SwiGLU models, CATS (Lee et al., 2024), CHESS (He et al., 2024) and M-COUNTDOWN (Cheon & Kang, 2025) threshold after computing a full projection; TEAL (Liu et al., 2025a), WINA (Chen et al., 2026b), R-Sparse (Zhang et al., 2025), ActTail (Hou et al., 2026), WiSparse (Chen et al., 2026a) and La RoSA (Liu et al., 2025b), under a calibrated rotation, sparsify matrix inputs, at their best with calibrated thresholds or allocations; R-Sparse adds a low-rank path for omitted inputs. GRIFFIN (Dong et al., 2024), DART (Tyagi et al., 2026) and batch-wise adaptive pruning (Kim et al., 2026) need no calibration data but fix the neurons per sequence or batch from dense activations.

Selecting channels before reading them. Prox (Liu et al., 2026), contemporaneous with this work, is the closest method: it selects channels from INT4 proxies of the gate and up weights on a sparsified input, with thresholds calibrated layer by layer. HiRE (Samaga B L et al., 2024) shortlisted FFN coordinates with an INT4 weight copy and no calibration data, in a model pretrained for top-k sparsity. Reducing or pruning experts (Chen & Yao, 2026; Lu et al., 2024; Huang et al., 2026; Chitty-Venkata et al., 2025; Xu et al., 2026; Lasby et al., 2026) is orthogonal. We claim neither the first token-wise or intra-expert channel selection nor the first calibration-free one; our claim is that calibration is not what decides quality, and a weight-only ranking built on that finding.

3 Setting and cost model

To make the choice of channels precise, we first define a channel and what a policy pays for.

Channels. A mixture-of-experts (MoE) layer receives a hidden state $x \in \mathbb{R}^H$, and a frozen router selects the top-K experts with weights w_e. Expert e computes $D_e\big(\phi(G_e x) \odot V_e x\big)$ with gate and up matrices $G_e, V_e \in \mathbb{R}^{I \times H}$, down matrix $D_e \in \mathbb{R}^{H \times I}$, and ϕ SiLU or tanh-GELU. Row i of G_e and V_e together with column i of D_e form a *channel*. Indexing the $N = KI$ candidate channels of one token jointly by j, with router weight w_j, gate and up rows g_j, v_j and down column d_j, the routed output is

$$y = \sum_{j=1}^{N} w_j d_j a_j, \qquad a_j = \phi(g_j^\top x)\,(v_j^\top x). \tag{1}$$

A channel-sparse policy computes only a subset $\mathcal{F}$ of these terms exactly.

Score and keep rule. Every such policy makes two choices per token and layer: a *score* that ranks the candidate channels from something cheaper to read than their weights, and a *keep rule* that decides how many to compute. A threshold keeps the channels whose score exceeds a value tuned on calibration data, so how many it keeps depends on the input; a top-k keeps a fixed number and needs no data. This paper asks which of the two decides quality. Section 6.1 holds the score fixed and changes the keep rule; Section 4 builds a better score.

Causality and cost. The router and all weights stay frozen, and each policy keeps the router's natural top-K on its own hidden state; a selected expert may receive no exact channels. A decision must precede the reads it saves: the exact a_j is the best selection signal but already pays for g_j and v_j. Every byte a policy reads, including predictors, sketches, thresholds and recovery parameters, counts against it. We measure cost as these bytes per token and layer relative to the $3NH\eta$ bytes of stock (η bytes per element), priced in 32-byte DRAM sectors (a partly touched sector is paid in full) with each method on its best single weight layout (Appendix C): an analytical count for one token, not measured traffic. Shared experts, dense MLPs, attention and the router are unchanged and excluded; resident memory and measured time are reported separately. All methods run inside vLLM (Kwon et al., 2023), so each policy evolves its own KV cache and routing.

4 METHOD

Since quality follows the ranking (Section 6.1), the reader puts its effort into the score and uses the simplest keep rule, one top-F per token. Everything it uses is built from the weights alone. It makes one paid screening decision per token and MoE layer: it (i) screens all N candidate channels with a resident ternary sketch, (ii) selects F of them by one global top-F, (iii) computes the selected channels exactly from the native weights and (iv) recovers the omitted ones through a low-rank map built from the down projections, using screen features already paid for (Sections 4.1–4.4, Figure 1; pseudocode in Appendix A). This is *Ours*; without (iv) it is an ablation. For extreme sparsity, (v) an input-sparse screen reads the sketch only at each token's largest rotated input coordinates (Section D), an extension line because its saving is priced analytically. Fitting the map of (iv) on calibration text gives a reference (Appendix Table 3).

4.1 TERNARY SKETCH SCREEN

The score must be computed before a channel's weights are read, so it needs a stand-in for those weights that costs far fewer bytes. Reading a few exact coordinates of each row is cheap in elements but not in memory traffic: a GPU moves data between its DRAM and L2 cache in 32-byte sectors of 16 BF16 values, so a scattered coordinate subset touches most sectors of the rows it samples. We keep a low-bit copy of *every* coordinate. Let $R = \mathrm{blockdiag}(H_{128}, \ldots, H_{128})\,\mathrm{diag}(s)/\sqrt{128}$ be a block-diagonal randomized Hadamard transform with random signs s. Since R is orthogonal, $g^\top x = (Rg)^\top (Rx)$, and rotation spreads outlier coordinates across a block (Ashkboos et al., 2024; Tseng et al., 2024). For each gate or up row g and each 128-coordinate block b of Rg we store ternary codes (Li et al., 2016) $t_{b,c} = \mathrm{sign}((Rg)_{b,c})\mathbb{1}[|(Rg)_{b,c}| > \Delta_b]$ with $\Delta_b = 0.7\,\mathrm{mean}|(Rg)_b|$, the least-squares scale α_b for this support, and the mean squared residual $\rho_b = \mathrm{mean}_c((Rg)_{b,c} - \alpha_b t_{b,c})^2$. At decode time Rx is formed once per token and layer, and each candidate row receives an estimate and a residual variance

$$\hat{g} = \sum_b \alpha_b\, t_b^\top (Rx)_b, \qquad \sigma_g^2 = \sum_b \rho_b\, \|(Rx)_b\|^2, \tag{2}$$

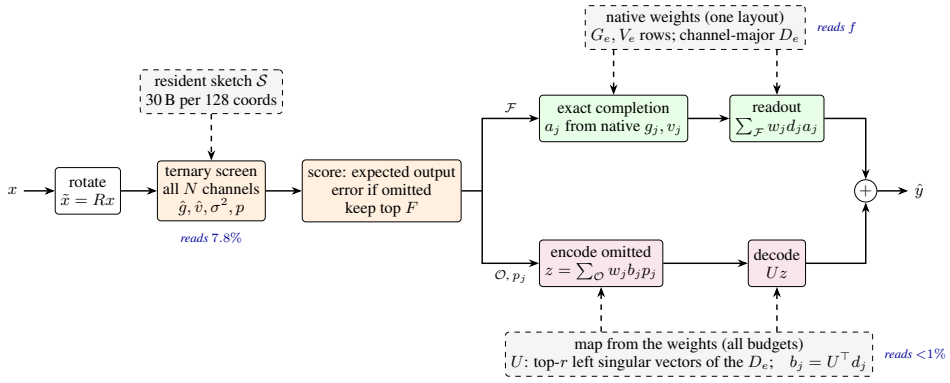

Figure 1: One MoE layer of the reader for one token; every part is built from the weights alone. One screen of all N candidate channels, from a sketch of the weights, decides which F channels are completed exactly from the native weights (top); a rank-r map built from the down projections recovers the omitted channels from their paid sketch activations p_j (bottom). Dashed arrows: resident state read. Blue: fraction of stock routed-expert bytes read (BF16, $f = F/N$).

and likewise $\hat{v}, \sigma_v^2$ for the up row. Five trits pack into a byte, so a block costs 26 bytes of codes plus FP16 α_b and ρ_b: 30 bytes instead of 256 (11.7% of the gate and up bytes, hence 7.8% of the candidate experts' routed bytes). It is computed from the weights once at load time and never replaces them; the threshold 0.7 of Li et al. (2016) and blocks of 128 were chosen on OLMoE development documents and transferred unchanged to every model.

4.2 Selection

A single top-F over all of a token's candidates is fair only if the scores of channels in different experts share one scale. Each token completes the $F = \mathrm{round}(fN)$ channels with the largest score among its N candidates, where the retention f is fixed by the read budget (plan P10 keeps $f = 0.10$). The score puts every channel on the scale of the layer's output, its expected squared contribution to the layer output, $w_j^2 \|d_j\|^2 \, \mathbb{E}[a_j^2]$, so the count can be global over the token's candidates rather than a per-expert quota: a routed expert may receive no channels, and the single screen fixes membership before any exact activation is computed. $\mathbb{E}[a_j^2]$ treats the sketch residuals as independent Gaussian noise around $\hat{g}_j$ and $\hat{v}_j$ and is computed with a fixed five-node quadrature (Appendix B); without recovery the point estimate $\phi(\hat{g}_j)^2 \hat{v}_j^2$ ranks as well (Table 6); the recovery-aware score needs the moments. Because the score is in output units, no threshold or per-layer budget is needed, and the reads are fixed per token by construction.

4.3 Exact completion and readout

For $j \in \mathcal{F}$ the reader reads the native gate and up rows, computes a_j exactly and reads the down column d_j. To make that read contiguous we re-lay D_e once at load time into channel-major storage; this one re-laid layout serves both stock prefill and sparse decode, and no weight is copied.

4.4 Recovery of omitted channels

The omitted channels are individually small, but together they carry the rest of the output, 19% of its energy at P10. Recovering them needs no data either: each $j \in \mathcal{O}$ has a sketch activation $p_j = \phi(\hat{g}_j)\, \hat{v}_j$, already paid for, and a per-layer rank-r map carries it to the output:

$$\hat{y} = \underbrace{\sum_{j \in \mathcal{F}} w_j d_j a_j}_{\text{exact}} + U \underbrace{\sum_{j \in \mathcal{O}} w_j \, b_j \, p_j}_{\text{latent } z \in \mathbb{R}^r}, \tag{3}$$

with a shared decoder $U \in \mathbb{R}^{H \times r}$ and a per-expert encoder whose row b_j belongs to channel j. We build this map from the weights as well: U holds the top-r left singular vectors of the layer's down-projection columns d_j over all its experts, and $b_j = U^\top d_j$, so recovery adds the rank-r projection

of the sketch's estimate of the omitted output. We use $r = 32$; the map is about 0.5% of each model, serves every read budget, and reads only the encoder rows of omitted channels and U (0.5–0.8 points of reads). With recovery the score becomes the output error left after recovery (Appendix B); fitting the map to stock outputs on calibration text, one bank per budget, gives the calibrated reference (Appendix G).

Cost and an extension. Per token and layer the reader reads the sketch of every candidate row, the native rows of the F chosen channels and the recovery map's rows for the omitted ones: about $0.078 + f$ of stock's routed bytes, plus 0.5–0.8 points for recovery, or 18.5% at $f = 0.10$ on OLMoE (Eq. (5), Appendix C). Every term reads contiguous runs, so sector pricing adds under 1%. At the sparsest budget the sketch becomes the main cost (7.8 of 13.5% at P05); an *input-sparse screen* reads it only at each token's 80 largest rotated input coordinates per block (Appendix D). We report it as an extension line at its analytical price and claim no speed for it, since our kernels still read the whole sketch.

5 Experimental setup

Models. We evaluate four frozen MoE models in their official BF16 releases (widths and sketch sizes in Table 5, Appendix C): OLMoE-1B-7B (Muennighoff et al., 2025), a base model; Qwen3.5-35B-A3B (Qwen Team, 2026), with an always-active shared expert and hybrid linear/full attention, the model Park et al. (2026) evaluate; Qwen3-30B-A3B-Thinking-2507 (Yang et al., 2025), a reasoning model; and Gemma 4 26B-A4B-it (Gemma Team et al., 2026), a second family with GELU experts beside a dense MLP.

Evaluation design. Three rules fix what is measured. *Format:* each checkpoint runs in the format in which its stock model reproduces a published number: likelihood and generation for OLMoE and Qwen3.5, chat-format generation for Qwen3-Thinking and Gemma 4 (Appendix H). *Exposure: full-sequence sparsity* also sparsifies the prompt (Park's and Prox's likelihood protocols, harsher than deployment); *decode-only sparsity* prefills with stock vLLM and sparsifies every generated token (the deployed regime, as prefill is compute-bound); the two are never pooled. *Matched reads:* baseline thresholds are calibrated on common data with targets at our budgets, we report the reads each realizes on each task, and we claim a difference only under dominance or bracketing (then marked interpolated), never by extrapolation.

Read budgets. Plans are named by retention, $f \in \{0.05, 0.10, 0.20, 0.30\}$ (P05–P30); our reader reads about 13.5, 18.5, 28.5 and 38.4% of stock routed bytes on OLMoE (Eq. 5; 0.5–0.7 points less without recovery), and points are named by reads with the plan in parentheses, e.g. 18.5% (P10). P10 and P30 were fixed by cost before any result and P05 added for the heaviest budgets; OLMoE, Qwen3.5 under Park's protocol and the fidelity measurements cover all four plans.

Baselines. All baselines run in the same vLLM runtime on the same checkpoints, calibrate on common data that our reader does not use and are priced on their best single layout (Appendices F and G). *Input-sparse* methods, TEAL (Liu et al., 2025a) with its greedy per-layer allocation, WINA (Chen et al., 2026b) and R-Sparse (Zhang et al., 2025), reach any budget. *Gate- or up-thresholded* methods, CATS (Lee et al., 2024), CHESS (He et al., 2024), M-COUNTDOWN (Cheon & Kang, 2025) and Park et al. (2026), read a dense gate (or up) projection first, a third of routed bytes. *Prox* (Liu et al., 2026), designed for dense models, selects channels with a resident INT4 proxy of the gate and up weights evaluated on the inputs that survive a calibrated first threshold; we port it to experts, price its proxy coordinate-major, and sweep the split of reads between its two stages, keeping the allocation with the lowest development KL before any held-out result. *Expert-count reduction* (Chen & Yao, 2026) is tuned as in its paper. OLMoE runs every baseline; the larger models carry Prox and TEAL, the strongest on OLMoE, and Qwen3.5 every baseline but R-Sparse.

Benchmarks. Full standard test sets with LM Evaluation Harness 0.4.3 (Gao et al., 2023) unless noted. Full-sequence sparsity uses the baselines' own protocols: Park's five zero-shot tasks, ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020) and TruthfulQA-mc2 (Lin et al., 2022) (*zero-shot accuracy*, their mean), and Prox's few-shot MMLU

(Hendrycks et al., 2021) and PIQA (Bisk et al., 2020) set. GPQA-Diamond (Rein et al., 2024), IFEval (Zhou et al., 2023) and a fixed 1,000-question MMLU-Pro subset (Wang et al., 2024) run in each card's thinking format; GSM8K (Cobbe et al., 2021) is in Appendix J. OLMoE adds OLMES (Gu et al., 2025) BIG-Bench Hard with chain of thought (Suzgun et al., 2023), GSM8K, the OLMo 2 generative tasks (Walsh et al., 2025) and WikiText-2 perplexity (Merity et al., 2017) (Appendix H). We report standard errors (clustered by question), paired differences and capped outputs.

Fidelity to stock. Because accuracy can hide changed answers (Dutta et al., 2024), we also report KL divergence from stock and top-1 agreement over the full vocabulary: on OLMoE on 192 held-out Pile documents disjoint from all calibration and development data, and on the post-trained models on chat conversations held out from fitting but used during development (Appendix G), with the fraction of benchmark answers that change.

