CCQ: COMPENSATION-CONDITIONED QUADRATIC SELECTION FOR ABSORBING SKIPPED EXPERTS IN MOE LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Executing only half of the routed experts per token halves the routed-expert work of a fine-grained Mixture-of-Experts (MoE) model, but the absent experts leave an error. We split it into a *common mode*, the mean expert output scaled by the absent routing mass, and an *expert-specific* remainder. Restoring each part exactly on three MoEs shows that both matter, in model-dependent proportions, and that their costs nearly add, as a second-order analysis of the KL divergence predicts when their interaction is small. We therefore compensate them with different tools. A mass-gated low-rank readout of the always-executed shared expert, trained by KL distillation, recovers what the shared expert predicts of the absent output. Diagnostics show it is not merely a common-mode estimate, yet on the model where the common mode dominates, it removes as much KL as the exact common mode. The readout leaves most of the specific part, but each routed expert's output energy is predictable. We skip the half with the least predicted cost and add a sketch of the costliest absent expert at an eleventh of its width, trained on the residual the readout leaves. Implemented in vLLM with CUDA kernels, on Moonlight-16B-A3B, DeepSeek-V2-Lite and Qwen3.6-35B-A3B, the method, CCQ, lowers the full-vocabulary KL to the full model by 23–80% against skipping alone and by 13–18% against a LoRA with the same distillation budget, the strongest of seven baselines. Selection and sketch provide this gain; the readout alone only matches the LoRA. The method retains 98.0–99.2% of the full model's average score on 12 benchmarks, though mostly not significantly different from the readout or the LoRA. In vLLM's throughput benchmark on one GPU, it generates 1.19–1.23$\times$ as many tokens per second as the full model at batch size 1, up to 1.48$\times$ at batch size 4 and 1.06–1.15$\times$ at 256, mostly owing to skipping itself.

1 INTRODUCTION

Mixture-of-Experts (MoE) layers decouple parameters from per-token compute (Shazeer et al., 2017; Fedus et al., 2022). Fine-grained MoEs such as DeepSeek-V2-Lite, Moonlight and Qwen3.6 (DeepSeek-AI, 2024; Liu et al., 2025; Qwen Team, 2026) always run one or two shared experts and route each token to 6–8 of 64–256 small experts, whose weights dominate the memory reads that bound small-batch decoding. To run fewer of them, prior work uses adaptive routing thresholds (Lu et al., 2024; Zhong et al., 2024; Huang et al., 2024); to offset the skipped ones, it rescales kept weights (Chen & Yao, 2026), substitutes compressed vectors (He et al., 2025) or aligns output statistics (Kim et al., 2026). These designs are largely heuristic and empirical; to our knowledge, none analyzes what error the skipped experts leave, or which part is recoverable without running them. We answer both questions under a fixed budget: every token runs half of its routed experts in every layer, keeping routed work constant and kernel shapes static. Dropping the lower-weighted half (*plain skipping*) raises the KL divergence from the full model to 0.055–0.36.

To dissect this error (Section 2), we write each expert's output as a *common mode*, the mean of all routed experts' outputs, plus a *specific part*, its deviation from that mean. Restoring either part exactly shows **both parts matter, in model-dependent proportions**: the common mode removes 76% of the KL on Moonlight, the specific part 85% and 88% on DeepSeek-V2-Lite and Qwen3.6. Their costs nearly add, consistent with a second-order model (Section 3). Two observations show

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.

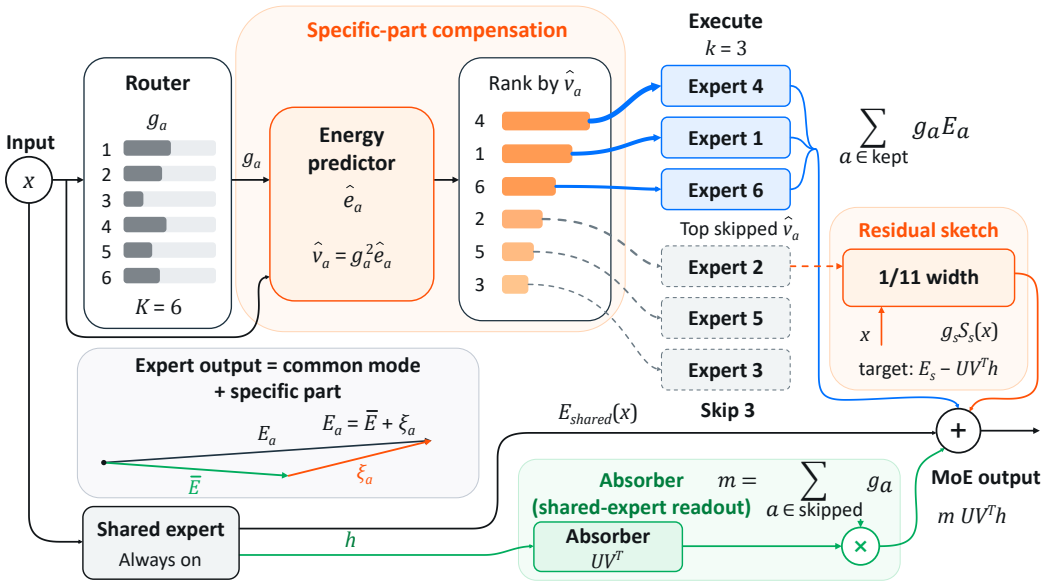


Figure 1: The method in one MoE layer (K=6, k=3). Slots are ranked by $\hat{v}_a = g_a^2 \hat{e}_a$ and the top k run (slots 2 and 6, of similar weight, swap relative to router order); the absorber adds a readout of the shared expert, scaled by the skipped mass; the top-$\hat{v}_a$ absent slot adds a residual sketch. Inset: expert output = common mode + specific part; $m = m_T$, subscript s: sketched slot.

what is recoverable without the skipped outputs. First, a KL-trained low-rank readout of the always-executed shared expert removes as much KL as the exact common mode on Moonlight, **yet it is not only a common-mode estimator**: it learns what the shared expert predicts about the absent output. Second, restoring the specific part on top of it would remove 78–82% of the remaining KL; that part's direction varies across experts, but **each expert's output energy is predictable**.

Our method, **CCQ**, has three components (Figure 1). The *absorber* adds this readout, scaled by the skipped routing mass. For the specific part, *energy-guided selection* runs not the heaviest experts but those whose absence would cost most, scored by squared routing weight times predicted output energy, $\hat{v}_a = g_a^2 \hat{e}_a$. A *residual sketch*, a copy of the highest-scoring skipped expert pruned to 1/11 of its width, approximates what the absorber misses for it. Selection and sketch cost about a tenth of an expert per token and layer; all run in vLLM (Kwon et al., 2023) with CUDA kernels.

We contribute (1) an anatomy of the skipping error with a second-order model that explains it (Sections 2–3), (2) a method built on it (Section 4), and (3) a quality and speed evaluation on one vLLM deployment (Section 5). Our method has a lower KL than all seven baselines on every model: 23–80% below plain skipping and 13–18% below the strongest, a LoRA (Hu et al., 2022) trained on the skipped model with the same distillation budget. The absorber alone only matches the LoRA; selection and sketch provide the margin. On 12 benchmarks, our method retains 98.2%, 98.0% and 99.2% of the full model's average score on Moonlight, DeepSeek-V2-Lite and Qwen3.6 (plain skipping: 91.1%, 96.3%, 98.2%). Against the absorber alone and the LoRA, however, its average score differs by at most half a point, mostly not significantly. It decodes up to 1.5× faster than the full model at small batch sizes, mostly due to skipping itself.

2 Anatomy of the Half-Skip Error

Fix an MoE layer and a token with layer input $x \in \mathbb{R}^d$. The router selects a set A of K experts with weights $g_a \geq 0$ and outputs $E_a(x) \in \mathbb{R}^d$. A skip rule keeps $H \subset A$ with $|H| = k = K/2$, leaving the absent set $T = A \setminus H$ with absent mass $m_T = \sum_{a \in T} g_a$. The shared expert always runs and computes a hidden vector $h(x) \in \mathbb{R}^{d_s}$. Skipping removes $\Delta = \sum_{a \in T} g_a E_a$ from the layer output.

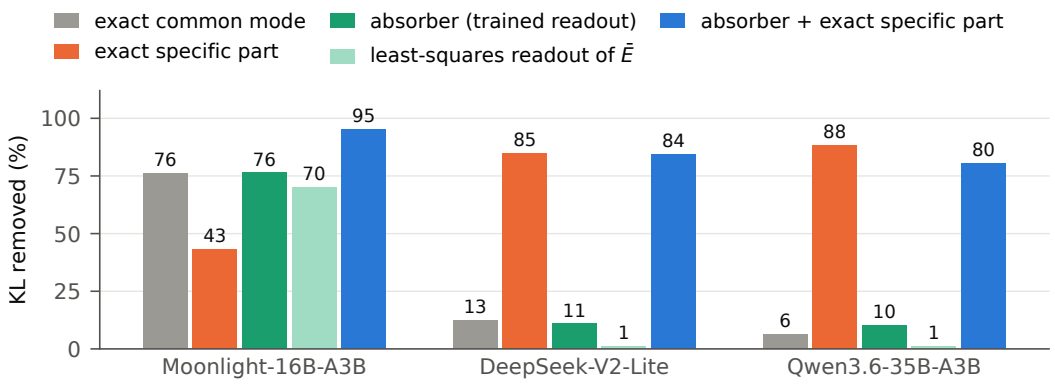


Figure 2: Share of plain skipping's test KL removed by restoring each part of the absent output exactly (Eq. (1)), by the trained absorber, by a same-rank ℓ_2 ridge readout of $\bar{E}$, and by the absorber plus the exact specific part. Values and intervals: Table 6.

With $\bar{E}(x) = \frac{1}{N}\sum_{b=1}^{N} E_b(x)$ the uniform mean of all N routed experts' outputs,

$$\Delta = \underbrace{m_T\,\bar{E}}_{\text{common mode}} + \underbrace{\textstyle\sum_{a\in T} g_a \xi_a}_{\text{specific part}}, \qquad \xi_a = E_a - \bar{E}. \tag{1}$$

Figure 2 restores each part exactly under router-order half skipping, in analysis runs that execute all N experts to form $\bar{E}$. **Both parts matter, in model-dependent proportions.** Restoring the common mode removes 76% of the KL on Moonlight but 13% and 6% on DeepSeek-V2-Lite and Qwen3.6. On these two, restoring the specific part removes 85% and 88% (43% on Moonlight). Moonlight scales its routing weights by 2.45, so they sum to more than one and both parts grow; the scale alone does not make the common mode dominate. **The exact parts are close to additive.** The KLs with only the common mode missing and with only the specific part missing sum to 0.81, 1.03 and 1.06 times the KL with both missing. F-orthogonal parts give 1 in the quadratic model (Section 3); the 95% intervals (Table 6, Appendix C) exclude 1.

A trained readout is not only a common-mode estimator. The absorber (Section 4) removes as much KL as the exact common mode on Moonlight (76% vs 76%) and a similar share on the others (11% vs 13%, 10% vs 6%). Yet its m_T^2-weighted mean cosine with $\bar{E}$ in the estimated Fisher metric $\tilde{F}$ is small (0.17, 0.10 and 0.03), and a same-rank ℓ_2 ridge readout of $\bar{E}$ from the same hidden units removes less (70%, 1% and 1%). Trained on the KL, the absorber learns what the shared expert predicts of the absent output, *specific error included*. With the specific part restored exactly, adding the absorber raises the KL on DeepSeek-V2-Lite and Qwen3.6 (0.0084→0.0087, 0.0107→0.0176) but lowers it on Moonlight (0.205→0.017). With the absorber standing in for the missing common mode, the additivity ratio rises to 1.16–1.27. **Restoring the specific part removes most of what remains**: 80%, 82% and 78% of the KL left by the absorber, the target of selection and sketches.

3 A SECOND-ORDER VIEW OF THE HALF-SKIP ERROR

This section gives an *explanatory model, not a guarantee*. A local quadratic approximation of the KL splits the error into the parts Section 2 measures, gives our rules their form and names what each rule ignores; Sections 2 and 5 test parts of it. We keep the notation of Section 2. A compensator adds $m_T c + \sum_{a\in T'} g_a S_a$: a mass-gated readout $c = s\,c_1$, $c_1 = U\zeta$, $\zeta = V^\top h$, of the shared expert's hidden units at a development-chosen scale s, and width-w sketches S_a on a subset $T' \subseteq T$ of the absent slots. The layer-output error is

$$\delta = \sum_{a\in T} g_a E_a - m_T c - \sum_{a\in T'} g_a S_a. \tag{2}$$

We make three approximations, following the Fisher view of pruning (Hassibi & Stork, 1992; Singh & Alistarh, 2020) applied to the layer output. First, we keep the second-order term $\frac{1}{2}\mathbb{E}\|\delta\|_F^2$ of the