6 RESULTS

The results follow the paper's argument. We first ask whether channel selection needs calibration data and what decides its quality (Section 6.1), then compare our reader with every baseline at equal reads on OLMoE (Section 6.2) and on the post-trained models in long reasoning (Section 6.3), and end with what each component contributes and how fast the reader runs.

6.1 CALIBRATION IS NOT WHAT DECIDES QUALITY

Top-k in place of thresholds. If channel selection needed calibration data, removing it from the strongest calibrated selector should cost quality. *Prox without calibration* keeps Prox's INT4 proxy and score and replaces only its two thresholds by top-k selections: each token's 30% largest inputs (50% from P20; chosen on OLMoE development KL, as Prox's allocation, and used unchanged on every model) and its top-F proxy scores. On OLMoE's held-out documents it is closer to stock than calibrated Prox at every budget, by 2–17% in KL (0.438, 0.278, 0.118, 0.057 at 13.5–38.4% of reads; Figure 2(a)). On the post-trained models' chat panels, which come from Prox's own calibration distribution and so favour it, it changes Prox's KL by -6 to $+24\%$ at 13–18% of reads (Gemma 4 0.296 and 0.181, Qwen3-Thinking 0.202 and 0.131, Qwen3.5 0.097 and 0.068). On GPQA-Diamond calibration costs Prox 50 points on Gemma 4 (below) and buys it 5.6 on Qwen3-Thinking (68.2 against 62.6), where our reader is level with calibrated Prox (-1.5 ± 2.7): its effect depends on the input.

The ranking decides. Changing the score moves KL far more. With the same top-F, an INT4 gate screen has 3.5× our KL at 18% of reads on OLMoE and TEAL's input magnitudes 6.4× (Figure 5), and our sketch score ranks about as well as Prox's INT4 score with half its resident bytes (KL without recovery at 18% of reads: 0.301, 0.217, 0.134, 0.063 against 0.278, 0.181, 0.131, 0.068); recovery from the weights, fed by the same paid features, puts the reader 11–33% closer to stock (0.245, 0.122, 0.116, 0.057). A single quantity explains the order: the share of the layer's output energy that the kept channels carry. Across four scores and four budgets, held-out KL falls monotonically with it (rank correlation -0.96; Appendix E), and a per-layer threshold fitted on data, in place of top-F, lowers that share for every score. At 13% of reads Prox's score is still 2–10% closer on OLMoE and the Qwen models while ours is 21% closer on Gemma 4.

Calibrated budgets drift. Thresholds also make the cost unpredictable. A threshold keeps as many channels as the activations exceed it, so its reads follow the input (Appendix Table 8). Prox thresholds that read 15–17% of routed bytes on held-out chat from their calibration distribution read 10–15% on the benchmarks and fall from 15.6% to 10.6% along Qwen3-Thinking's reasoning traces. On Gemma 4, thresholds that read 17.3% on chat read 13.3% on Prox's own GPQA-Diamond answers, inside the region where it collapses into repetition (Section 6.3). A top-F reads its budget on every token of every input, so its read cost and workspace are known before traffic arrives.

6.2 THE READER AGAINST EVERY BASELINE

We now compare the full reader, with its score and recovery, against all nine baselines at their best calibrated settings. On OLMoE it nearly matches Prox at 13% of routed reads and is closer to

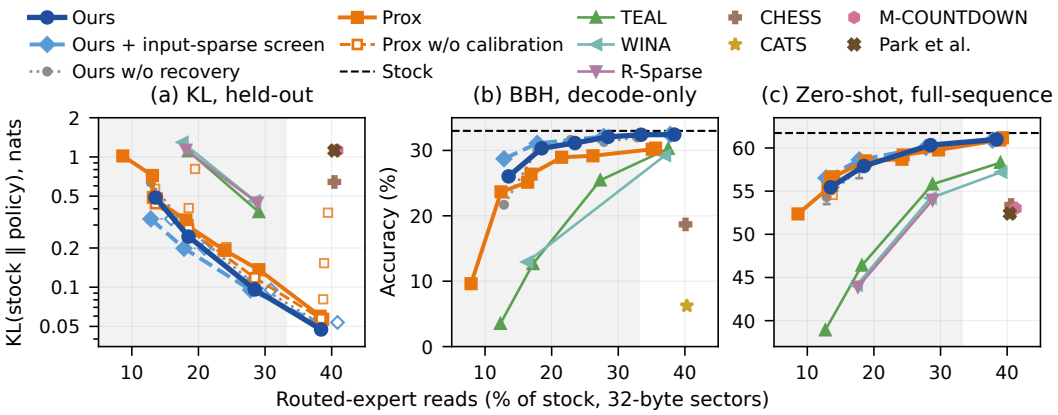

Figure 2: OLMoE-1B-7B versus sector-priced routed-expert reads. (a) KL divergence from stock over the full vocabulary on 192 held-out documents (12,096 decode positions; log scale) and (b) BIG-Bench Hard, 3-shot chain of thought, under decode-only sparsity (stock prompt prefill, every generated token sparse); (c) zero-shot accuracy, the mean of the five tasks of Park et al. (2026), under full-sequence sparsity. Ours: the data-free reader with weight-only recovery; light dashed: with the input-sparse screen at its analytical batch-one price (hollow: the whole sketch charged); grey dotted: without recovery (bars: five rotation seeds). Prox: the line through its best measured points; hollow: its other allocations. Thresholded baselines at the reads they realize; shaded: below the reads of a dense gate projection.

stock from 18% up (Figure 2(a): OLMoE, held-out documents): KL 0.488 against 0.471 on the curve through Prox's best points at 13.5%, 0.245 against 0.293 at 18.5% (both interpolated), and it dominates Prox's 29.1% and 38.7% points (0.096 against 0.136, 0.047 against 0.057). At 13–18% every other method is at least three times farther from stock. The input-sparse extension reaches 0.33 at 12.8%, below every Prox point under 18%, even when charged for the whole sketch.

Task accuracy follows (panel (b)): on BIG-Bench Hard with chain of thought the reader keeps 79% of stock accuracy at 13.5% and 92% at 18.5%, 5 and 9 points of stock accuracy above Prox's interpolated curve, and 98% at 38%; the extension keeps 87% at 12.8%. On GSM8K Prox leads by up to 1.7 points below 38% (SE 0.9; other tasks in Appendix M). With the prompt sparse too, on Park's and Prox's own protocols (panel (c)), the reader trails Prox by at most a point and leads every other method widely (Appendix L).

6.3 Post-trained models and long reasoning

The post-trained models test the claim where it matters most: long answers generated token by token, in which every step's error feeds the next. In KL the reader is closer to stock than Prox at 18% of reads on Gemma 4 and level on the Qwen models (Figure 3, top; KL 0.122 against 0.193, 0.116 against 0.121, 0.057 against 0.055, Prox interpolated log-linearly); below 16% Prox is closer on both Qwen models. Long generation separates the methods further (bottom; Table 1). On Gemma 4 GPQA-Diamond, where Prox's thresholds drift to fewer reads than on chat (Section 6.1), Prox falls off a cliff below 16% of reads (45.5 at 15.5%, 20.2 at 14.0%, 11.6 at 13.3%), repeating lines until the length cap, whereas ours keeps 51.0 at 13.4% (paired $+30.8 \pm 4.1$ over Prox at 14.0%) and 73.2 at 18.4% (stock 80.3); without recovery the reader falls off the same cliff (3.5 at 12.8%; Appendix K). Prox without calibration does not collapse (62.1 at 13.4%, ahead of ours, and 68.7 at 18.4%), so at equal reads the collapse comes from the threshold, not from Prox's score. At 13%, where our reader is 21% closer in KL, Prox's score keeps more of GPQA-Diamond (62.1 against 51.0): at the sparsest budget KL does not fully predict long reasoning. On IFEval ours keeps 93.0 at 18.4% (stock 93.5) and 80.2 at 13.4%, where Prox scores 11.3 at 14.2%. Qwen3-Thinking rarely loops, and there Prox is as good as ours (68.2 at 17.9% against 66.7 at 18.6%, paired -1.5 ± 2.7) and at P05 matches or beats us on all three tasks with 2.4–2.9 fewer points of reads; on Qwen3.5 all strong selectors stay within 2.5 points of stock at P10.

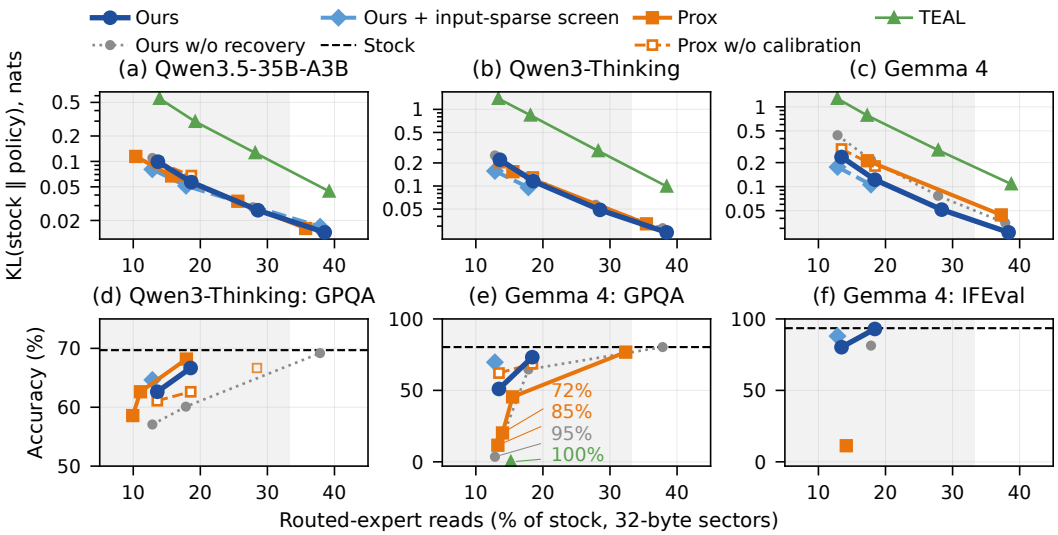

Figure 3: Decode-only sparsity on the post-trained models versus sector-priced routed-expert reads. Top: KL divergence from stock on 128 development chat conversations held out from fitting (log scale). Bottom: GPQA-Diamond (first of four samples per question, each card's thinking format) and IFEval prompt-level strict; labels give the share of outputs that reach the length cap. Lines and markers as in Figure 2.

6.4 What each component contributes

Each component helps most where reads are scarcest (Table 3). Recovery from the weights costs 0.5–0.8 points of reads, removes 12–23% of the reader's KL on OLMoE and 10–44% at P10 on the post-trained models, and lifts Gemma 4 GPQA-Diamond at P05 from 3.5 to 51.0 and IFEval at P10 from 81.3 to 93.0. The input-sparse screen dominates Ours at P05 and P10 on all four models. Without recovery, a per-expert quota instead of the global top-F costs 72% more KL and the rotation lowers KL by 18% (Table 6).

6.5 Speed

Fewer reads matter only if they make decoding faster. We time the full MoE block (router, routed and shared experts or dense MLP; Table 2) on captured decode states against unmodified vLLM with CUDA graphs and its tuned fused-MoE kernels (the fastest configuration per batch size) on one RTX PRO 6000. At batch one the MoE block runs 1.67× faster on OLMoE and 1.38× on Qwen3-Thinking, and end-to-end decoding 1.30× and 1.16×; the gain peaks at batch 4–8 and falls to 0.77–1.03× at batch 128, as a batch reads the union of its tokens' channels. On Qwen3.5, with half of OLMoE's routed bytes per layer, the reader's fixed latency outweighs the saving at batch one (0.85×), and Gemma 4's dense MLP bounds its gain to 1.25–1.28× (Appendix N).

7 Limitations and conclusion

Limitations. The reader's design, constants and input-sparse keep fraction were chosen on OLMoE development documents. Heavy sparsity costs long reasoning: at 13% of reads Gemma 4 loses 29 GPQA-Diamond points with our reader and 18 with Prox without calibration, and on Qwen3-Thinking Prox matches ours with fewer reads, and our models activate only 1–4B parameters per token. On quantized weights the sketch's share of the bytes grows, as does Prox's (Appendix C).

Conclusion. We asked whether choosing which channels of an MoE expert to read needs calibration data. It does not: the strongest calibrated selector loses little fidelity when its thresholds become top-k selections, quality follows how well channels are ranked, and calibrated thresholds make both cost and quality depend on the input, to the point of collapsing Gemma 4's long reasoning. A reader

Table 1: Decode-only sparsity on the post-trained models. GPQA-Diamond: on Qwen3.5 the mean of four samples per question, on the other models the first sample (198 answers, paired by question; Appendix K). Reads are those realized on GPQA-D; a baseline's reads on other tasks are in parentheses where they differ by more than a point. GSM8K in Appendix J.

Model	Method	Reads (%)	GPQA-D	IFEval	MMLU-Pro
Qwen3.5-35B-A3B	Stock	100	83.6	92.1	85.5
	Ours	18.7	82.6	92.2	83.0
	Ours	13.7	–	87.8	83.0
	w/o recovery	17.9	83.2	91.7	84.8
	Prox	14.2	83.1	91.1	84.1
Qwen3-30B-A3B-Thinking	Stock	100	69.7	87.8	80.5
	Ours	18.6	66.7	85.4	78.2
	Ours	13.6	62.6	82.3	75.2
	w/o recovery	17.8	60.1	85.2	76.4
	+ input-sparse screen	12.8	64.6	85.0	–
	Prox w/o calibration	18.6	62.6	–	–
	Prox w/o calibration	13.6	61.1	–	–
	Prox	17.9	68.2	–	–
	Prox	11.1	62.6	–	–
	Prox	9.9	58.6	82.8 (10.7)	76.3 (11.2)
Gemma 4 26B-A4B-it	Stock	100	80.3	93.5	85.1
	Ours	18.4	73.2	93.0	81.7
	Ours	13.4	51.0	80.2	66.7
	w/o recovery	17.8	64.6	81.3	77.9
	+ input-sparse screen	12.8	69.7	88.0	–
	Prox w/o calibration	18.4	68.7	–	–
	Prox w/o calibration	13.4	62.1	–	–
	Prox	15.5	45.5	–	–
	Prox	14.0	20.2	–	–
	Prox	13.3	11.6	11.3 (14.2)	–
	TEAL	15.2	0.0	–	–

Table 2: MoE-block speedup of Ours (P10) over the fastest tuned stock vLLM configuration per batch size, and end-to-end decode time per output token (TPOT) speedup, on one RTX PRO 6000. Full grid, the reader without recovery and the input-sparse extension in Appendix N.

Model	Measure	B=1	4	8	32	128
OLMoE-1B-7B	MoE block	1.67	1.92	1.81	1.29	0.77
	decode TPOT	1.30	1.60	1.60	1.29	0.83
Qwen3-30B-A3B-Thinking	MoE block	1.38	1.64	1.66	1.37	0.98
	decode TPOT	1.16	1.39	1.43	1.32	1.00
Qwen3.5-35B-A3B	MoE block	0.85	1.28	1.66	1.54	1.28
Gemma 4 26B-A4B-it	MoE block	1.01	1.25	1.28	1.27	1.03

that ranks channels from the weights alone matches or beats the strongest calibrated selector in fidelity to the stock model at 18–28% of reads on all four models, reads exactly its budget on every input and speeds up the MoE block at small batches.

AI USE STATEMENT

We used LLM assistants extensively to write code and kernels, run and monitor experiments, analyze results and draft text. The authors set the research questions and design, reviewed code and results, checked reported numbers against run outputs, and take full responsibility for the paper.

REPRODUCIBILITY STATEMENT

Sections 4–5 and Appendices C and F specify the method, protocols and baselines; code and recovery banks will be released upon publication.

REFERENCES

Yash Akhauri, Ahmed F. AbouElhamayed, Jordan Dotzel, Zhiru Zhang, Alexander M. Rush, Safeen Huda, and Mohamed S. Abdelfattah. ShadowLLM: Predictor-based contextual sparsity for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024. doi: 10.18653/v1/2024.emnlp-main.1068. URL https://aclanthology.org/2024.emnlp-main.1068/.

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*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/b5b939436789f76f08b9d0da5e81af7c-Abstract-Conference.html.

Yonatan Bisk, Rowan Zellers, Ronan Le bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.

Lei Chen, Yuan Meng, Xiaoyu Zhan, Zhi Wang, and Wenwu Zhu. Wisparse: Boosting llm inference efficiency with weight-aware mixed activation sparsity, 2026a. URL https://arxiv.org/abs/2602.14452.

Sihan Chen, Dan Zhao, Jongwoo Ko, Colby Banbury, Huiping Zhuang, Luming Liang, Pashmina Cameron, and Tianyi Chen. Wina: Weight informed neuron activation for accelerating large language model inference. In *International Conference on Learning Representations*, 2026b. URL https://arxiv.org/abs/2505.19427.

Xing Chen and Hengshuai Yao. Training-free halving of activated experts in fine-grained mixture-of-experts models, 2026. URL https://arxiv.org/abs/2609.04575.

Jaewon Cheon and Pilsung Kang. COUNTDOWN: Contextually sparse activation filtering out unnecessary weights in down projection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025. doi: 10.18653/v1/2025.emnlp-main.1442. URL https://aclanthology.org/2025.emnlp-main.1442/.

Krishna Teja Chitty-Venkata, Sandeep Madireddy, Murali Emani, and Venkatram Vishwanath. LExI: Layer-adaptive active experts for efficient MoE model inference, 2025.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

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, 2021. URL https://arxiv.org/abs/2110.14168.

Harry Dong, Beidi Chen, and Yuejie Chi. Prompt-prompted adaptive structured pruning for efficient LLM generation. In *Conference on Language Modeling*, 2024.

Abhinav Dutta, Sanjeev Krishnan, Nipun Kwatra, and Ramachandran Ramjee. Accuracy is not all you need. In *Advances in Neural Information Processing Systems*, 2024. doi: 10.52202/079017-3950. URL https://doi.org/10.52202/079017-3950.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL https://jmlr.org/papers/v23/21-0998.html.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Aditya Chawla, Victor Cotruta, Alice Coucke, Phil Culliton, Robert Dadashi, Lucas Dixon, Mohamed Elhawaty, Utku Evci, Clément Farabet, Johan Ferret, Filippo Galgani, Sertan Girgin, Jean-Bastien Grill, Maarten Grootendorst, Jiaxian Guo, Cassidy Hardin, Yanzhang He, Steven M. Hernandez, Omri Homburger, Léonard Hussenot, Juyeong Ji, Armand Joulin, Aishwarya Kamath, Parnian Kassraie, Olivier Lacombe, Preethi Lahoti, Gaël Liu, Gus Martins, Luciano Martins, Tatiana Matejovicova, Ramona Merhej, Nikola Momchev, Sneha Mondal, Ryan Mullins, Sindhu Raghuram Panyam, Shreya Pathak, Sarah Perrin, André Susano Pinto, Etienne Pot, Angéline Pouget, Alexandre Ramé, Sabela Ramos, Douglas Reid, David Rim, Morgane Rivière, Karsten Roth, Louis Rouillard, Omar Sanseviero, Pier Giuseppe Sessa, Shane Settle, Danila Sinopalnikov, Sara Smoot, Piotr Stanczyk, Andreas Steiner, Lawrence Stewart, Ilya Tolstikhin, Michael Tschannen, Anton Tsitsulin, Nino Vieillard, Renjie Wu, Pingmei Xu, Haichuan Yang, Edouard Yvinec, Biao Zhang, Li Zhang, Joe Zou, Nicolas Aagnes, Abdelrahman Abdelhamed, Jakub Adamek, Shivani Agrawal, Shubham Agrawal, Ibrahim Alabdulmohsin, Jean Baptiste Alayrac, Uri Alon, Chandramouli Amarnath, Ankesh Anand, Chrysovalantis Anastasiou, Setareh Ariafar, François-Xavier Aubet, Kyriakos Axiotis, Federico Barbero, Joelle Barral, Alexei Bendebury, Urs Bergmann, Stanley Bileschi, Kat Black, Mathieu Blondel, Sebastian Borgeaud, Arthur Bražinskas, Ryan Burnell, Robert Busa-Fekete, Mu Cai, Daniele Calandriello, Glenn Cameron, Charlotte Caucheteux, Rahma Chaabouni, Garima Chadha, Jetha Chan, Blake Jianhang Chen, Jesse Chen, Lin Chen, Xu Chen, Derek Cheng, Tzu hsiang Chien, Nikolai Chinaev, Yi Chou, Zhaohui Chu, Benjamin Coleman, Pooja Consul, Sam Conway-Rahman, Scott Crowell, Dylan Cutler, Vivek Dani, Samira Daruki, Anil Das, Daniel Deutsch, Nishanth Dikkala, Li Ding, Qiuhan Ding, Shenil Dodhia, Konstantin Donhauser, Tulsee Doshi, Anca Dragan, Alex Druinsky, Sahil Dua, Zoltan Egyed, Danielle Eisenbud, Daniel Eppens, Cindy Fan, Bahare Fatemi, Yassir Fathullah, Vlad Feinberg, Milen Ferev, Sebastian Flennerhag, Takumi Fujimoto, João Gabriel Oliveira, Isaac Galatzer-Levy, João Gante, Simon Geisler, Soham Ghosal, Antonious M. Girgis, Tamara von Glehn, Alec Go, Alhaad Gokhale, Alex Grills, Yiming Gu, Mayank Gupta, Pramod Gupta, Guru Guruganesh, Raia Hadsell, Hamza Harkous, Jitendra Harlalka, Demis Hassabis, Anja Hauth, Joe Heyward, Arian Hosseini, Chih-Yang Hsia, I-Hung Hsu, Xiaopeng Huang, Yangsibo Huang, Kevin Hui, Adrian Hutter, Te I, Fotis Iliopoulos, Advait Jain, Ganesh Jawahar, Ziwei Ji, Qilin Jin, Melvin Johnson, Kandarp Joshi, Arun Kandoor, Wang-Cheng Kang, Koray Kavukcuoglu, Mehran Kazemi, Kathleen Kenealy, Amr Khalifa, Phoebe Kirk, Ivan Korotkov, Suraj Kothawade, Vitaly Kovalev, Neel Kovelamudi, Adam Kraft, Ravin Kumar, Vivek Kumar, Harish Kuppam, Justin Lannin, Chen-Yu Lee, Seungji Lee, Dmitry Lepikhin, Alon Levkovitch, Dongdong Li, Qiujia Li, Valentin Liévin, Ethan Lin, Ziqian Lin, Casper Liu, Tianlin Liu, Tianqi Liu, Xin Liu, Ivan Lobov, Mayank Lunayach, Min Ma, Gagan Madan, Andrii Maksai, Eric Malmi, Michal Matuszak, Daniel McDuff, Gaurav Menghani, Maciej Mikuła, Daniil Mirylenka, Karolis Misiunas, Vedant Misra, Andreea Mitran, Kareem Mohamed, Maksim Mukha, Eric Noland, James O'Donnell, Brendan O'Donoghue, Kate Olszewska, Bernett Orlando, Wanqiong Pan, Rina Panigrahy, Unnati Parekh, Nicolas Perez-Nieves, Chunjong Park, Eric Paskie, Liqian Peng, Bryce Petrini, Slav Petrov, Jonas Pfeiffer, Bilal Piot, Martyna Plomecka, Siim Poder, Octavio Ponce, Arijit Pramanik, David Racz, Anish Rajan, Michelle Ramanovich, Anand Rao, Marvin Ritter, Vitor Rodrigues, Evan Rosen, Mikołaj Rybiński, Noveen Sachdeva, Michaël E. Sander, Rohit Sathyanarayana, Sagar Savla, Samuel Schmidgall, Tal Schuster, George Scrivener, Benoit Seguin, Andrew Sellergren, Aliaksei Severyn, Izhak Shafran, Dhruv Shah, Bobak Shahriari, Yuan Shangguan, Ashish Shenoy, Pradeep Shenoy, Rakesh Shivanna, Pauline Sho, Lucas Spangher, Wojciech Stokowiec, Tim Strother, Yao Su, Yinghao Sun, Mukund Sundararajan, Andrea Tacchetti, Mor Hazan Taege, Pouya Tafti, Jean Tarbouriech, Chetan Tekur, Shantanu Thakoor, Rahul Thapa, Madeleine Traverse, Lenart Treven, Tao Tu, Chien Te Tung, Çağlar Ünlü, Petar Veličković, Malini Pooni Venkat, Sagar Gubbi Venkatesh, Vidya Venkiteswaran, Francesco Visin, Alex Vitvitskyi, Kiran Vodrahalli, Weiyi Wang, Xin Wang, Tris Warkentin, Jan Wassenberg, John Wieting, Cindy Wu, Lechao Xiao, Hao Xu, Yuhui Xu, Fuzhao Xue, Arun Yadav, Jun Yan, Antoine Yang, Lin Yang, Ming-Hsuan Yang, Ziyu Ying, Jae Hyeon Yoo, Morteza Zadimoghaddam, Sajjad Zafar, Fred Zhang, Jiageng Zhang, Jianyi Zhang, Xiaofan Zhang, Chao Zhao, David Zhou, and Chen Zou. Gemma 4 technical report, 2026. URL `https://arxiv.org/abs/2607.02770`.

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. OLMES: A standard for language model evaluations. In *Findings of the Association for Com-*

putational Linguistics: NAACL 2025, 2025. doi: 10.18653/v1/2025.findings-naacl.282. URL https://aclanthology.org/2025.findings-naacl.282/.

Junhui He, Shangyu Wu, Weidong Wen, Chun Jason Xue, and Qingan Li. CHESS: Optimizing LLM inference via channel-wise thresholding and selective sparsification. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024. doi: 10.18653/v1/2024.emnlp-main.1038. URL https://aclanthology.org/2024.emnlp-main.1038/.

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*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Wenwen Hou, Xinyuan Song, and Shiwei Liu. Acttail: Global activation sparsity in large language models, 2026. URL https://arxiv.org/abs/2603.12272.

Yushi Huang, Zining Wang, Zhihang Yuan, Yifu Ding, Ruihao Gong, Jinyang Guo, Xianglong Liu, and Jun Zhang. Modes: Accelerating mixture-of-experts multimodal large language models via dynamic expert skipping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30205–30215, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Huang_MoDES_Accelerating_Mixture-of-Experts_Multimodal_Large_Language_Models_via_Dynamic_Expert_CVPR_2026_paper.html.

Yongmin Kim, Shota Takashiro, Yusuke Iwasawa, Takeshi Kojima, and Yutaka Matsuo. Batch-wise adaptive pruning: Periodic neuron activation-aware weight pruning for language reasoning model, 2026. URL https://arxiv.org/abs/2608.14003.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph 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*, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Mike Lasby, Ivan Lazarevich, Nish Sinnadurai, Sean Lie, Yani Ioannou, and Vithursan Thangarasa. Reap the experts: Why pruning prevails for one-shot moe compression. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.13999.

Aristotelis Lazaridis, Aman Sharma, Dylan Bates, Brian King, Vincent Lu, and Jack FitzGerald. When can one neuron fix repetition loops in LLMs?, 2026. URL https://arxiv.org/abs/2606.13705.

Donghyun Lee, Jaeyong Lee, Genghan Zhang, Mo Tiwari, and Azalia Mirhoseini. Cats: Context-aware thresholding for sparsity in large language models. In *Conference on Language Modeling*, 2024. URL https://colmweb.org/2024/AcceptedPapers.html.

Junyoung Lee, Sehyeon Park, Shinhyoung Jang, Seonha Ryu, Hojeong Kim, Hyunsei Lee, Il Hong Suh, and Yeseong Kim. FOCUS & RePAIR: Mitigating text degeneration via token-level guidance for pruned large language models, 2026. URL https://arxiv.org/abs/2608.26676.

Fengfu Li, Bin Liu, Xiaoxing Wang, Bo Zhang, and Junchi Yan. Ternary weight networks, 2016. URL https://arxiv.org/abs/1605.04711.

Xinyu Lian, Walid Krichene, Beichen Huang, Masahiro Tanaka, Olatunji Ruwase, Li Zhang, and Minjia Zhang. Quantization inflates reasoning: Token inflation as a hidden cost of low-bit reasoning models, 2026. URL https://arxiv.org/abs/2606.25519.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229/.

James Liu, Pragaash Ponnusamy, Tianle Cai, Han Guo, Yoon Kim, and Ben Athiwaratkun. Training-free activation sparsity in large language models. In *International Conference on Learning Representations*, 2025a. URL https://arxiv.org/abs/2408.14690.

Jinyi Liu, Wei Chen, Pengyu Chen, Xinyi Yuan, Minghe Bai, Guoquan Wu, and Jun Wei. Prox: Training-free ffn activation sparsity via approximate intermediate-channel salience in llms, 2026. URL https://arxiv.org/abs/2607.27591.

Kai Liu, Bowen Xu, Shaoyu Wu, Xin Chen, Hao Zhou, Yongliang Tao, and Lulu Hu. La RoSA: Enhancing LLM efficiency via layerwise rotated sparse activation, 2025b. URL https://arxiv.org/abs/2507.01299.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, and Beidi Chen. Deja vu: Contextual sparsity for efficient LLMs at inference time. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22137–22176. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/liu23am.html.

Sanae Lotfi, Polina Kirichenko, Steven Li, and Zechun Liu. Quantized reasoning models think they need to think longer, but they do not, 2026. URL https://arxiv.org/abs/2606.00206.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024. doi: 10.18653/v1/2024.acl-long.334. URL https://aclanthology.org/2024.acl-long.334/.

Ryan Lucas, Kayhan Behdin, Zhipeng Wang, Qingquan Song, Shao Tang, and Rahul Mazumder. Reasoning models can be accurately pruned via chain-of-thought reconstruction, 2025. URL https://arxiv.org/abs/2509.12464.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1609.07843.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. Olmoe: Open mixture-of-experts language models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2409.02060.

Jongseok Park, Sunga Kim, Zhenyu Gu, Ion Stoica, and Alvin Cheung. Uncovering intra-expert activation sparsity for efficient mixture-of-expert model execution, 2026. URL https://arxiv.org/abs/2605.08575.

Charilaos Pipis, Shivam Garg, Vasilis Kontonis, Vaishnavi Shrivastava, Akshay Krishnamurthy, and Dimitris Papailiopoulos. Wait, wait, wait... why do reasoning models loop?, 2025. URL https://arxiv.org/abs/2512.12895.