KL $D_{\mathrm{KL}}(p_{\mathrm{full}}\|p)$, whose value and gradient vanish at $\delta = 0$; here $\|v\|_F^2 = v^\top F v$ and $F = F(x) \succeq 0$ is the Fisher information of the output distribution with respect to the layer output. This local expansion assumes no downstream routing decision flips (Appendix A). Second, we sum costs over layers and positions and drop cross terms. Third, we replace F by a token-independent low-rank-plus-isotropic estimate $\tilde{F}$ of $\mathbb{E}F$ per layer (Appendix B), ignoring any correlation of F with $\delta\delta^\top$. The results below are exact for this quadratic cost, not bounds on the KL. A residual sketch is trained on $E_a - c_1$, so $\varepsilon_a = E_a - c - S_a$ is its fitting residual at $s = 1$ and also carries $(1-s)c_1$ otherwise. With $E_a = \bar{E} + \xi_a$ and $m_{T\setminus T'} = \sum_{a\in T\setminus T'} g_a$,

$$\delta = \underbrace{m_{T\setminus T'}(\bar{E} - c)}_{\gamma} + \underbrace{\textstyle\sum_{a\in T\setminus T'} g_a\xi_a + \sum_{a\in T'} g_a\varepsilon_a}_{r=\sum_{a\in T} r_a}. \tag{3}$$

Here r_a is $g_a\xi_a$ (unsketched) or $g_a\varepsilon_a$ (sketched). The explicit common-mode term carries only the unsketched mass; the readout also enters each ε_a. A *plain* sketch (target E_a) with the readout on all of m_T counts slot a's readout twice, adding $-g_a c$ to δ. Residual sketches at $s = 1$, or a plain sketch with the readout on $m_{T\setminus T'}$, avoid this; on DeepSeek-V2-Lite all three tie (Table 3).

Proposition 1 (Two-part split)**.** $\mathbb{E}\|\delta\|_F^2 = \mathbb{E}\|\gamma\|_F^2 + \mathbb{E}\|r\|_F^2 + 2\mathbb{E}[\gamma^\top F r]$. *Under (S1)* $\mathbb{E}[\gamma^\top F r] = 0$ *(the parts are F-orthogonal in expectation), the common-mode cost* $\mathbb{E}\|\gamma\|_F^2$ *and the per-slot cost* $\mathbb{E}\|r\|_F^2$ *add. Under (S2)* $\mathbb{E}\sum_{a\neq b\in T} r_a^\top F r_b = 0$*, the per-slot cost is* $\mathbb{E}\sum_{a\in T}\|r_a\|_F^2$*, which is the specific cost* $\mathbb{E}\sum_{a\in T} g_a^2\|\xi_a\|_F^2$ *if* $T' = \emptyset$.

At $c = 0$, $T' = \emptyset$, (S1) is exactly additivity of the quadratic costs of the exact parts; (S2) does not enter. Read through this cost, the ratios 0.81, 1.03 and 1.06 (Table 6) put the cross term at $+19\%$, -3% and -6% of plain skipping's KL. Measured KLs also contain higher-order and cross-layer terms, so the ratios test the quadratic approximation and (S1) jointly; (S2) is untested.

What a KL-trained readout does. A readout can gain without estimating $\bar{E}$. Without sketches, it changes the cost of plain skipping by exactly $\big(\mathbb{E}\|m_T(\bar{E} - c)\|_F^2 - \mathbb{E}\|m_T\bar{E}\|_F^2\big) - 2\mathbb{E}[m_T c^\top F r]$: a change in the common-mode cost plus one in the (S1) cross term, which it can lower by predicting part of the specific error from h. Under the quadratic cost, the KL with the specific part restored exactly is $\frac{1}{2}\mathbb{E}\|m_T(\bar{E} - c)\|_F^2$, so Table 6 separates the two terms. On DeepSeek-V2-Lite and Qwen3.6 this KL rises with the absorber (Section 2), so the absorber's gain comes entirely from the cross term. On Moonlight the common-mode term's fall is about two thirds of the gain, although the $\tilde{F}$ diagnostics show little alignment with $\bar{E}$ ($R^2_{\bar{E}} = 0.05$, cosine 0.17; Appendix C). In the same reading, the additivity ratio with the absorber, 1.16–1.27, means $\mathbb{E}[\gamma^\top F r] < 0$: the readout's error is anti-aligned with r, as expected of a readout that predicts part of r, and (S1) fails for it.

Proposition 2 (Selection under the separable cost)**.** *Let $T' = \emptyset$, condition on token-level information $\mathcal{G}$ that fixes A, g, F, $\bar{E}$ and c (e.g. with $F = \tilde{F}$), and compare $\mathcal{G}$-measurable $T \subseteq A$ with $|T| = K - k$ under one c. Assume* $\mathbb{E}[(\bar{E} - c)^\top F\xi_a \mid \mathcal{G}] = 0$ *and* $\mathbb{E}[\xi_a^\top F\xi_b \mid \mathcal{G}] = 0$ *for all $a \neq b \in A$, i.e. (S1)–(S2) for every candidate T. With* $e_a = \mathbb{E}[\|\xi_a\|_F^2 \mid \mathcal{G}]$ *and* $\kappa = \|\bar{E} - c\|_F^2$,

$$J(T) = \mathbb{E}\big[\|\delta\|_F^2 \mid \mathcal{G}\big] = \kappa\, m_T^2 + \sum_{a\in T} v_a, \qquad v_a = g_a^2 e_a.$$

(i) If $\kappa = 0$, T minimizes J iff $\max_{a\in T} v_a \leq \min_{b\in A\setminus T} v_b$*; router order (skipping the $K - k$ smallest g_a) is optimal iff no skipped slot has a larger v_a than a kept one, e.g. if e is constant on A. (ii) Swapping a skipped slot a for a kept slot b lowers J iff* $v_a - v_b > \kappa\,(g_b - g_a)(2m_T + g_b - g_a)$.

Part (i) is our rule, extending router order to slots of unequal specific energy. Part (ii) is what it ignores: a residual common mode ($\kappa > 0$) makes skipping heavy slots cost more. The conditional assumptions are stronger than (S1)–(S2). Where they fail, as the ratio above suggests for the trained readout, J omits the coupling $2m_T\sum_{a\in T} g_a(\bar{E} - c)^\top F\,\mathbb{E}[\xi_a \mid \mathcal{G}]$ and pairwise terms; the proposition fixes the form of the rule, not a guarantee. Prop. 3 (Appendix A) splits what selecting by a score $\hat{v}$ loses against the best set into the score error where the sets differ and the change of the ignored terms; its terms are unknown at inference. As e_a needs $\bar{E}$, we rank by $\hat{v}_a = g_a^2\hat{e}_a$, with $\hat{e}_a$ predicted from x for the total energy $\|E_a\|_{\tilde{F}}^2$ (validation R^2 per layer 0.67–0.94, Appendix B). For $F = \tilde{F}$, $\mathbb{E}[\|E_a\|_F^2 \mid \mathcal{G}] = e_a + \|\bar{E}\|_F^2 + 2\bar{E}^\top F\,\mathbb{E}[\xi_a \mid \mathcal{G}]$. The total energy adds to v_a the term $g_a^2\|\bar{E}\|_F^2$, which, like (ii), favors keeping heavy slots, and a cross term of either sign. Fitted in log space, $\hat{e}_a$ is also a ranking proxy, not an estimate of e_a; Section 5 measures what it gains.