Qwen Team. Qwen3.5-35b-a3b. Official model card; accessed 2026-09-24, 2026. URL https://huggingface.co/Qwen/Qwen3.5-35B-A3B.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. In *First Conference on Language Modeling*, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. doi: 10.1609/aaai.v34i05.6399. URL https://doi.org/10.1609/aaai.v34i05.6399.

Yashas Samaga B L, Varun Yerram, Chong You, Srinadh Bhojanapalli, Sanjiv Kumar, Prateek Jain, and Praneeth Netrapalli. HiRE: High recall approximate top-k estimation for efficient LLM inference, 2024.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1701.06538.

Jiho Shin, Hoeseok Yang, and Youngmin Yi. Sparseinfer: Training-free prediction of activation sparsity for fast llm inference. In *2025 Design, Automation & Test in Europe Conference (DATE)*, 2025. doi: 10.23919/DATE64628.2025.10992997. URL https://doi.org/10.23919/DATE64628.2025.10992997.

Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. Powerinfer: Fast large language model serving with a consumer-grade gpu. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, 2024. doi: 10.1145/3694715.3695964. URL https://doi.org/10.1145/3694715.3695964.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with hadamard incoherence and lattice codebooks. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 48630–48656. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/tseng24a.html.

Abhishek Tyagi, Yunuo Cen, Shrey Dhorajiya, Bharadwaj Veeravalli, and Xuanyao Fong. DARTing through the drift: Dynamic tracing of knowledge neurons for adaptive inference-time pruning, 2026. URL https://arxiv.org/abs/2601.22632.

Evan Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James Validad Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 olmo 2 furious (colm's version). In *Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=2ezugTT9kU.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

Jin Xu, Xiaojiang Liu, Jianhao Yan, Deng Cai, Huayang Li, and Jian Li. Learning to break the loop: Analyzing and mitigating repetitions for neural text generation. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2206.02369.

Zukang Xu, Zhixiong Zhao, Xing Hu, Jiangyong Yu, Houji Wen, Jun Li, Zhe Jiang, and Dawei Yang. ACE: Adaptive calibration-free expert skipping for MoE-based LLMs, 2026. URL https://arxiv.org/abs/2609.05228.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Zhenyu Zhang, Zechun Liu, Yuandong Tian, Harshit Khaitan, Zhangyang Wang, and Steven Li. R-sparse: Rank-aware activation sparsity for efficient llm inference. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2504.19449.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. Large language models are not robust multiple choice selectors. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=shr9PXz7T0.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023.

The appendix follows the paper's argument: Appendices A–C complete the method and its read ledger, Appendix E shows why the screen works, Appendices F–H fix the evaluation, Appendices I–M give the results behind each claim of Section 6 (fidelity, short answers, long reasoning, full-sequence sparsity and OLMoE tasks), and Appendix N gives the speed at every batch size.

Names. *Ours* is the data-free reader with the data-free weight-only recovery map (rank 32, the top singular vectors of the stacked down projections; no data). *Ours w/o recovery* is the same reader without the map, an ablation. *Ours + input-sparse screen* is the extension for extreme sparsity (Section D): its reads are an analytical batch-one price, current kernels read the whole sketch, and we make no speed claim for its saving. *Calibrated recovery* is the same map fitted to the stock model's output distribution on calibration text, a reference for what data add (Appendix G). Plans are named by retention (P10 keeps 10% of candidate channels); reads are routed-expert bytes in % of stock. A difference between two methods is stated only when one dominates the other at matched reads or its measured points bracket the other's reads; interpolated values are marked as such.

Table 3: The reader and its variants on OLMoE at P10: data each needs, sector-priced reads and KL divergence from stock on the development documents under decode-only sparsity. The extension's reads are an analytical batch-one price (its kernels read the whole sketch: 20.9%).

Variant	Components	Data	Reads (%)	KL
Ours	(i)–(iv)	none	18.5	0.231
w/o recovery (ablation)	(i)–(iii)	none	17.8	0.285
+ input-sparse screen (extension)	(i)–(v)	none	17.8	0.194
Calibrated recovery (reference)	(iv) fitted to stock outputs	calibration text	18.5	0.167

A Reader pseudocode

Algorithm 1 summarizes one token and one MoE layer; Section 4 defines each step.

Algorithm 1 Reader for one token and one MoE layer (budget F; optional recovery of rank r).

Require: hidden state x; natural top-K experts and weights; sketch $\mathcal{S}$; optional bank $(U, \{B_e\})$

$\tilde{x} \leftarrow Rx$ ▷ block Hadamard, once per token and layer
for each candidate channel j of the K experts **do**
  $(\hat{g}_j, \sigma^2_{g,j}), (\hat{v}_j, \sigma^2_{v,j}) \leftarrow$ Eq. (2) from $\mathcal{S}$ and $\tilde{x}$
  $p_j \leftarrow \phi(\hat{g}_j)\hat{v}_j$; $s_j \leftarrow$ Eq. (4) ▷ no bank: $w_j^2 n_j^2 m_j$
$\mathcal{F} \leftarrow$ the F channels with largest s_j; $\mathcal{O} \leftarrow$ the rest
$y \leftarrow \sum_{j\in\mathcal{F}} w_j d_j \, \phi(g_j^\top x)(v_j^\top x)$ ▷ native rows, exact
if a bank is loaded **then**
  $z \leftarrow \sum_{j\in\mathcal{O}} w_j b_j p_j$; $y \leftarrow y + Uz$ ▷ second stream
return y

B SELECTION SCORE

We model the sketch residuals as independent Gaussian noise, $g_j \sim \mathcal{N}(\hat{g}_j, \sigma^2_{g,j})$ and $v_j \sim \mathcal{N}(\hat{v}_j, \sigma^2_{v,j})$, a heuristic uncertainty model for a deterministic residual rather than a posterior. Because $a_j = \phi(g_j)v_j$ is linear in v_j, $\mu_j = \mathbb{E}[a_j] = \mathbb{E}[\phi(g_j)]\,\hat{v}_j$ and $m_j = \mathbb{E}[a_j^2] = \mathbb{E}[\phi(g_j)^2]\,(\hat{v}_j^2 + \sigma^2_{v,j})$, so only one-dimensional expectations over g_j need quadrature (five Gauss–Hermite nodes). If channel j is omitted, recovery replaces $w_j d_j a_j$ by $w_j q_j p_j$ with response $q_j = Ub_j$. Writing $n_j = \|d_j\|$, $\beta_j^{\parallel} = d_j^\top q_j / n_j$ and $\beta_j^{\perp} = (\|q_j\|^2 - (\beta_j^{\parallel})^2)^{1/2}$, the expected squared error of omitting j decomposes exactly as

$$\mathbb{E}\big\|w_j d_j a_j - w_j q_j p_j\big\|^2 = w_j^2\Big[n_j^2\big(m_j - \mu_j^2\big) + \big(n_j\mu_j - \beta_j^{\parallel} p_j\big)^2 + \big(\beta_j^{\perp} p_j\big)^2\Big]: \tag{4}$$

the irreducible uncertainty of the activation, the error of the recovered component along d_j, and the error recovery introduces orthogonal to d_j. The values $n_j, \beta_j^{\parallel}, \beta_j^{\perp}$ are precomputed per channel in FP32 when a bank is loaded. The score ignores cross-channel error terms. With $\beta_j^{\parallel} = \beta_j^{\perp} = 0$ (no bank: Ours w/o recovery, Section 4.2) it reduces to the plain second moment $w_j^2 n_j^2 m_j$.

C READ ACCOUNTING

The reader's ledger. With η bytes per weight element, the bytes our reader reads per token and layer are

$$W = \underbrace{\tfrac{30}{128}\,2NH + 4N + H}_{\text{sketch, norms, signs}} + \underbrace{3\eta FH}_{\text{completion, readout}} + \underbrace{2r(N - F + H) + 8N}_{\text{recovery}}, \tag{5}$$

against $3NH\eta$ for stock: for BF16 about $0.078 + f$ plus 0.5–0.8 points of recovery, 18.5% at $f = 0.10$ on OLMoE. Every term reads contiguous runs of hundreds of bytes, so sector pricing adds less than 1% to the logical count; arithmetic and workspace are charged separately (Appendix C).

What is charged. The headline axis is the number of routed-expert bytes read per token, summed over MoE layers and expressed as a percentage of stock. It is an analytical count for one token: each required access is priced in 32-byte DRAM sectors (an access that touches any byte of a sector pays for the whole sector), without modeling cache hits or repeated loads. Auxiliary state that a method reads to decide what to read (sketches, proxies, thresholds, low-rank factors, per-channel metadata) is charged on the same axis. The always-active paths (Qwen3.5's shared expert, Gemma's dense MLP), attention and the router are identical for every method and are reported separately. Arithmetic, selection and ranking work, and workspace are reported beside the read axis, not folded into it. Logical (element-count) reads are reported alongside the sector price. Because Park et al. (2026) report sparsity over routed and shared-expert neurons together, we also give that figure and the fraction of all feed-forward weight bytes read. When several tokens in a batch share an expert they also share its reads, so the per-token count is an upper bound on the routed bytes per token of a batched decode step.

Best single layout. Each method is priced on the single weight layout that is best for it, and may not keep a second copy of any weight. Our reader stores gate and up rows natively and the down projection channel-major, so completion reads whole gate/up rows and readout reads whole down rows of length H. Gate- or up-thresholded methods (CATS, CHESS, Park, M-COUNTDOWN) read the dense gate (or up) projection, then the surviving rows of the other projection and of the channel-major down projection. Input-sparse methods (TEAL, WINA, R-Sparse) read the surviving input coordinates of coordinate-major gate/up storage and the surviving channel rows of channel-major down storage; R-Sparse also reads its low-rank factors and their scales, charged once per routed slot. Prox reads its INT4 proxy and then the selected channels' rows; on its best layout the proxy is stored coordinate-major with two output channels per byte, so each retained input coordinate of a routed expert costs exactly its width in bytes for gate and up together, plus the scales. Under expert parallelism, CHESS's and WINA's per-channel metadata is charged per rank-local slot. Expert-count methods read whole experts. With these layouts every method reads contiguous runs of at least 256 bytes, so its sector price equals its logical byte count plus its auxiliary bytes to within 1%. Threshold methods realize a data-dependent read fraction, which we count on the GPU for every task and report beside each score. For quality evaluation the baselines execute as masked dense kernels, which compute the same outputs as their sparse forms; their read fractions therefore describe the method on its best layout, not the traffic of these evaluation kernels, and we make no speed claim for them.

Our ledger. Eq. (5) charges, per token and layer: the ternary sketch of every gate and up row of the K routed experts (30 bytes per 128 coordinates); the full native gate and up rows and the down row of each of the F completed channels; a 4-byte FP32 norm per candidate channel and the rotation signs (one byte each, as stored); and, with the optional recovery, the r-dimensional encoder rows of the $N - F$ omitted channels, the shared decoder U and 8 more bytes of score geometry per candidate. The reader also serves official block-128 FP8 checkpoints, where the block scales of the rows read are charged too; every evaluation in this paper uses BF16. Under quantized weights the sketch, a fixed 30 bytes per 128 coordinates, grows as a share of routed bytes from 7.8% (BF16) to about 15.6% (FP8) and about 30% (INT4 with group scales); with recovery, P10 then reads about 27% of FP8 and 43% of INT4 routed bytes at OLMoE's widths (analytical), and a lower-bit screen (Appendix E) is the natural response. Prox's INT4 proxy grows likewise. Table 4 lists the BF16 points. On OLMoE at P10 the 17.84% of Ours w/o recovery splits into 7.81 points of sketch, 10.00 of native rows and 0.03 of norms and rotation signs; the recovery of Ours adds 0.67 (encoder rows, decoder and score geometry). We do not report DRAM traffic from profiling counters; Section 6.5 reports measured time.

Table 4: Read fraction (%) of routed-expert bytes per token: Ours w/o recovery at each plan, and Ours (with the weight-only recovery) in parentheses. The input-sparse extension is priced at about the reads of Ours w/o recovery (12.8, 17.8, 27.8 and 37.8% on OLMoE at batch one); charged its whole sketch, as current kernels read it, it reads 15.9, 20.9, 30.9 and 40.9%.

Model	P05	P10	P20	P30
OLMoE-1B-7B	12.9 (13.5)	17.8 (18.5)	27.8 (28.5)	37.9 (38.4)
Qwen3-30B-A3B-Thinking	12.8 (13.6)	17.8 (18.6)	27.9 (28.5)	37.8 (38.5)
Qwen3.5-35B-A3B	12.9 (13.7)	17.9 (18.7)	27.8 (28.6)	37.9 (38.5)
Gemma 4 26B-A4B-it	12.8 (13.4)	17.8 (18.4)	27.8 (28.4)	37.8 (38.4)

Table 5: Models and evaluation. E routed experts, top-K, expert width I, hidden width H; sketch size relative to the BF16 checkpoint (the recovery bank adds about 0.5%). Full-sequence: likelihood protocols with the prompt sparse too. Decode-only: stock prompt prefill, every generated token sparse.

Model	E / K	I / H	Always active	Sketch	Full-sequence	Decode-only
OLMoE-1B-7B (base)	64 / 8	1024 / 2048	–	7.3%	Park; WikiText-2	BBH, GSM8K, OLMo 2 gen.; fidelity
Qwen3.5-35B-A3B	256 / 8	512 / 2048	shared expert	7.3%	Park; Prox	GSM8K (chat), GPQA-D, IFEval, MMLU-Pro
Qwen3-30B-A3B-Thinking	128 / 8	768 / 2048	–	7.4%	–	GSM8K, GPQA-D, IFEval, MMLU-Pro (thinking)
Gemma 4 26B-A4B-it	128 / 8	704 / 2816	dense MLP	6.8%	–	GSM8K (chat), GPQA-D, IFEval, MMLU-Pro

Residency. Resident memory is reported separately from reads; Table 5 lists each model's widths and sketch size. Beyond the unchanged model weights Ours w/o recovery keeps the ternary sketch (7.3% of the OLMoE BF16 checkpoint) and per-channel norms; Ours adds one weight-only bank that serves every budget (about 0.5%) and its score geometry. Prox keeps a 2.16 GB INT4 proxy on OLMoE (15.6%).

D INPUT-SPARSE SCREEN

At the sparsest budget the screen itself becomes the main cost: at P05 it is 7.8 of the reader's 13.5% of reads. A rotated input concentrates its energy in a few coordinates of each block, so the screen can read less of the sketch. For each token and block b, let $\mathcal{K}_b$ hold the 80 of 128 coordinates with the largest $|(Rx)_{b,c}|$; the screen then estimates

$$\hat{g} = \sum_b \alpha_b \sum_{c \in \mathcal{K}_b} t_{b,c}\,(Rx)_{b,c} \tag{6}$$

and likewise $\hat{v}$, reading only the codes of those coordinates, and ranks channels by $w_j^2 \|d_j\|^2 \phi(\hat{g}_j)^2 \hat{v}_j^2$. This reads about two thirds of the sketch, and the saved bytes buy more exact channels. Realizing the saving needs the sketch stored coordinate-major and a gather kernel we have not built; our kernels read the whole sketch (3.1 more points), so we report the extension at its analytical batch-one price with the whole-sketch price beside it and claim no speed for it. A batch reads the union of its tokens' coordinates. Prox's first stage prunes its proxy's inputs likewise, with calibrated thresholds.

E SCREEN ANALYSIS AND FURTHER ABLATIONS

Screens at equal bytes. Figure 4 measures each screen by the routed output energy $1 - \|y - y_{\mathcal{F}}\|^2/\|y\|^2$ that its top-F channels carry when computed exactly, on all 16 OLMoE layers at 3,000 held-out positions each, against its reads and its resident bytes (analytical logical proxy bytes including scales, residuals and down-column norms; not measured traffic). At P10, selection by exact activations carries 86.4%, our rotated ternary sketch 81.2% at 7.8% of routed bytes, the same sketch without rotation 80.2%, binary 76.3% (5.2%), 2-bit 83.2% (9.4%) and 4-bit 86.1% (17.2%); an INT4 gate screen in the style of HiRE carries 72.3% at 8.4%. Screens that read a low-bit proxy only at each token's largest input coordinates, as Prox's first stage does, keep the full proxy resident but read less: rotated ternary at 75% of coordinates carries 80.8% (0.4 below our sketch) at 5.6% of reads (7.3% resident), and rotated 2-bit 83.0% at 6.8% (8.9% resident), above our sketch in all 16 layers; they need a coordinate-major sketch layout, in which one coordinate's codes for an expert occupy 416–1,024 contiguous bytes. Earlier two-layer probes (layers 8 and 15) found exact values of one eighth of the coordinates capture only 57.8 / 70.3%, and because they touch 88% of the gate and up sectors their 19% logical reads cost about 70% once priced in sectors; a whitened per-expert rank-64 screen reaches 50.8 / 64.7% at 4.2%: per-channel detail is not low rank.

Energy, ranking and thresholds. Held-out KL follows the routed output energy that a selector's exact channels carry (Figure 6): across our sketch score, Prox's score at two input keeps and an INT4 gate screen, each at its four held-out budgets, the rank correlation is -0.96, and KL scales roughly as $(1-E)^{1.2\text{–}1.9}$. The gate-only screen sits above the common curve, since its errors fall on the channels that the up projection makes large. The same energy measurement separates the two axes of Table 7. For every score, a per-layer threshold fitted on the fitting split to keep a fraction f of channels carries less energy on held-out positions than one top-F with $F = fN$, while keeping 0.1–0.6% more channels on average (per-token standard deviation 3–10 points of f at P10); changing the score moves energy by up to 13 points. Calibration does not help any score; the ranking decides.

Sparse shared expert. Qwen3.5's always-active shared expert is 11% of its FFN bytes. Screening it too at P10 lowers whole-FFN reads from 27.7% to 18.9% and costs 1–2 points of zero-shot accuracy (67.8 without shared-expert recovery, 68.7 with a PCA map, against 69.8 routed-only); with the PCA map it matches the routed-only P05 plan, which needs 23.3% whole-FFN reads for 68.7. Our routed-read comparisons keep shared experts and dense MLPs stock.

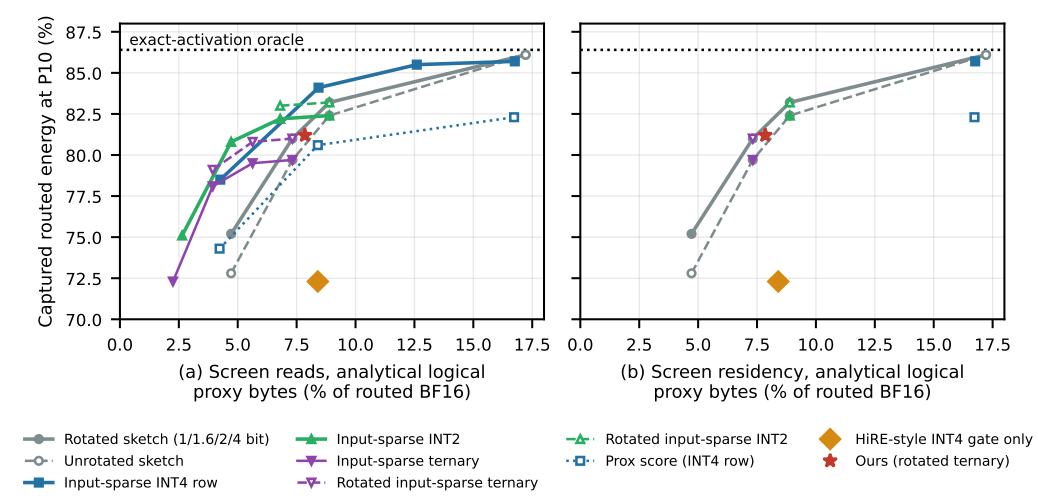

Figure 4: Channel screens on OLMoE at P10: routed output energy carried by each screen's top-F channels (16 layers, 3,000 held-out positions each) against (a) screen reads and (b) resident screen bytes, analytical logical proxy bytes in % of routed BF16 bytes. Input-sparse screens read each token's top 25, 50 or 75% of input coordinates and assume a coordinate-major layout; they keep the whole proxy resident at every keep, so (b) shows each at full input reads. Horizontal dotted line: selection by exact activations.

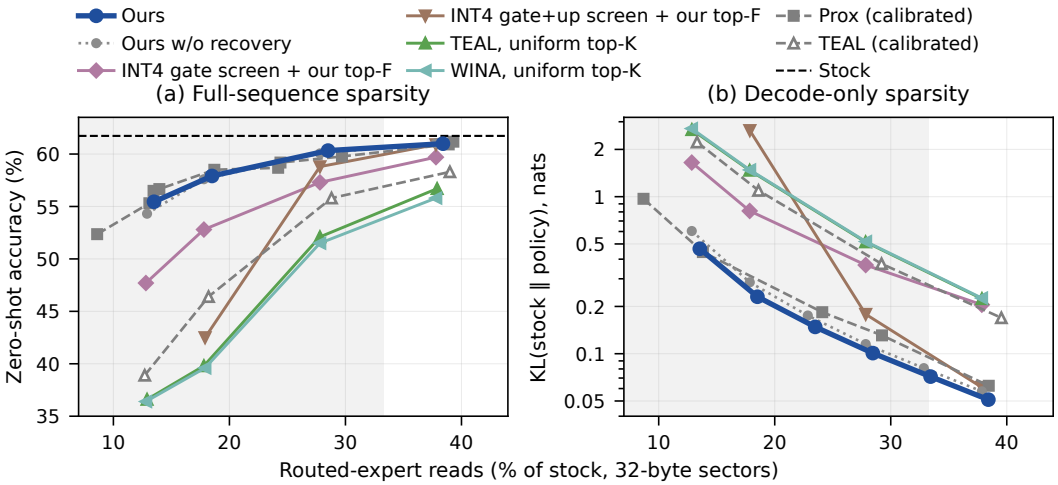

Figure 5: The calibration-free frontier on OLMoE-1B-7B. (a) Zero-shot accuracy under full-sequence sparsity; (b) KL divergence from stock under decode-only sparsity (development documents, log scale). Solid: selectors that use no calibration data (INT4 screens with our top-F; TEAL and WINA with a uniform per-token top-K); grey dashed: calibrated baselines at the reads they realize.

F Baseline recipes

All baselines are reimplemented in the same vLLM runtime as grouped Triton kernels that apply the method's mask to the routed experts; the router and its top-K are unchanged. Each baseline's thresholds are calibrated only on the fitting split of the common calibration data (Appendix G).

- **TEAL** (Liu et al., 2025a): magnitude thresholds on the inputs of the gate/up and down projections with the paper's greedy per-layer allocation (step 0.05, 512 states per layer); the development comparison also shows uniform TEAL, the same thresholds without the

Table 6: Ours, its ablations and calibration-free alternatives on OLMoE at P10: KL divergence from stock and top-1 agreement under decode-only sparsity (192 development documents), and zero-shot accuracy under full-sequence sparsity (Park's tasks). Reads and additional resident state in % of routed bytes; the design variants modify Ours w/o recovery. The input-sparse row (with the weight-only recovery) is priced at batch one; charged its whole sketch it reads 20.9%. Calibrated rows for reference; their thresholds realize slightly different reads on the zero-shot tasks (Prox 14.0 and 18.7%, TEAL 18.2%). Rotation seeds change KL by less than 0.003.

Variant	Data	Reads	Resident	KL	Top-1	Zero-shot
Ours (weight-only recovery)	none	18.5	8.4	0.231	0.784	57.9
Ours w/o recovery	none	17.8	7.8	0.286	0.768	57.6
without rotation	none	17.8	7.8	0.348	0.739	57.0
five rotation seeds	none	17.8	7.8	–	–	57.1 ± 0.5
per-expert quota	none	17.8	7.8	0.491	0.697	55.5
point estimate	none	17.8	7.8	0.265	0.777	57.2
Ours + input-sparse screen	none	17.8	8.4	0.194	0.808	58.6
INT4 gate screen, our top-F	none	17.8	8.6	0.811	0.619	52.8
INT4 gate and up screen, our top-F	none	17.9	17.2	2.640	0.367	42.5
TEAL, uniform top-K	none	17.8	0	1.471	0.547	39.8
WINA, uniform top-K	none	17.9	0	1.480	0.543	39.6
Calibrated recovery	calibrated	18.5	8.4	0.167	0.824	56.2
Prox, paper allocation	calibrated	13.8	15.6	0.444	0.721	56.7
Prox, best allocation	calibrated	18.4	15.6	0.277	0.777	58.5
TEAL	calibrated	18.6	0	1.095	0.619	46.4

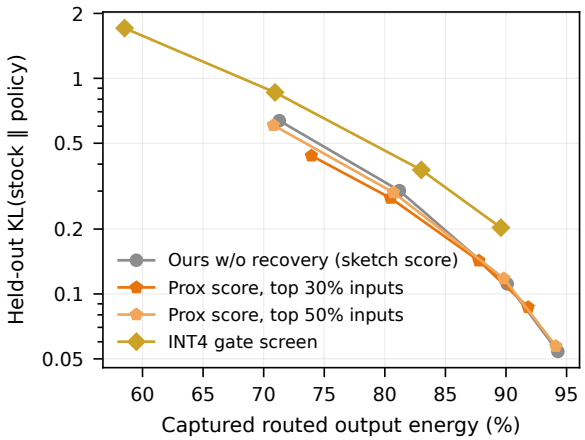

Figure 6: Held-out KL on OLMoE (192 documents, no recovery) against the routed output energy that the same selector's top-F channels carry at the same retention (16 layers, 3,000 held-out positions each). Markers: P05, P10, P20 and P30.

allocation. Read-matched budgets choose the target sparsity whose predicted read fraction is closest to each of our points; the 0.05 greedy step prevents an exact match.

- **WINA** (Chen et al., 2026b): weight-norm-scaled input magnitudes with uniform thresholds, without the optional orthogonal transform.
- **R-Sparse** (Zhang et al., 2025): input sparsity plus a low-rank term for the remaining inputs; rank and sparsity chosen by a mean-squared-error search on 64 fitting states per layer, calibrated so that its charged reads, low-rank factors and scales included, equal the target.
- **CATS** (Lee et al., 2024), **CHESS** (He et al., 2024) and **M-COUNTDOWN** (Cheon & Kang, 2025): thresholds on the dense gate (or up) activation at a target sparsity. CHESS uses its channel-wise thresholds; experts that the fitting data never route to fall back to the layer threshold.

Table 7: Routed output energy (%) carried on OLMoE at P10 and P30 (16 layers, 3,000 held-out positions each) by one top-F per token and by a per-layer score threshold fitted on the fitting split, for each score; Prox's scores read the INT4 proxy of the gate and up rows.

	P10		P30	
Score	top-F	threshold	top-F	threshold
Exact activations	86.4	–	97.3	–
Ours, rotated ternary sketch	81.2	79.6	94.3	93.7
Prox score, all inputs	82.3	81.6	95.6	95.5
Prox score, top 30% inputs	76.3	75.8	90.7	90.7
Output-unit score on the INT4 proxy	85.7	84.5	96.8	96.6
INT4 gate screen	72.3	71.4	90.0	89.8

- **Park et al.** (Park et al., 2026): post-activation magnitude thresholds. We evaluate the global and per-layer variants and report per-layer absolute thresholds, which are best at 40–48% reads. Park and CHESS are reported at the target sparsity whose development reads are closest to our P30 point.
- **Prox** (Liu et al., 2026): an INT4 proxy of the gate and up weights scores each channel, and the selected channels are computed exactly. Calibration is sequential: layer ℓ's thresholds are fitted on inputs produced by the already-sparsified layers $< \ell$. We replace the reference single-batch kernel with a grouped proxy kernel that produces identical scores (no score differences or mask flips over 4.19M real scores) and identical outputs. Besides the paper's allocation between its input-pruning first stage and its channel stage, we sweep the first-stage input sparsity over $\{0.5, 0.7, 0.85, 0.95\}$ at total read targets of about 13, 18 and 38% on OLMoE, each calibrated with the paper's recipe, and keep the allocation with the lowest development KL, a rule fixed before the held-out panel was scored (0.7 at 13 and 18%, 0.5 at 38%): the *best allocation*.
- **Halving** (Chen & Yao, 2026): activates $k < K$ experts per token in two stages; the second-stage count is chosen on MMLU's validation split, as in its paper.

Table 8: Reads (% of routed bytes) that the same calibrated thresholds realize on held-out text from their calibration distribution, on the benchmarks, and at token depths 0 and 6,000 of teacher-forced GPQA-Diamond reasoning traces. Ours reads its budget on every input (P10: 18.4–18.7%).

Model	Method	Calibration text	GPQA-D	IFEval	MMLU-Pro	Trace, 0	Trace, 6k
Gemma 4	Prox	17.3	13.3	14.2	–	15.1	13.8
	TEAL	17.3	15.2	–	–	–	–
Qwen3-Thinking	Prox	15.5	9.9	10.7	11.2	15.6	10.6
Qwen3.5	Prox	15.7	14.2	14.7	14.5	–	–

G CALIBRATION DATA AND THE CALIBRATED RECOVERY

Ours, recovery included, uses no data. Calibration data serve the baselines' thresholds and one reference: the recovery map of Eq. (3) fitted to the stock model's output distribution, which measures what data add to recovery (Table 9).

Data. For OLMoE we use Pile documents: 177 fitting and 40 tuning documents; the 192 development documents are disjoint from both. For the post-trained models we draw UltraChat-200k `train_sft` conversations, split 7:1:2 into disjoint pools with a fixed seed, and take 448 fitting, 64 tuning and 128 development conversations. Each conversation is rendered with the provider's chat template in its non-thinking form, truncated to 2,048 tokens, and contributes one 128-token window inside an assistant turn. No benchmark data are used. The screen, recovery rank and plans were developed on the OLMoE development documents; every calibrated bank's checkpoint is chosen on its tuning split; the development sets are never used for fitting but were used during method development, so the fidelity measurements on them are not confirmatory. P10 and P30 were fixed by cost before any benchmark result.

Calibration domain. To test whether our choice of calibration text handicaps the calibrated baselines, we recalibrate Prox on OLMoE (its paper's allocation, five read targets) on Alpaca instructions and on UltraChat conversations instead of Pile, with the same recipe, and evaluate each on the Pile development documents, Park's zero-shot tasks and BIG-Bench Hard. The calibration text shifts the realized reads (at the read-15 target 13.8, 13.6 and 13.3% on the development documents) and barely moves quality: at matched reads the KL of the three calibrations differs by less than 5% after log-linear read matching, zero-shot accuracy averaged over the five targets changes by -0.25 ± 0.16 (Alpaca) and -0.47 ± 0.17 (chat) points against Pile, and BIG-Bench Hard at the read-15 target by -0.77 ± 0.54 and -0.24 ± 0.56. These OLMoE results show little sensitivity to which of the three corpora is used, so our choice of calibration text does not handicap Prox.