What a sketch can capture. A sketch keeps w of expert a's hidden units $u \in \mathbb{R}^I$ ($E_a = W_a u$) with a linear readout; on fixed units P, its F-weighted least-squares solution $B^\star$ has a closed form. Whatever its hidden features, a width-w sketch outputs vectors in a fixed subspace of dimension at most w. It thus captures no more of its target's energy than the top w uncentered principal components of the F-weighted target $F^{1/2}y$, with $y = E_a$, or $E_a - c_1$ for a residual sketch (Prop. 4 in Appendix A). $B^\star$ on a greedy P is one of our two initializations; training the gate and up rows can improve on any fixed-unit readout, but not beyond this ceiling. Under the separable cost with $\kappa = 0$, a sketch that captures a fraction $\rho_a(w)$ of slot a's specific energy lowers the cost by $v_a \rho_a(w)$. One sketch thus belongs on the absent slot with the largest $v_a \rho_a(w)$, for a common capture curve the one with the largest v_a; if that v_a is strictly largest, a small total width is best not split (Prop. 5). Our kernels sketch the top-$\hat{v}_a$ absent slot among experts with an enabled sketch. This heuristic also replaces specific by predicted total energy and, at $s \neq 1$, fits $E_a - c_1$ rather than $E_a - c$.

4 METHOD

Absorber. Every MoE layer adds $s\, m_T U\zeta$, $\zeta = V^\top h(x)$, with $V \in \mathbb{R}^{d_s \times q}$ and $U \in \mathbb{R}^{d \times q}$ ($q = 64$ for Moonlight and DeepSeek-V2-Lite, 32 for Qwen3.6). It is exactly zero when nothing is skipped and costs $q(d + d_s)$ multiply-accumulates per token. We train it once per model by KL distillation of the frozen model under a mixture of skip policies (Appendix B) and set $s \in \{0.5, 0.75, 1\}$ on development.

Energy-guided selection. Every token executes the k slots with the largest $\hat{v}_a = g_a^2 \hat{e}_a$. A per-expert ridge regression predicts $\log \|E_a(x)\|^2_{\tilde{F}}$ from the layer input, and $\hat{e}_a = \exp(\widehat{\log \|E_a\|^2_{\tilde{F}}})$ is the predicted output energy, a proxy for the specific energy e_a of Prop. 2. Energy is measured in the layer's output Fisher metric $\tilde{F}$: the top-256 eigenpairs of the gradient second moment of sampled-label log-likelihoods, plus an isotropic rest. The predictor costs K dot products per token.

Residual sketches. Each expert's sketch keeps $w \approx I/11$ SwiGLU units (128 of 1408; 48 of 512 for Qwen3.6), chosen greedily by captured ℓ_2 energy on the router-order absent slots. We initialize it by the readout of Prop. 4 or by truncation and train it on the residual target $E_a - U\zeta$ (at $s = 1$). A sketch is disabled if its validation residual exceeds its target energy (0.4%, 0% and 0.9% of expert-layers). At inference, the absent slot with the largest $\hat{v}_a$ among experts with an enabled sketch adds $g_a S_a(x)$. Nominal routed work in experts (projection MACs, one sketch) is $k + w/I + K/(3I)$ per token and layer: $+0.09$ for $I = 1408$, $+0.10$ for $I = 512$.

Deployment in vLLM. A general plugin replaces vLLM's MoE layer forward. It keeps the model's gate, router and routed-expert kernel, and adds three fused kernels: one for the energy selection, one for the sketch slot, and one for the shared expert's down projection together with ζ. The absorber term $s\, m_T U\zeta$ is added to the shared output. As in stock vLLM, the shared expert (here with the absorber) runs on a second CUDA stream, overlapped with the routing and the routed experts; the absorber waits for the absent mass on a cross-stream event (Figure 6 in Appendix F). At decode sizes, one CUDA kernel computes the gate, the routing (bit-exact to vLLM given the logits) and the selection; two more compute the routed experts and the sketch without an alignment pass (Appendix F). These kernels compute the same function as the general path up to floating-point summation order; Appendix E measures the difference.

5 EXPERIMENTS

Setup. We evaluate Moonlight-16B-A3B (base; K=6 of N=64 experts, 26 MoE layers), DeepSeek-V2-Lite (base; 6 of 64, 26 layers) and Qwen3.6-35B-A3B (post-trained; 8 of 256, 40 layers) in BF16 with $k = K/2$ in every MoE layer, on vLLM 0.29 (Kwon et al., 2023) and a single RTX PRO 6000 Blackwell Server Edition GPU. *KL* is the full-vocabulary KL from the full model, computed from vLLM's final hidden states and the LM head in FP32. We make every choice (scales, top-p level, reference mass, LoRA checkpoint) on 56 development documents (23K tokens) and test on 152 other documents (35K tokens), macro-averaged over five domains (Appendix C). One document per engine step makes every forward bitwise repeatable: the full model and the plugin at

Table 1: Half skipping on vLLM: test KL to the full model (lower is better) and each method's KL relative to ours. Every baseline (a)–(g) is worse than ours on every model, with a 95% paired interval excluding zero (Appendix C). All methods execute $K/2$ routed experts per token and layer, except ours ($K/2 + 0.09$ to 0.10) and Top-p routing (matched to ours on development). Training-free halving selects $k_2 = K/2$ on DeepSeek-V2-Lite and Qwen3.6. Bold: best; underlined: second best. Bottom: ours relative to the router-order frontier of Figure 5, interpolated at our $\mathcal{E}$.

	Moonlight		DeepSeek-V2-Lite		Qwen3.6	
	KL	vs ours	KL	vs ours	KL	vs ours
Top-$K/2$ (plain skipping)	0.3600	+406%	0.0551	+39%	0.0899	+30%
Renormalized top-$K/2$	0.2303	+224%	0.1258	+217%	0.2060	+199%
Top-p routing (Huang et al., 2024)	0.2875	+304%	0.0597	+50%	0.0863	+25%
Training-free halving (Chen & Yao, 2026)	0.2018	+184%	0.0551	+39%	0.0903	+31%
LDA (Kim et al., 2026)	0.1317	+85%	0.0558	+40%	0.0889	+29%
MoNE (Zhang et al., 2026)	0.4395	+518%	0.0693	+74%	0.1018	+48%
LoRA (Hu et al., 2022)	<u>0.0848</u>	+19%	<u>0.0483</u>	+22%	<u>0.0794</u>	+15%
CCQ (ours)	**0.0711**	–	**0.0397**	–	**0.0690**	–
ours vs absorber + top-p / keep-k frontier		-9.3%		-15%		-11%

$k = K$ give KL exactly 0. Intervals are 95% paired bootstraps over test documents, stratified by domain, and omit training variance. *Compute* $\mathcal{E}$ counts executed routed-expert units per token and layer, including the sketch (w/I per sketched slot) and the predictor ($K/(3I)$). Outside $\mathcal{E}$, the absorber adds 0.036, 0.036 and 0.026 units of MACs and the unmerged LoRA 0.066, 0.066 and 0.18 (parameters in Table 5). Given one training seed and KL gaps of up to 0.4% between arms that compute the same function (floating-point order; e.g. training-free halving at $k_2 = K/2$ vs plain skipping), we do not interpret KL differences below 2%.