Fitting. One calibrated bank is fitted per read budget. The fitter runs a differentiable Transformers implementation of the model with the reader's exact membership rule, sketch features and recovery. A stock prefix is prefilled, and the KL divergence (Hinton et al., 2015) from the stock next-token distribution is summed over the following 64 tokens. U starts from the top-r principal directions of the omitted output on calibration states with $b_j = U^\top d_j$; we run 128 Adam updates (learning rate 10^{-3}, 8 documents per update), recompute memberships with the current bank every four updates, since the score depends on the bank, and check the tuning KL every 8 updates. For large models the decoder layers are split across two GPUs; for Qwen3.5 the gated-delta attention layers use the `fla` kernels (at most 0.55% relative gradient difference from the reference implementation).

What data add. Fitting sharpens fidelity where the stock prefix it was fitted after is present: under decode-only sparsity on OLMoE the calibrated map removes 42–52% of the KL of Ours w/o recovery at P05 and P10 (22% at P30), against 12–23% for the weight-only map of Ours (Table 9). Under full-sequence sparsity it meets sparse context it was not fitted on and scores below Ours w/o recovery from P10 on, while Ours does not. On long reasoning the two maps are level on Gemma 4 and Qwen3.5 and the weight-only map is ahead on Qwen3-Thinking (66.8 against 62.8).

Calibration exposure. Fitting the calibrated OLMoE P10 bank with the prefix sparse as well (teacher stock, all else unchanged) raises the zero-shot accuracy from 56.5 to 57.2 at 18.5% reads (paired SE 0.27; the control is a rerun of the stock-prefix P10 arm, which scores 56.2 in Table 9, 56.4 in a paired rerun and 56.5 here, the run-to-run spread from batch composition) and leaves stock-prefill quality unchanged or better (development NLL -0.003 ± 0.002, WikiText NLL -0.018 ± 0.001); one seed, not converged at 128 updates.

Table 9: What calibration data add to recovery, at the reads of Ours at each plan (Ours w/o recovery reads 0.6–0.8 points less). Ours: the weight-only map; calibrated: the same map fitted per budget to the stock model's output distribution on calibration text. KL: decode-only fidelity to stock on the development panels; zero-shot: full-sequence sparsity (Park's tasks); BBH and GPQA-D (avg@4): decode-only.

Model	Metric	Plans	Ours w/o recovery	Ours	Calibrated recovery
OLMoE-1B-7B	KL	P05 / P10 / P30	0.604 / 0.285 / 0.058	0.466 / 0.231 / 0.051	0.288 / 0.167 / 0.045
	Zero-shot	P05 / P10 / P30	54.1 / 57.6 / 61.0	55.5 / 57.9 / 61.0	54.3 / 56.2 / 60.4
	BBH CoT	P05 / P10	21.7 / 30.1	26.0 / 30.3	28.0 / 29.7
Qwen3.5-35B-A3B	KL	P10 / P30	0.063 / 0.015	0.057 / 0.014	0.047 / 0.013
	Zero-shot	P05 / P10 / P30	68.7 / 69.8 / 70.8	69.3 / 69.9 / 70.7	68.7 / 69.9 / 70.4
	GPQA-D	P10	83.2	82.6	83.7
Qwen3-30B-A3B-Thinking	KL	P10 / P30	0.134 / 0.028	0.116 / 0.025	0.076 / 0.020
	GPQA-D	P10	61.5	66.8	62.8
Gemma 4 26B-A4B-it	KL	P10 / P30	0.217 / 0.035	0.122 / 0.027	0.099 / 0.026
	GPQA-D	P10	67.9	73.7	72.2

H PROTOCOLS

Harnesses. Likelihood tasks use the LM Evaluation Harness (v0.4.3) task definitions, scored in vLLM from prompt log-probabilities. Park's protocol is zero-shot throughout; Prox's uses 5-shot MMLU, 10-shot HellaSwag, 0-shot ARC-Easy, 25-shot ARC-Challenge and 0-shot PIQA. GPQA-Diamond draws four samples per question with the card's sampling; IFEval is scored with the har-

ness's official checker (we report prompt-level strict accuracy; all four IFEval metrics are recorded); the MMLU-Pro subset is a fixed stratified sample of 1,000 test questions. Thinking-mode sampling follows each card: Gemma 4 temperature 1.0, top-p 0.95, top-k 64; Qwen3-Thinking 0.6, 0.95, 20, min-p 0; Qwen3.5 1.0, 0.95, 20, min-p 0 with presence penalty 1.5. The presence penalty that Qwen3-Thinking's card offers as an option against endless repetition is not used. Output caps: Qwen3-Thinking's card sets 32,768 tokens outside competition math and programming (its stock GPQA run used 81,920, but no stock answer exceeds 23,432 tokens, so the cap does not bind); Qwen3.5's card reserves 81,920 for highly complex problems, which we use for GPQA-Diamond (3.5% of stock answers exceed 32,768) and 32,768 elsewhere; Gemma 4's card gives no cap, and we use 32,768 (0.8% of stock GPQA answers reach it). The OLMoE suite uses OLMES task definitions, including its 8-shot chain-of-thought GSM8K. GSM8K under the Prox protocol uses the harness's `gsm8k_cot` task (8-shot chain of thought) with greedy decoding and 256 new tokens; for the post-trained chat models the prompt is rendered with the model's chat template, each shot as a user/assistant turn (the harness's multi-turn few-shot option) and reasoning disabled. Rendered as raw text, stock Qwen3.5 opens a reasoning block in every answer and 756 of 1,319 answers reach the token limit (26.5 strict), whereas the chat rendering scores 88.9; Gemma 4 needs the beginning-of-sequence token its tokenizer does not add (GSM8K 91 versus 67 on 100 questions), and its raw-text likelihood scores stay far below its chat-format quality, so it is evaluated on generation only. These format decisions were made on stock results alone. Thinking-mode tasks use each model card's chat template, sampling and output cap; answers are extracted from the text after the reasoning block.

Sparse prefill for evaluation. The Park and Prox protocols sparsify every token, including prompt tokens. Every compared policy decides from the current token's hidden state at the current layer only, and attention is causal. Hence the hidden state of position t at layer ℓ depends only on positions $\leq t$ at layer $\ell - 1$, each already computed under the same sparse policy, whether the prompt is processed in one prefill pass or token by token as forced decode steps. The two procedures compute the same function in exact arithmetic and differ only in batch-shape-dependent floating-point rounding. We therefore apply the policy inside vLLM prefill for these protocols. Measured on OLMoE, sparse prefill and replay with a sparse lead differ by no more than each policy's own replay gap between batch sizes 64 and 16 (mean per-token $|\Delta \log p|$: stock 0.028 against 0.036, our reader at P10 0.129 against 0.132, Prox 0.364 against 0.362). Deployment and all speed measurements keep stock prefill with sparse decode.

Parallel layouts. Qwen3 and Gemma 4 run either on two GPUs (tensor-parallel attention, expert-parallel experts) or on one 96 GB GPU. The two layouts compute the same function up to floating-point reduction order: on 1,026 Qwen3.5 MMLU questions the answers that change between layouts are no more frequent than those that change between batch sizes (McNemar $p = 0.27$, stock and sparse).

TEAL-style likelihood (OLMoE). The OLMoE likelihood table keeps the first half of each request stock and sparsifies the second half, following TEAL's evaluation, for every method.

I FIDELITY TO STOCK

Fidelity carries the paper's claim (Section 6.2); these tables give every measured point. The primary OLMoE panel (Table 10) is held out: 192 Pile documents from rows 5,000–10,000, disjoint from the calibration and development documents, scored after the Prox allocation rule was fixed. On it the input-sparse extension has lower KL than every other point at each budget; Ours is level with Prox at P05 (Prox's points at 13.2 and 13.7% give 0.489 and 0.462 around Ours' 0.488 at 13.5%) and below Prox from P10 up (Prox interpolated to 18.5%: 0.29 against 0.245; at P20 Ours reads fewer bytes than Prox's 29.1% point and has lower KL; at P30 Prox's points at 38.3 and 38.7% bracket Ours with KL 0.057–0.060 against 0.047). Ours w/o recovery trails Prox at P05 (Prox interpolated to 12.9%: 0.53 against 0.637), is level at P10 and has lower KL at P30 with fewer reads. Every baseline other than Prox is farther from stock than all three variants of ours at every budget where both were measured. The post-trained panels (Table 11) are the development conversations.

Between measured points we interpolate log-linearly in KL, as the figures draw the lines. On the post-trained panels the input-sparse extension has lower KL than Prox at P05 and P10 (Gemma 4 at

Table 10: Fidelity to stock on OLMoE under decode-only sparsity: KL divergence from stock to the policy over the full vocabulary, top-1 agreement and the increase in next-token NLL over stock (1.874), at the reads each method realizes. Held out: 192 Pile documents disjoint from calibration and development (12,096 decode positions); development: the 192 documents used during method development. Input-sparse reads are the batch-one price, whole-sketch reads in parentheses. Prox best: the first-stage allocation with the lowest development KL (Appendix F); Prox paper: the paper's allocation. Lower KL and NLL, higher agreement are better.

		Held out				Development		
Budget	Method	Reads	KL	Top-1	ΔNLL	Reads	KL	Top-1
∼13%	Ours (P05)	13.5	0.488	0.699	0.469	13.5	0.466	0.703
	Ours w/o recovery (P05)	12.9	0.637	0.668	0.613	12.9	0.604	0.665
	Ours + input-sparse screen (P05)	12.8 (15.9)	0.334	0.751	0.317	12.8	0.316	0.755
	Prox paper	8.6	1.018	0.594	0.994	8.7	0.969	0.599
	Prox best	13.2	0.489	0.709	0.465	13.3	0.471	0.713
	Prox paper	13.7	0.462	0.723	0.444	13.8	0.444	0.721
	TEAL	–	–	–	–	13.3	2.226	0.465
	HiRE-style gate screen	12.8	1.709	0.469	1.691	12.8	1.649	0.482
∼18%	Ours (P10)	18.5	0.245	0.782	0.235	18.5	0.231	0.784
	Ours w/o recovery (P10)	17.8	0.301	0.764	0.284	17.8	0.286	0.768
	Ours + input-sparse screen (P10)	17.8 (20.9)	0.199	0.808	0.184	17.8	0.194	0.808
	Prox best	18.3	0.298	0.774	0.280	18.4	0.277	0.777
	TEAL	18.5	1.109	0.613	1.097	18.6	1.095	0.619
	WINA	17.6	1.295	0.583	1.261	17.7	1.289	0.581
	R-Sparse	18.1	1.132	0.577	1.095	18.2	1.158	0.575
	HiRE-style gate screen	17.8	0.860	0.602	0.843	17.8	0.811	0.619
∼28%	Ours (P20)	28.5	0.096	0.860	0.085	28.5	0.101	0.855
	Ours w/o recovery (P20)	27.8	0.111	0.852	0.102	27.8	0.115	0.848
	Ours + input-sparse screen (P20)	27.8 (30.9)	0.095	0.864	0.088	27.8	0.098	0.862
	Prox paper	24.0	0.192	0.813	0.178	24.1	0.184	0.820
	Prox paper	29.1	0.136	0.842	0.122	29.3	0.131	0.842
	TEAL	29.1	0.378	0.760	0.351	29.2	0.374	0.759
	WINA	28.5	0.461	0.731	0.434	28.6	0.457	0.735
	R-Sparse	28.8	0.445	0.717	0.417	29.0	0.450	0.728
	HiRE-style gate screen	27.8	0.376	0.720	0.359	27.8	0.367	0.723
∼40%	Ours (P30)	38.4	0.047	0.900	0.042	38.4	0.051	0.901
	Ours w/o recovery (P30)	37.9	0.054	0.895	0.049	37.9	0.058	0.891
	Ours + input-sparse screen (P30)	37.8 (40.9)	0.054	0.895	0.047	–	–	–
	Prox paper	38.3	0.060	0.892	0.048	38.5	0.063	0.887
	Prox best	38.7	0.057	0.891	0.048	38.8	0.059	0.895
	TEAL	–	–	–	–	39.5	0.169	0.830
	WINA	–	–	–	–	39.5	0.197	0.814
	CHESS	40.4	0.643	0.663	0.625	40.4	0.621	0.670
	CATS	40.5	1.113	0.555	1.090	40.6	1.064	0.569
	M-COUNTDOWN	40.9	1.123	0.612	1.097	40.9	1.140	0.607
	Park (layer)	40.4	1.120	0.557	1.096	40.4	1.077	0.569
	HiRE-style gate screen	37.8	0.203	0.783	0.196	37.8	0.205	0.787

12.8% against Prox at 17.3%: 0.176 against 0.211; Qwen3-Thinking interpolated to 15.5%: 0.120 against 0.154; Qwen3.5 against Prox interpolated to 12.8 and 17.9%: 0.081 against 0.090, 0.051 against 0.058); at P30 on Qwen3.5 Prox dominates it (0.016 at 35.7% against 0.017 at 37.9%). Ours has lower KL than Prox interpolated to its P10 and P20 reads on Gemma 4 (0.122 against 0.193, 0.052 against 0.089), is level on the Qwen models at P10 (Qwen3-Thinking 0.116 against 0.121, Qwen3.5 0.057 against 0.055) and at P20 on Qwen3.5 (0.026 against 0.027), and trails Prox below 16% on both (Qwen3-Thinking interpolated to 15.5%: 0.172 against 0.154; Qwen3.5 at 13.7%: 0.099 against 0.082). Without recovery the reader is level with Prox on Gemma 4 at P10 (0.217 at

Table 11: Fidelity to stock on the post-trained models (128 development UltraChat conversations, chat template, sparse decoding): KL divergence from stock over the full vocabulary and top-1 agreement, at the reads each method realizes. Input-sparse reads are the batch-one price (whole sketch: 3.1 points more).

	Gemma 4 26B-A4B-it			Qwen3-30B-A3B-Thinking			Qwen3.5-35B-A3B		
Method	Reads	KL	Top-1	Reads	KL	Top-1	Reads	KL	Top-1
Ours (P05)	13.4	0.235	0.855	13.6	0.220	0.843	13.7	0.099	0.887
Ours (P10)	18.4	0.122	0.893	18.6	0.116	0.881	18.7	0.057	0.914
Ours (P20)	28.4	0.052	0.930	28.5	0.049	0.920	28.6	0.026	0.939
Ours (P30)	38.4	0.027	0.950	38.5	0.025	0.943	38.5	0.014	0.954
Ours w/o recovery (P05)	12.8	0.441	0.798	12.8	0.251	0.831	12.9	0.110	0.879
Ours w/o recovery (P10)	17.8	0.217	0.858	17.8	0.134	0.870	17.9	0.063	0.907
Ours w/o recovery (P20)	27.8	0.077	0.914	27.9	0.057	0.911	27.8	0.029	0.937
Ours w/o recovery (P30)	37.8	0.035	0.940	37.8	0.028	0.937	37.9	0.015	0.954
Ours + input-sparse screen (P05)	12.8	0.176	0.871	12.8	0.157	0.865	12.8	0.081	0.898
Ours + input-sparse screen (P10)	17.8	0.106	0.900	17.8	0.096	0.892	17.9	0.051	0.917
Prox	–	–	–	–	–	–	10.4	0.115	0.877
Prox	17.3	0.211	0.860	15.5	0.154	0.863	15.7	0.067	0.906
Prox	–	–	–	–	–	–	25.6	0.034	0.934
Prox	37.3	0.044	0.935	35.4	0.032	0.933	35.7	0.016	0.953
TEAL	12.8	1.270	0.628	13.4	1.383	0.628	13.9	0.555	0.741
TEAL	17.3	0.780	0.710	18.1	0.837	0.718	19.2	0.296	0.806
TEAL	27.9	0.286	0.830	28.3	0.287	0.827	28.2	0.126	0.868
TEAL	38.8	0.108	0.893	38.5	0.099	0.888	39.2	0.044	0.921

17.8% against 0.211 at 17.3%) and behind it on the Qwen models. At P30 Prox's points lie 1–3 points of reads below ours with KL above ours, so no ordering is claimed there.