Baselines. (a) *Top-$K/2$* is plain skipping by router weight; (b) *renormalized top-$K/2$* rescales the kept weights to the full routed mass; (c) *Top-p routing* (Huang et al., 2024) is applied post hoc within the router's top K, with p set on development to our $\mathcal{E}$. Compensators at $K/2$: (d) *Training-free halving* (Chen & Yao, 2026) rescales the kept weights to the router mass of the top k_2 experts, $k_2 \in \{K/2, \ldots, K\}$ chosen on development ($k_2 = K/2$ is plain skipping, $k_2 = K$ is (b); the source also considers $k_2 > K$, which we do not). (e) *LDA* (Kim et al., 2026) aligns output distributions using moments from 32K calibration tokens, applied only if it lowers the development KL. (f) *MoNE* (Zhang et al., 2026) replaces each skipped expert by its novice, $\sum_{a \in T} g_a \mu_a$, where μ_a is expert a's mean output over the calibration tokens routed to it (training-time defaults of Panda et al. (2025) are similar). (g) *LoRA* (Hu et al., 2022) (rank 64 on the attention output and shared-expert down projections) uses the absorber's KL distillation, steps and tokens, at plain skipping, with the checkpoint chosen on development. Merged into the weights, it costs no inference compute; the absorber is gated by the absent mass and cannot be merged. Self-distillation of the whole model (Lv et al., 2026) targets a similar budget by post-training the backbone and is not compared.

KL against the baselines. Table 1 compares all methods at the same number of executed experts; ours adds 0.09–0.10 experts for selection and sketch. **Ours has a lower KL than each of the seven baselines on all three models.** Renormalizing the kept weights helps on Moonlight, whose absent mass is large, but more than doubles the KL of plain skipping on the other two models: restoring the absent mass through the kept experts amplifies their specific parts. Top-p routing without compensation stays far above ours (+25% to +304%). Training-free halving and LDA help only on Moonlight (-44% and -63% against plain skipping). On the other two models, the development pool selects no rescaling for halving (a factor that is 1 only up to floating point: 0.0903 vs 0.0899 on Qwen3.6), and LDA changes the KL by $+1\%$ and -1%. *MoNE's novices raise the KL* above plain skipping on every model (+22%, +26%, +13%). An expert's mean output over its routed tokens is a poor stand-in for its output on the current token, whereas the token-dependent $\bar{E}(x)$ removes 76% of the KL on Moonlight. **The strongest baseline is LoRA**, with a KL 19%, 22% and 15% above ours. The absorber alone matches it (Table 2: 0.0852 vs 0.0848, 0.0491 vs 0.0483, 0.0805 vs 0.0794): a LoRA on the shared-expert down projection is an ungated readout of the same hidden units. **The margin over LoRA thus comes from the selection and the sketch.**

Table 2: Ablations (test KL): every combination of absorber, energy-guided selection and sketch at $K/2$ executed experts (sketch: residual with the absorber, plain without; Qwen3.6: residual only).

	absorber	selection	sketch	Moonlight	DeepSeek-V2-Lite	Qwen3.6
plain skipping				0.3600	0.0551	0.0899
	✓			0.0852	0.0491	0.0805
		✓		0.3510	0.0492	0.0786
	✓	✓		0.0818	0.0445	0.0728
			✓	0.1715	0.0450	0.0849
	✓		✓	0.0722	0.0429	0.0760
		✓	✓	0.1706	0.0413	0.0752
CCQ (ours)	✓	✓	✓	**0.0711**	**0.0397**	**0.0690**

Table 3: Controls and variants on DeepSeek-V2-Lite (test KL), where every variant was fitted.

	test KL
selection, with the absorber and no sketch	
predicted energies (ours)	0.0445
constant energy per expert	0.0514
true energies (oracle)	0.0407
sketch, with the absorber and selection	
residual, width $I/11$ (ours)	0.0397
plain, readout on the unsketched mass	0.0397
plain, readout on all absent mass	0.0397
residual, half width	0.0417
other	
one constant vector per layer, no selection or sketch	0.0645
keep $K/2+1$ + absorber	0.0206

Against the routed-unit frontier. Ours lies 9%, 15% and 11% below the frontier of the absorber with router-order top-p or keep-k skipping at our $\mathcal{E}$ (Figure 5, Appendix C). Relative to the absorber alone at $K/2$, the nearly free selection lowers the KL by 4%, 9% and 10%, and the sketch by a further 13%, 11% and 5%. Yet **one more executed expert (keep-$K/2+1$, with absorber) is far better than any compensator** (0.0206 vs 0.0397 on DeepSeek-V2-Lite, Table 3). The method is thus a tool for a fixed budget of $K/2$ in every layer.

Ablations. Table 2 runs every combination of the absorber, the energy-guided selection and the sketch. **Each component lowers the KL on its own, and each added component lowers it further**, though barely in one case. On Moonlight, once the sketch is on, selection adds only 0.5% without the absorber (0.1715 vs 0.1706) and 1.5% with it (0.0722 vs 0.0711, interval including zero); Moonlight's error is dominated by the common mode. On DeepSeek-V2-Lite and Qwen3.6, where the specific part dominates, selection lowers the KL by a further 7% and 9% on top of the absorber and sketch. *The learned energy predictor is what makes selection work.* On DeepSeek-V2-Lite (Table 3), selecting with a constant energy per expert (the predictor at the mean layer input) raises the KL of selection plus absorber from 0.0445 to 0.0514, worse than router order (0.0491); the true total energies (oracle, not a bound) lower it only to 0.0407. Across the three models, the predicted energies select the same k-set as the true ones for 88%, 82% and 66% of the token-layers (Kendall τ 0.92, 0.89, 0.84), against 82%, 58% and 44% for router order. On DeepSeek-V2-Lite, plain sketches match the residual one (0.0397), even with the readout on the full absent mass; the chosen scale may offset this double count (Section 3). Halving the sketch width costs 5%.

Downstream accuracy. Table 4 evaluates the methods on 12 lm-evaluation-harness 0.4.13 tasks with the timed vLLM engine (Appendix D). Log-likelihood tasks use the general path and generation the decode kernels, with standard prompts, no chat template and a 4,096-token GSM8K generation budget (default 256). **Ours retains 98.2%, 98.0% and 99.2% of the full model's average score**, against 91.1%, 96.3% and 98.2% for plain skipping (paired differences $+5.0$, $+1.1$ and $+0.7$ points; 95% intervals, not tabulated, exclude zero). *The KL gains of selection and sketch do not become*

Table 4: Twelve benchmarks (lm-evaluation-harness on the timed vLLM engine): primary score (%) per benchmark and average, grouped by category (HS: HellaSwag, WG: WinoGrande, OBQA: OpenBookQA, MATH: MATH-500, Knowl.: knowledge). Bold: best, underlined: second-best skip-half arm per column. Absorber-only, keep-$K/2{+}1$, standard errors and paired tests: Appendix D.

	Science QA				Commonsense			Reading/NLI		Knowl.	Math		
	ARC-E	ARC-C	OBQA	SciQ	HS	PIQA	WG	BoolQ	RTE	MMLU	GSM8K	MATH	Avg.
Moonlight													
Full model	82.7	58.0	45.4	96.0	78.4	80.7	71.1	80.3	66.1	69.8	75.7	42.4	70.5
Top-$K/2$ (plain skipping)	78.0	52.0	41.2	96.4	68.3	75.7	60.5	75.9	61.4	65.7	58.8	37.2	64.3
Renormalized top-$K/2$	77.9	53.7	42.0	94.0	75.8	79.4	66.5	69.3	60.3	64.6	58.9	20.6	63.6
Top-p routing (Huang et al., 2024)	79.4	53.0	40.4	**96.6**	69.3	77.0	66.9	76.5	62.5	66.2	63.5	38.2	65.8
Training-free halving (Chen & Yao, 2026)	79.2	54.4	43.0	95.2	73.6	78.8	66.4	74.4	**66.4**	66.4	63.5	34.8	66.3
LDA (Kim et al., 2026)	80.9	55.2	42.0	96.4	73.8	79.3	69.9	78.3	**66.4**	67.0	69.7	38.0	68.1
MoNE (Zhang et al., 2026)	77.3	50.9	39.4	94.7	67.5	78.2	67.8	75.0	65.0	57.3	57.2	32.0	63.5
LoRA (Hu et al., 2022)	**82.6**	56.4	43.2	96.5	**77.2**	80.3	70.9	78.8	65.7	67.2	**72.9**	37.2	69.1
CCQ (ours)	82.0	**56.7**	**43.6**	96.5	77.0	**80.4**	**71.4**	**79.9**	65.3	**67.3**	71.3	**40.0**	**69.3**
DeepSeek-V2-Lite													
Full model	76.2	48.7	43.4	94.8	78.0	80.4	71.0	79.8	61.7	58.0	40.6	16.8	62.5
Top-$K/2$ (plain skipping)	73.3	45.9	42.6	94.6	76.2	79.7	69.2	76.2	60.6	55.6	34.2	13.2	60.1
Renormalized top-$K/2$	71.8	45.5	43.8	92.9	76.1	79.2	67.1	70.2	55.6	54.2	30.4	12.2	58.2
Top-p routing (Huang et al., 2024)	74.3	45.6	42.4	94.6	**76.7**	**80.3**	70.4	77.1	60.3	**56.5**	35.0	**16.2**	60.8
Training-free halving (Chen & Yao, 2026)	73.3	45.9	42.6	94.6	76.2	79.7	69.2	76.2	60.6	55.6	34.2	13.2	60.1
LDA (Kim et al., 2026)	73.2	45.6	42.4	94.9	75.8	79.3	69.1	77.0	61.4	55.5	32.8	14.0	60.1
MoNE (Zhang et al., 2026)	73.1	45.1	41.6	94.9	74.9	79.3	**70.7**	77.2	**61.7**	55.0	30.9	13.2	59.8
LoRA (Hu et al., 2022)	74.0	**46.4**	43.0	94.7	76.5	79.7	69.9	76.4	61.4	55.7	37.5	15.2	60.9
CCQ (ours)	**74.8**	45.6	**44.4**	**95.1**	76.2	79.3	70.3	**77.8**	61.4	56.3	**38.1**	15.4	**61.2**
Qwen3.6													
Full model	72.1	55.5	44.0	96.9	83.5	82.6	74.4	85.0	76.2	82.3	94.2	11.0	71.5
Top-$K/2$ (plain skipping)	72.3	54.4	42.4	**96.8**	81.6	82.2	71.9	81.4	75.1	80.5	**93.5**	10.2	70.2
Renormalized top-$K/2$	66.9	49.1	41.2	92.3	81.4	81.1	65.4	77.5	54.9	76.1	92.7	3.2	65.2
Top-p routing (Huang et al., 2024)	73.3	53.2	42.8	96.5	**82.0**	**82.7**	73.6	80.1	74.4	80.1	91.9	9.6	70.0
Training-free halving (Chen & Yao, 2026)	72.3	54.4	42.4	**96.8**	81.6	82.2	71.9	81.4	75.1	80.5	**93.5**	10.2	70.2
LDA (Kim et al., 2026)	71.8	53.3	42.8	96.5	**82.0**	81.8	73.0	80.9	68.2	79.9	92.4	9.2	69.3
MoNE (Zhang et al., 2026)	73.1	53.2	41.6	96.5	81.1	81.8	74.3	81.8	77.3	79.7	93.3	12.0	70.5
LoRA (Hu et al., 2022)	**74.0**	**55.2**	**43.2**	96.6	81.8	82.1	**74.6**	**85.5**	75.8	**80.6**	92.9	10.8	**71.1**
CCQ (ours)	73.5	54.9	42.0	96.5	81.5	82.3	73.3	**85.5**	**77.6**	79.4	92.3	**12.2**	70.9

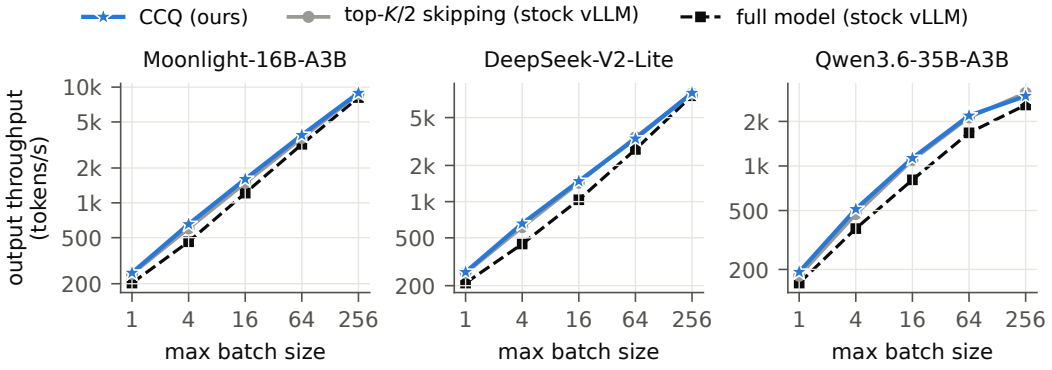


Figure 3: Output throughput against the batch limit with vLLM's official benchmark on one GPU (256 prompts of 256 tokens, 256 generated tokens each): the full model and top-$K/2$ skipping on stock vLLM (renormalized on Moonlight, Qwen3.6) and ours; throughput includes prefill. Values: Table 18.

significant average-accuracy gains: against the absorber alone and LoRA, ours differs by -0.2 to $+0.5$ points, and only the $+0.5$ over the absorber on DeepSeek-V2-Lite has an interval excluding zero. The largest per-benchmark gap to the full model is 4.4 points on Moonlight (GSM8K), 3.2 on DeepSeek-V2-Lite (ARC-Challenge) and 2.9 on Qwen3.6 (MMLU).