J SHORT ANSWERS

Short answers are a floor test: they expose selectors that break the model but barely separate the strong ones (Table 12). On Qwen3.5, Ours (18.7%), Ours w/o recovery (17.9%), Prox (9.8 and 14.4%), TEAL (19.5%) and WINA (15.4%) stay within 1.3 points of stock, each changing 5–10% of the answers in both directions. The methods that fall behind are those that fail elsewhere: gate thresholds lose 3–5 points at 38% of reads (CATS, CHESS), and expert-count reduction to one expert loses 43 (Halving, k=1). In thinking mode on Qwen3-Thinking every listed method stays within 2.1 points of stock. Gemma 4 is the one model where short answers separate the strong selectors: without recovery the reader loses 9.9 points (164 answers turn wrong), Ours and Prox stay within a point of stock (83.4 at 18.4%, 83.1 at 15.7%), and TEAL at 15.7% loses 35. The paper therefore leads with fidelity and long reasoning.

K LONG REASONING

Matched reads. Repetition occurs without compression (Xu et al., 2022; Pipis et al., 2025; Lazaridis et al., 2026), and pruning amplifies it and lengthens reasoning (Lee et al., 2026; Lotfi et al., 2026; Lian et al., 2026; Lucas et al., 2025). Table 13 compares the methods on GPQA-Diamond at matched reads. Prox's thresholds realize fewer reads on reasoning text than on chat text, so further Prox calibrations at higher read targets place its realized GPQA reads around ours. The comparison uses the first sample of each question (198 answers per arm, paired by question and seed). A point counts as collapsed if its accuracy is below 30 or more than 50% of its samples reach the output cap; this rule was fixed before these results. On Gemma 4, Prox's points at 13.3 and 14.0% collapse and its 15.5% point scores 45.5, while Ours at P05 scores 51.0 at 13.4% (30.8 $\pm$ 4.1 points above Prox at 14.0%, paired) and the input-sparse extension scores 69.7 at 12.8%, above every Prox point at fewer reads; Ours w/o recovery collapses at P05. On Qwen3-Thinking Prox is level with Ours

Table 12: GSM8K under decode-only sparsity: lm-eval `gsm8k_cot` 8-shot in the chat template with reasoning disabled (Qwen3.5, Gemma 4) and zero-shot in thinking mode (Qwen3-Thinking), strict match. Changed: answers that turn wrong / right against stock (1,319 questions). Thresholded baselines at the reads they realize.

Model	Method	Reads (%)	GSM8K	Changed
Qwen3.5-35B-A3B	Stock	100	88.9	–
	Ours (P10)	18.7	89.8	27 / 39
	Ours w/o recovery (P10)	17.9	88.6	36 / 33
	Calibrated recovery (P05 / P10 / P30)	13.7 / 18.7 / 38.5	88.5 / 89.7 / 89.0	55/50, 27/38, 23/25
	Prox	9.8 / 14.4 / 32.8	88.9 / 89.8 / 89.8	37/37, 24/37, 23/35
	TEAL	14.6 / 19.5 / 38.4	85.7 / 89.5 / 89.1	97/56, 48/56, 29/32
	WINA	15.4 / 36.4	87.6 / 90.4	70/53, 26/46
	CHESS	37.9	85.5	87 / 43
	CATS	37.6	83.8	101 / 34
	Halving (k=1, 2, 3)	12.5 / 25.0 / 37.5	45.5 / 83.2 / 90.0	599/27, 106/31, 30/45
Qwen3-30B-A3B-Thinking	Stock	100	96.7	–
	Ours (P10)	18.6	96.2	16 / 10
	Ours w/o recovery (P10)	17.8	95.4	25 / 8
	Calibrated recovery (P10)	18.6	95.7	19 / 6
	Prox	11.3	94.6	35 / 8
Gemma 4 26B-A4B-it	Stock	100	83.9	–
	Ours (P10)	18.4	83.4	58 / 52
	Ours w/o recovery (P10)	17.8	74.0	164 / 34
	Prox	15.7	83.1	71 / 61
	TEAL	15.7	49.1	490 / 31

at P10 (68.2 at 17.9% against 66.7 at 18.6%; difference -1.5 ± 2.7) and reaches Ours' P05 score (62.6) at fewer reads (11.1 against 13.6%): there Prox dominates Ours at P05, while the input-sparse extension scores 64.6 at 12.8%, between Prox's 11.1 and 17.9% points.

Table 13: GPQA-Diamond at matched reads, first sample of each question (198 answers per arm; card sampling and caps). Capped: % of answers that reach the output cap. Input-sparse reads are the batch-one price. The collapse rule (accuracy below 30 or more than 50% capped) was fixed before these results.

Model	Method	Reads	GPQA-D	Capped
Gemma 4	Stock	100	80.3	1.0
	Ours (P05)	13.4	51.0	31.8
	Ours (P10)	18.4	73.2	2.5
	Ours w/o recovery (P05)	12.8	3.5	94.9
	Ours w/o recovery (P10)	17.8	64.6	9.1
	Ours + input-sparse screen (P05)	12.8	69.7	4.5
	Prox	13.3	11.6	84.8
	Prox	14.0	20.2	72.2
	Prox	15.5	45.5	31.8
	Prox	32.3	76.8	0.5
	TEAL	15.2	0.0	100.0
Qwen3-Thinking	Stock	100	69.7	0.0
	Ours (P05)	13.6	62.6	0.5
	Ours (P10)	18.6	66.7	0.0
	Ours w/o recovery (P05)	12.8	57.1	4.0
	Ours w/o recovery (P10)	17.8	60.1	0.0
	Ours + input-sparse screen (P05)	12.8	64.6	0.0
	Prox	9.9	58.6	4.0
	Prox	11.1	62.6	0.5
	Prox	17.9	68.2	0.0
	Prox	28.4	66.7	0.0

How answers end. Table 14 splits all four samples per question by how they end. A sample is capped when it reaches the output cap (32,768 tokens; 81,920 for Qwen3.5) and loops when, in addition, from some line on every following window of 3,000 characters has at least 60% of its

characters in lines that already occurred earlier in the same sample. The looping text is fluent and on topic: TEAL's samples restate the question correctly, contain at most 6×10^{-5} malformed characters, and start repeating at a median of 162 generated tokens (Prox: 6,406). The thresholds of Prox and TEAL realize 17.3% of reads on Gemma 4 chat text and 13.3% and 15.2% here, mostly on looping text; our read count is fixed per token. On Gemma 4 loops account for most of the lost accuracy, while the accuracy of completed samples moves much less; the Qwen models rarely loop. Sampling follows each card: Gemma 4 and Qwen3-Thinking use no repetition penalty, whereas Qwen3.5's card adds a presence penalty of 1.5, so its absence of loops is not comparable with the other two models. Table 14 is a failure analysis at each method's P10 or read-18 setting, not a matched-read comparison.

Validity controls. Two controls check the dense limit and preemption sensitivity. With every threshold set to keep all channels, Prox and TEAL match stock on the Gemma 4 chat panel as closely as a second stock run does (KL 8.8×10^{-4} and 9.0×10^{-4} against 6.1×10^{-4}), and Prox's grouped proxy kernel reproduces the reference kernel's scores and masks (Appendix F). Rerun without preemption (first sample per question, identical seeds), Prox scores 8.0 on questions 1–25 with 92% of samples looping, agreeing with the full run's correctness on 23 of 25, and TEAL scores 0.0 on questions 1–18, while stock (90.0) and Ours w/o recovery (77.5, 5% looping) stay healthy on questions 1–40.

Table 14: How GPQA-Diamond samples end (four per question, 792 in total), at P10 unless noted. Capped and loops: percent of samples; completed: accuracy over samples that end before the cap; repeated lines: median share of a sample's characters in lines that already occurred in it. Reads differ between rows.

Model	Method	Reads	GPQA-D	Capped	Loops	Completed	Repeated lines
Gemma 4	Stock	100	79.9	0.8	0.1	80.5	0.02
	Ours	18.4	73.7	2.9	2.4	75.9	0.05
	Ours w/o recovery (P05)	12.8	2.8	96.1	93.8	71.0	0.83
	Ours w/o recovery	17.8	67.9	7.7	7.7	73.6	0.07
	Ours w/o recovery (P30)	37.8	79.4	0.3	0.0	79.6	0.03
	Calibrated recovery	18.4	72.2	1.3	0.5	73.2	0.04
	Prox	13.3	11.7	85.1	84.7	78.8	0.78
	TEAL	15.2	0.0	99.7	94.2	–	0.97
Qwen3-Thinking	Stock	100	70.8	0.0	0.0	70.8	0.00
	Ours	18.6	66.8	0.0	0.0	66.8	0.00
	Ours w/o recovery (P05)	12.8	54.5	3.5	0.5	56.5	0.00
	Ours w/o recovery	17.8	61.5	0.1	0.0	61.6	0.00
	Calibrated recovery	18.6	62.8	0.3	0.0	62.9	0.00
	Prox	9.9	59.7	2.9	2.4	61.5	0.00
Qwen3.5	Stock	100	83.6	0.0	0.0	83.6	0.00
	Ours	18.7	82.6	0.6	0.6	83.1	0.01
	Ours w/o recovery	17.9	83.2	0.0	0.0	83.2	0.00
	Calibrated recovery	18.7	83.7	0.8	0.8	84.4	0.00
	Prox	14.2	83.1	0.1	0.1	83.2	0.00

L FULL-SEQUENCE SPARSITY PER TASK

Several baselines report results with the prompt sparse too; on their own protocols, with every token sparse on Park's zero-shot tasks, the reader trails Prox by at most a point and leads every other method widely (Figure 2(c); Qwen3.5 in Figure 7 below). On OLMoE Prox leads by 0.5–1.0 points at 13–18% (56.5 against our 55.5 at 13.5%; 58.4 interpolated against 57.9 at 18.5%), the extension matches or dominates it (56.5 at 12.8%, 58.6 at 17.8%), and from 28% the two are level; TEAL, WINA and R-Sparse trail by 11–14 points at 18%. On Qwen3.5 our reproduction of Park et al. (2026) matches its published operating point (67.6 against 67.5 at 38% of routed reads), and our reader exceeds it with 36% of the reads (69.3 at 13.7%); Prox reaches our P05, P10 and P30 scores with 3–5 fewer points of reads, needs 3 more at P20 (interpolated), and on its own few-shot protocol leads MMLU by about two points at our P10 reads (Tables 16 and 17).

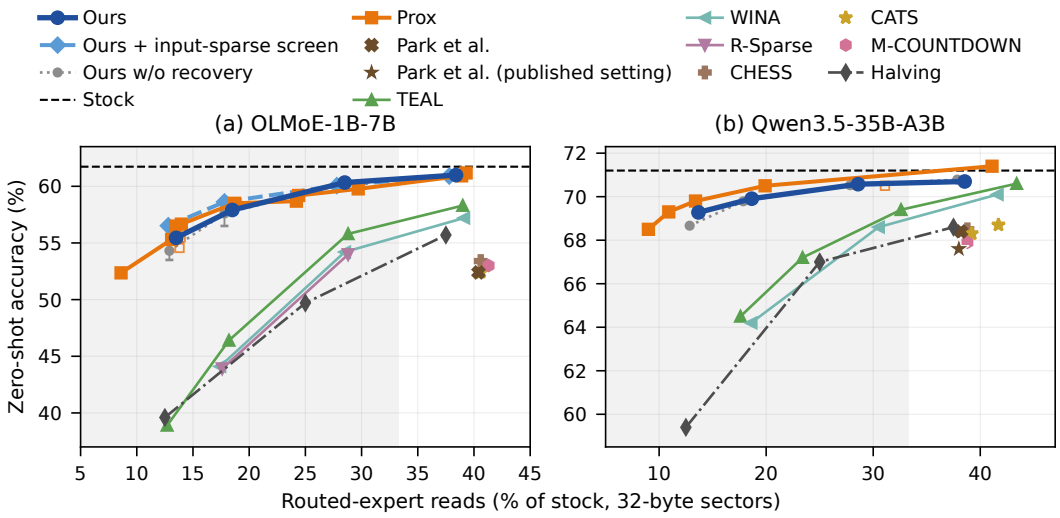

Figure 7: Full-sequence sparsity (prompt and answer sparse): zero-shot accuracy, the mean of the five tasks of Park et al. (2026) (ARC-Challenge and HellaSwag normalized, ARC-Easy, WinoGrande, TruthfulQA-mc2; lm-eval 0.4.3), versus sector-priced routed-expert reads. Lines and markers as in Figure 2; star: our run of Park et al.'s published setting (global thresholds). Per-task scores in Table 15.

Under the baselines' own protocols (Section 6.2) every token is sparse, the prompt included. Tables 15 and 16 give the per-task scores behind the zero-shot averages: HellaSwag and ARC-Challenge separate the methods most, while on OLMoE TruthfulQA rises as the other tasks fall, for every method. On OLMoE Prox's best allocation is ahead of Ours by 0.5–1.0 points at P05 and P10 (56.5 at 13.5% against 55.5 at 13.5%; 58.5 at 18.7% against 57.9 at 18.5%) and level from P20 up (61.2 at 39.3% against 61.0 at 38.4%); the input-sparse extension matches it at fewer reads (56.5 at 12.8%, 58.6 at 17.8%). On Qwen3.5 Prox reaches the scores of Ours with 3–5 fewer points of reads at P05, P10 and P30 and needs about 3 more at P20 (interpolated). Under Prox's few-shot protocol on Qwen3.5 (Table 17) Prox scores higher on MMLU than Ours at fewer reads (73.4 at 13.3% against 72.7 at 18.7%); on the other tasks the points differ by at most 1.5 at unequal reads.

M OLMoE TASK QUALITY

Table 18 extends the comparison to standard OLMoE tasks. Ours scores above Prox's interpolated curve on BBH and OLMo 2 generation at every plan where Prox's points bracket its reads (P05–P25); on GSM8K (SE about 0.9) Prox is ahead below P30; on WikiText and likelihood Prox is ahead at P05 and level or behind from P10 up. Ours w/o recovery trails Prox at P05 on every task (BBH 21.69 at 12.9% against 23.66 at 12.4%). The input-sparse extension scores above every Prox point at matched reads except on likelihood and GSM8K near 38%.

N SPEED PER BATCH SIZE

Table 19 gives the measured speedup at every batch size; the main text carries a compact version. Setup: one RTX PRO 6000 Blackwell GPU, vLLM 0.29 with CUDA graphs, real chat decode states, plan P10, three processes per configuration. Stock uses vLLM's fused-MoE Triton kernels tuned with vLLM's own tuner, and at each batch size the reference is the fastest of every stock configuration we measured. The MoE block time covers the routed experts and, for our reader, the screen, selection, completion and recovery; end-to-end decode is the time per output token (TPOT) of the whole model with stock prefill. The input-sparse extension runs on the same kernels, which read the whole sketch, so its row measures the extra selection work, not a read saving. Recovery costs 0.5–3.5% against Ours w/o recovery.