Kernel latency and batch throughput. Per MoE layer (Figure 4), ours beats top-$K/2$ skipping on vLLM's kernels at batch size 1 and is within 5% at 8. In vLLM's throughput benchmark on one GPU (Figure 3), **ours generates 1.19–1.23× as many tokens per second as the full model at batch size 1, 1.35–1.48× at 4 and 1.06–1.15× at 256**. It is 2–9% faster than top-$K/2$ on stock

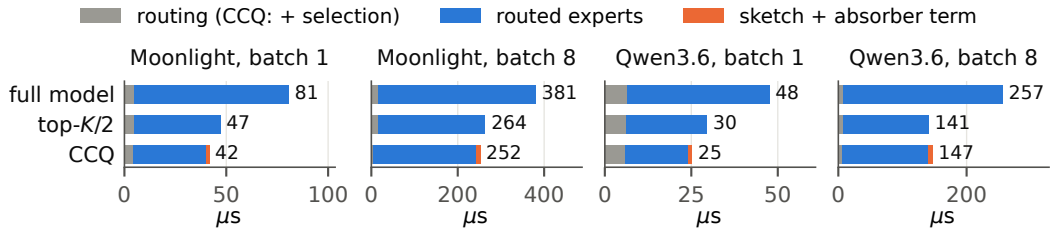


Figure 4: Latency (μs) of the routed part of one MoE layer at decode, by part (cold weights, CUDA graphs; the shared expert runs on a second stream and is excluded). Full model and top-$K/2$ use vLLM's kernels. DeepSeek-V2-Lite has Moonlight's expert shapes. Values: Table 17.

vLLM up to batch size 16, owing to our kernels (Appendix F), and within 6% at 64 and 256. With tensor and expert parallelism on two and four GPUs, top-$K/2$ on stock vLLM still runs 1.14–1.29$\times$ as fast as the full model at batch size 16 and 1.03–1.11$\times$ at 256, so the gain of skipping is not an artifact of a single GPU.

6 Related work

Dynamic expert skipping runs fewer experts, chosen by top-p, thresholds or learned allocators (Huang et al., 2024; Lu et al., 2024; Zhao et al., 2025; Xu et al., 2026; Huang et al., 2026), sensitivity-based gating (Zhong et al., 2024) or per-layer and per-token budgets (Liu et al., 2026; Chen et al., 2025; Chitty-Venkata et al., 2025). Others rescale kept weights (Chen & Yao, 2026), align output statistics (Kim et al., 2026), substitute similar experts (Wu et al., 2026), or retrain the model to tolerate fewer experts (Lv et al., 2026; Hannah et al., 2025, see also Chen et al., 2026). These methods decide how many or which experts run; we analyze the absent experts' error and compensate it at a fixed count. Our score $g_a^2 \hat{e}_a$ is a per-token, predicted counterpart of the router-weighted activation-norm saliency used for static expert pruning (Lasby et al., 2026). **Surrogates for missing experts** include compressed-expert vectors for lower-ranked activated experts at inference (He et al., 2025), default vectors for unexecuted experts in training (Panda et al., 2025) or for pruned experts (Zhang et al., 2026), a hypernetwork that transfers knowledge from unselected experts (Zhao et al., 2024), and a residual draft expert for self-speculative decoding (Han, 2026). These surrogates are per expert or per model; our readout conditions on the token's shared-expert state, and our sketch repairs one absent slot per token. **Expert compression** uses low-rank, pruned or merged experts (Li et al., 2025; Ai et al., 2025; Gu et al., 2025; Li et al., 2026) and repairs the router or model afterwards (Hyeon & Do, 2026b;a). **Second-order views of pruning** (Hassibi & Stork, 1992; Singh & Alistarh, 2020) motivate our metric; we serve with vLLM (Kwon et al., 2023).

7 Conclusion

Skipping half of a fine-grained MoE's routed experts leaves an error with a common mode and an expert-specific part, whose costs add approximately. A readout of the always-executed shared expert recovers the part of the absent output that this expert can predict. Most of the specific part remains, but per-expert output energy is predictable, which helps choose which half to run and which absent expert to sketch. On three MoEs, this lowers the KL below seven baselines in vLLM while retaining 98.0–99.2% of the full model's benchmark average and running up to 1.5$\times$ faster on one GPU at small batch sizes.

Limitations. We evaluate CCQ only at the half-expert budget ($k = K/2$) and on three MoEs; it has not yet been tried on much larger models, such as the trillion-parameter Kimi K2 and the 2.8T-parameter Kimi K3 (Kimi Team, 2025; 2026). Its performance on other GPUs and in real-world inference scenarios also remains to be explored.

AI USE STATEMENT

We used an LLM-based coding assistant, under the authors' direction, to assist with implementing the methods and experiments, preprocessing public text corpora, and interpreting results. We did not use generative AI to create data; qualitative data analysis is not applicable. We also used it to edit code, search the literature, and draft and polish the text. The research goals and key decisions are the authors'. We reviewed all AI-assisted work, tested the implementation against reference implementations, and generated every reported number from the experiment outputs by script. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We will release the vLLM plugin, the CUDA kernels with their tests, the runners for the KL experiments, the benchmarks and the timing, the fitted compensators, the LoRA baselines and the development/test document lists. All models are public checkpoints and the evaluation text comes from public corpora. Appendix B gives the fitting recipes, Appendix C the evaluation details, the software versions, the seed and the hardware, Appendix A the proofs and Appendix F the kernels and the timing protocol.

REFERENCES

Mengting Ai, Tianxin Wei, Yifan Chen, Zhichen Zeng, Ritchie Zhao, Girish Varatkar, Bita Darvish Rouhani, Xianfeng Tang, Hanghang Tong, and Jingrui He. ResMoE: Space-efficient compression of mixture of experts LLMs via residual restoration. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julen Etxaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, Charles Lovering, Niklas Muennighoff, Ellie Pavlick, Jason Phang, Aviya Skowron, Samson Tan, Xiangru Tang, Kevin A. Wang, Genta Indra Winata, François Yvon, and Andy Zou. Lessons from the trenches on reproducible evaluation of language models. *arXiv preprint arXiv:2405.14782*, 2024.