Table 15: Per-task accuracy under full-sequence sparsity on Park et al.'s zero-shot tasks (lm-eval 0.4.3), with threshold baselines at the reads they realize on this task set. ARC-C and HellaSwag report acc_norm, ARC-E and WinoGrande acc, TruthfulQA mc2; Avg is their mean, the zero-shot accuracy. Input-sparse reads are the batch-one price. Prox best: the first-stage allocation chosen on development KL (Appendix F). Qwen3-Thinking and Gemma 4 are evaluated on generation only (Section 5).

Model	Method	Reads (%)	ARC-C	ARC-E	HellaSwag	WinoGr.	TQA	Avg
OLMoE-1B-7B	Stock	100	48.7	78.1	76.9	69.0	35.9	61.7
	Ours (P05)	13.5	41.8	68.2	69.3	60.1	37.8	55.5
	Ours (P10)	18.5	44.5	73.1	73.2	63.3	35.5	57.9
	Ours (P20)	28.5	48.2	76.1	76.2	65.3	35.8	60.3
	Ours (P30)	38.4	49.4	77.4	77.0	66.0	35.2	61.0
	Ours w/o recovery (P05)	12.9	41.2	64.9	67.3	58.2	38.9	54.1
	Ours w/o recovery (P10)	17.8	44.5	71.4	72.9	62.9	36.1	57.6
	Ours w/o recovery (P20)	27.8	47.4	75.5	76.2	65.7	35.5	60.1
	Ours w/o recovery (P30)	37.9	49.6	77.2	77.1	66.1	35.1	61.0
	Ours + input-sparse screen (P05)	12.8	43.6	70.7	71.3	61.2	35.9	56.5
	Ours + input-sparse screen (P10)	17.8	46.2	74.3	74.3	63.8	34.6	58.6
	Ours + input-sparse screen (P20)	27.8	47.4	75.8	76.0	65.6	35.5	60.1
	Ours + input-sparse screen (P30)	37.8	48.5	77.4	76.8	66.8	35.0	60.9
	Prox paper	8.6	38.2	63.1	62.1	59.0	39.4	52.4
	Prox paper	14.0	43.2	72.3	71.5	61.8	34.5	56.7
	Prox paper	24.4	47.2	75.8	75.5	64.1	33.4	59.2
	Prox paper	29.7	46.8	76.6	75.9	66.0	33.5	59.8
	Prox paper	38.9	48.3	77.6	77.1	66.9	34.7	60.9
	Prox best	13.5	43.8	71.4	71.1	60.2	35.9	56.5
	Prox best	18.7	46.5	75.2	74.3	62.9	33.6	58.5
	Prox best	39.3	49.2	78.0	77.2	66.9	34.5	61.2
	TEAL	12.7	24.1	42.3	33.3	51.5	43.3	38.9
	TEAL	18.2	31.0	59.3	48.1	57.5	36.1	46.4
	TEAL	28.8	43.1	73.4	67.8	63.7	30.9	55.8
	TEAL	39.0	46.2	75.8	72.4	65.4	31.6	58.3
	CHESS	40.6	39.8	63.1	65.9	60.5	37.6	53.4
	Park (layer)	40.4	38.1	60.6	63.6	58.0	41.6	52.4
	R-Sparse	17.6	27.5	58.5	43.0	56.3	34.3	43.9
	R-Sparse	28.8	40.8	71.4	63.6	62.8	31.3	54.0
	WINA	17.4	27.6	57.4	46.0	54.8	34.9	44.1
	WINA	28.5	39.4	72.3	65.2	62.3	31.8	54.2
	WINA	39.1	43.2	74.8	71.3	64.5	32.3	57.2
	CATS	40.6	38.4	61.3	63.3	58.2	40.6	52.4
	M-COUNTDOWN	41.3	38.0	64.1	62.8	60.2	40.0	53.0

Table 16: Per-task accuracy under full-sequence sparsity on Park et al.'s zero-shot tasks, Qwen3.5 (continuation of Table 15; same metrics).

Model	Method	Reads (%)	ARC-C	ARC-E	HellaSwag	WinoGr.	TQA	Avg
Qwen3.5-35B-A3B	Stock	100	62.0	84.0	82.6	74.3	53.2	71.2
	Ours (P05)	13.7	60.8	83.2	80.6	72.1	49.6	69.3
	Ours (P10)	18.7	60.9	83.0	81.6	73.0	51.1	69.9
	Ours (P20)	28.6	61.4	83.5	82.1	73.7	52.2	70.6
	Ours (P30)	38.5	61.1	83.8	82.5	73.6	52.6	70.7
	Ours w/o recovery (P05)	12.9	58.9	82.3	79.7	71.6	50.9	68.7
	Ours w/o recovery (P10)	17.9	59.7	82.3	81.0	73.7	52.2	69.8
	Ours w/o recovery (P20)	27.8	60.3	83.0	81.9	75.1	52.3	70.5
	Ours w/o recovery (P30)	37.9	61.9	83.4	82.2	73.9	52.5	70.8
	Calibrated recovery (P05)	13.7	59.3	83.0	79.6	72.1	49.5	68.7
	Calibrated recovery (P10)	18.7	60.4	83.2	80.8	73.6	51.6	69.9
	Calibrated recovery (P20)	28.6	61.9	83.3	81.6	73.7	51.7	70.5
	Calibrated recovery (P30)	38.5	61.2	83.2	81.9	73.6	52.2	70.4
	Prox	9.0	61.6	83.4	78.1	69.7	49.5	68.5
	Prox	10.9	61.3	83.3	79.3	71.4	51.3	69.3
	Prox	13.4	61.3	82.8	80.4	72.1	52.2	69.8
	Prox	19.9	62.5	83.5	81.3	72.7	52.3	70.5
	Prox	31.1	60.5	83.7	82.0	73.2	53.2	70.5
	Prox	41.1	61.7	84.0	82.4	74.7	54.0	71.4
	TEAL	17.6	56.9	81.2	67.7	68.8	48.0	64.5
	TEAL	23.4	59.7	82.4	73.6	69.8	50.4	67.2
	TEAL	32.6	59.9	83.3	77.8	73.9	52.0	69.4
	TEAL	43.4	61.9	84.3	80.3	73.6	53.2	70.6
	Park (layer)	38.2	60.8	82.1	79.1	70.5	49.7	68.4
	Park (global)	38.0	58.5	82.0	78.3	69.9	49.2	67.6
	CHESS	38.8	59.9	82.5	78.5	71.8	49.7	68.5
	CHESS	41.1	61.1	83.0	79.6	73.1	48.5	69.1
	CATS	39.2	60.2	81.7	79.5	70.5	49.5	68.3
	CATS	41.7	59.9	81.9	80.1	71.0	50.6	68.7
	M-COUNTDOWN	38.8	59.3	82.9	75.6	69.9	51.8	67.9
	WINA	18.6	55.9	80.7	66.4	70.5	47.3	64.2
	WINA	30.5	59.6	83.2	75.5	71.9	52.9	68.6
	WINA	41.6	60.3	84.0	79.2	74.0	52.9	70.1
	Halving (k=1, 2, 3)	12.5	46.2	75.8	65.3	64.5	44.9	59.4
		25.0	55.6	81.2	75.9	71.0	51.3	67.0
		37.5	60.4	83.8	75.3	72.8	50.9	68.6

Table 17: Prox's few-shot protocol on Qwen3.5 (lm-eval 0.4.3, full-sequence sparsity): 5-shot MMLU (sample-weighted accuracy), 10-shot HellaSwag (acc_norm), 0-shot ARC-Easy (acc), 25-shot ARC-Challenge (acc_norm) and 0-shot PIQA (acc). Thresholded baselines at the reads they realize on each task, in parentheses; our variants read a fixed 18.7% and 38.5% (P10 and P30, with a recovery map) or 17.9% and 37.9% (without). Flips: MMLU questions whose correctness changes against stock (Dutta et al., 2024). In this raw-text format stock picks A or B for 55.7% of MMLU questions whose gold answers are A or B for 47.6%, the selection bias of multiple-choice scoring (Zheng et al., 2024); TEAL exceeds stock by flattening that preference (its net gains fall on gold C and D) while changing more answers than any other method. –: not run.

Method	MMLU	Flips (%)	HellaSwag	ARC-E	ARC-C	PIQA
Stock	78.4	–	82.2	84.0	73.6	82.5
Ours (P10)	72.7	11.2	82.6	83.1	72.1	81.9
Ours w/o recovery (P10)	71.9	11.7	81.8	82.6	72.2	81.8
Ours (P30)	76.1	5.9	82.8	83.4	74.0	82.3
Ours w/o recovery (P30)	75.7	6.4	82.6	83.5	73.3	82.4
Calibrated recovery (P10)	71.1	12.5	–	83.6	71.7	82.4
Calibrated recovery (P30)	75.3	6.7	–	83.4	73.1	82.3
Prox	73.4 (13.3)	11.9	–	–	–	–
Prox	74.8 (19.6)	9.5	80.7 (18.6)	83.7 (20.5)	73.6 (21.2)	82.1 (21.0)
Prox	77.4 (40.4)	5.5	–	83.8 (41.4)	73.9 (42.3)	82.4 (42.3)
TEAL	80.1 (23.2)	16.6	–	82.3 (24.0)	69.8 (24.5)	79.2 (24.3)
TEAL	82.4 (43.3)	10.2	–	84.3 (43.6)	74.0 (44.5)	82.4 (44.2)

Table 18: OLMoE-1B-7B task quality. Generation tasks (BBH 3-shot CoT, GSM8K 8-shot CoT, OLMo 2 generative mean) use stock prefill and sparse decode; WikiText-2 sparsifies every token after the first; likelihood is the mean of eight tasks with the second half of each request sparse. Our variants read a fixed fraction (input-sparse: batch-one price); threshold baselines: realized reads / score. Prox best: allocation chosen on development KL.

Method	BBH CoT	GSM8K	OLMo 2 gen.	WikiText PPL ↓	Likelihood
Stock	32.99	13.65	57.38	7.49	72.86
Ours 13.5% (P05)	26.03	7.88	55.19	15.99	60.11
Ours 18.5% (P10)	30.33	9.70	56.78	11.55	65.62
Ours 23.5% (P15)	31.13	10.24	57.17	9.92	68.29
Ours 28.5% (P20)	32.07	10.99	57.21	9.08	69.75
Ours 33.4% (P25)	32.40	12.28	57.46	8.57	70.47
Ours 38.4% (P30)	32.42	13.34	57.41	8.25	71.22
Ours w/o recovery 12.9% (P05)	21.69	5.23	54.93	20.39	58.23
Ours w/o recovery 17.8% (P10)	30.09	9.78	56.57	12.51	64.45
Ours w/o recovery 22.8% (P15)	31.61	10.46	56.98	10.32	–
Ours w/o recovery 27.8% (P20)	31.30	12.66	57.17	9.30	69.52
Ours w/o recovery 32.8% (P25)	31.99	13.19	57.40	8.70	–
Ours w/o recovery 37.9% (P30)	32.60	12.36	57.30	8.33	71.17
+ input-sparse screen 12.8% (P05)	28.72	9.86	55.82	12.91	63.71
+ input-sparse screen 17.8% (P10)	31.08	10.61	56.75	10.60	67.06
+ input-sparse screen 27.8% (P20)	32.13	11.60	57.27	8.93	70.54
+ input-sparse screen 37.8% (P30)	32.44	13.12	57.44	8.30	71.11
Prox paper	7.9 / 9.62	8.2 / 4.25	7.6 / 49.93	9.2 / 28.50	8.7 / 52.81
Prox paper	12.4 / 23.66	13.3 / 8.19	12.0 / 53.59	14.7 / 13.85	14.0 / 61.20
Prox paper	16.9 / 26.37	18.1 / 10.16	16.5 / 55.04	20.2 / 11.05	19.2 / 65.77
Prox paper	21.5 / 28.95	23.1 / 11.60	21.1 / 55.99	25.5 / 9.86	24.3 / 68.49
Prox paper	26.2 / 29.18	28.1 / 11.52	25.9 / 56.44	30.7 / 9.19	29.5 / 69.23
Prox paper	35.1 / 30.18	37.2 / 13.12	35.3 / 57.01	–	38.7 / 71.50
Prox best	16.4 / 25.14	17.6 / 11.37	16.0 / 54.71	19.6 / 11.25	18.7 / 65.31
Prox best	35.6 / 30.36	37.6 / 12.21	35.8 / 57.09	40.3 / 8.26	39.1 / 71.81
TEAL	12.3 / 3.50	12.7 / 3.11	11.5 / 43.25	13.5 / 262.0	13.0 / 40.36
TEAL	17.2 / 12.65	17.5 / 3.41	16.0 / 49.09	18.9 / 42.20	18.4 / 46.98
TEAL	27.3 / 25.44	28.3 / 9.63	25.6 / 55.40	30.2 / 12.21	28.9 / 60.32
TEAL	37.5 / 30.27	38.5 / 10.01	36.4 / 56.72	40.5 / 9.44	39.2 / 68.10
WINA	16.3 / 12.94	16.7 / 2.50	15.0 / 47.95	17.8 / 58.38	17.6 / 46.21
WINA	37.0 / 29.22	38.3 / 10.24	35.7 / 56.70	40.5 / 9.81	39.2 / 67.71
CHESS	40.1 / 18.69	40.2 / 5.08	39.9 / 53.71	40.9 / 18.69	40.8 / 59.64
CATS	40.3 / 6.23	40.6 / 2.43	40.0 / 53.62	40.9 / 34.93	40.7 / 55.86

Table 19: Speedup over the fastest tuned stock vLLM configuration at each batch size B (P10, one RTX PRO 6000 Blackwell). Values above 1 are faster than stock. –: not measured.

Measure	Method	Model	B=1	2	4	8	16	32	64	128
MoE block	Ours	OLMoE-1B-7B	1.67	1.61	1.92	1.81	1.47	1.29	1.04	0.77
		Qwen3-30B-A3B-Thinking	1.38	1.34	1.64	1.66	1.44	1.37	1.22	0.98
		Qwen3.5-35B-A3B	0.85	0.89	1.28	1.66	1.48	1.54	1.47	1.28
		Gemma 4 26B-A4B-it	1.01	1.05	1.25	1.28	1.25	1.27	1.17	1.03
	Ours w/o recovery	OLMoE-1B-7B	1.70	1.66	1.97	1.86	1.51	1.32	1.06	0.79
		Qwen3-30B-A3B-Thinking	1.40	1.38	1.69	1.71	1.48	1.41	1.24	1.00
		Qwen3.5-35B-A3B	0.86	0.91	1.31	1.71	1.51	1.58	1.50	1.31
		Gemma 4 26B-A4B-it	1.01	1.05	1.26	1.32	1.28	1.30	1.20	1.05
	Ours + input-sparse screen (whole sketch read)	OLMoE-1B-7B	1.57	1.57	1.83	1.72	–	–	–	–
		Qwen3-30B-A3B-Thinking	1.31	1.31	1.58	1.59	–	–	–	–
End-to-end TPOT	Ours	OLMoE-1B-7B	1.30	1.35	1.60	1.60	1.45	1.29	1.05	0.83
		Qwen3-30B-A3B-Thinking	1.16	1.19	1.39	1.43	1.36	1.32	1.20	1.00
	Ours w/o recovery	OLMoE-1B-7B	1.30	1.35	1.62	1.62	1.42	1.28	1.05	0.84
		Qwen3-30B-A3B-Thinking	1.16	1.20	1.41	1.46	1.36	1.33	1.21	1.02