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*, volume 34, pp. 7432–7439, 2020.

Xing Chen and Hengshuai Yao. Training-free halving of activated experts in fine-grained mixture-of-experts models. *arXiv preprint arXiv:2609.04575*, 2026.

Yuanteng Chen, Peisong Wang, Yuantian Shao, Nanxin Zeng, Chang Xu, and Jian Cheng. Ban&Pick: Enhancing performance and efficiency of MoE-LLMs via smarter routing. *arXiv preprint arXiv:2509.06346*, 2025.

Yuanteng Chen, Qiwei Lai, Chen Tianqi, Peisong Wang, Yuantian Shao, Nanxin Zeng, Zhilei Liu, Chuangyi Li, Jing Liu, and Jian Cheng. You only need 2/3 of the chosen experts: An empirical study of dynamic expert pruning in fine-grained MoE LLMs. *arXiv preprint arXiv:2609.25809*, 2026.

Krishna Teja Chitty-Venkata, Sandeep Madireddy, Murali Emani, and Venkatram Vishwanath. LExI: Layer-adaptive active experts for efficient MoE model inference. *arXiv preprint arXiv:2509.02753*, 2025.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2924–2936, 2019.

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. *arXiv preprint arXiv:1803.05457*, 2018.

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

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

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.

Hao Gu, Wei Li, Lujun Li, Qiyuan Zhu, Mark G. Lee, Shengjie Sun, Wei Xue, and Yike Guo. Delta decompression for MoE-based LLMs compression. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 20497–20514, 2025.

Dengke Han. DraftExpert: Expansion-aware self-speculative decoding for end-device MoE inference. *arXiv preprint arXiv:2607.24434*, 2026.

Lauren A. Hannah, Soheil Zibakhsh, Kumari Nishu, Arnav Kundu, Mohammad Samragh Razlighi, Mehrdad Farajtabar, and Minsik Cho. MoE-PHDS: One MoE checkpoint for flexible runtime sparsity. *arXiv preprint arXiv:2509.23012*, 2025.

Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, volume 5, pp. 164–171, 1992.

Yifei He, Yang Liu, Chen Liang, and Hany Hassan Awadalla. Efficiently editing mixture-of-experts models with compressed experts. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 7227–7238, 2025.

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*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021b.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. Harder task needs more experts: Dynamic routing in MoE models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12883–12895, 2024.

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 (CVPR)*, 2026.

Sieun Hyeon and Jaeyoung Do. Beyond retraining-free MoE compression: A cost-normalized study of post-compression adjustment. *arXiv preprint arXiv:2609.06076*, 2026a.

Sieun Hyeon and Jaeyoung Do. Is retraining-free enough? The necessity of router calibration for efficient MoE compression. *arXiv preprint arXiv:2603.02217*, 2026b.

Dohyeon Kim, Bedionita Soro, and Sung Ju Hwang. Distribution-consistent inference for dynamic sparse mixture-of-experts. *arXiv preprint arXiv:2609.09241*, 2026.

Kimi Team. Kimi K2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Kimi Team. Kimi K3: Open frontier intelligence. Technical report, https://github.com/MoonshotAI/Kimi-K3, 2026.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.

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.

Lujun Li, Qiyuan Zhu, Jiacheng Wang, Xiaoyu Qin, Wei Li, Hao Gu, Sirui Han, and Yike Guo. Sub-MoE: Efficient mixture-of-expert LLMs compression via subspace expert merging. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 22994–23002, 2026.

Wei Li, Lujun Li, Hao Gu, You-Liang Huang, Mark G. Lee, Shengjie Sun, Wei Xue, and Yike Guo. MoE-SVD: Structured mixture-of-experts LLMs compression via singular value decomposition. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 35209–35230, 2025.

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

Baihui Liu, Kaiyuan Tian, Wei Wang, Zhaoning Zhang, Linbo Qiao, and Dongsheng Li. AllocMoE: Budget-aware expert activation allocation for efficient mixture-of-experts inference. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9653–9667, 2026.

Jingyuan Liu, Jianlin Su, et al. Muon is scalable for LLM training. *arXiv preprint arXiv:2502.16982*, 2025.

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)*, pp. 6159–6172, 2024.

Xingtai Lv, Li Sheng, Kaiyan Zhang, et al. Post-trained MoE can skip half experts via self-distillation. *arXiv preprint arXiv:2605.18643*, 2026.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018.

Ashwinee Panda, Vatsal Baherwani, Zain Sarwar, Benjamin Thérien, Sambit Sahu, Tom Goldstein, and Supriyo Chakraborty. Dense backpropagation improves training for sparse mixture-of-experts. In *Advances in Neural Information Processing Systems*, 2025.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all. https://qwen.ai/blog?id=qwen3.6-35b-a3b, April 2026. Model weights: https://huggingface.co/Qwen/Qwen3.6-35B-A3B.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020.

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*, volume 34, pp. 8732–8740, 2020.

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.

Sidak Pal Singh and Dan Alistarh. WoodFisher: Efficient second-order approximation for neural network compression. In *Advances in Neural Information Processing Systems*, volume 33, pp. 18098–18109, 2020.

Lintang Sutawika, Hailey Schoelkopf, Leo Gao, Baber Abbasi, Stella Biderman, Jonathan Tow, et al. EleutherAI/lm-evaluation-harness: v0.4.13. Zenodo, August 2026. URL `https://doi.org/10.5281/zenodo.22214970`.

Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages (MAPL)*, pp. 10–19. ACM, 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, 2017.

Juntong Wu, Jialiang Cheng, Fuyu Lv, Ou Dan, and Li Yuan. SERE: Similarity-based expert re-routing for efficient batch decoding in MoE models. In *International Conference on Learning Representations*, 2026.

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. *arXiv preprint arXiv:2609.05228*, 2026.

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*, pp. 4791–4800, 2019.

Geng Zhang, Yuxuan Han, Yuxuan Lou, Yiqi Zhang, Wangbo Zhao, and Yang You. MoNE: Replacing redundant experts with lightweight novices for structured pruning of MoE. In *International Conference on Learning Representations*, 2026.

Hao Zhao, Zihan Qiu, Huijia Wu, Zili Wang, Zhaofeng He, and Jie Fu. HyperMoE: Towards better mixture of experts via transferring among experts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 10605–10618, 2024.

Zijia Zhao, Longteng Guo, Jie Cheng, Xuange Gao, Hua Huang, and Jing Liu. Ada-K routing: Boosting the efficiency of MoE-based LLMs. In *International Conference on Learning Representations*, 2025.

Shuzhang Zhong, Ling Liang, Yuan Wang, Runsheng Wang, Ru Huang, and Meng Li. AdapMoE: Adaptive sensitivity-based expert gating and management for efficient MoE inference. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2024.

A Proofs for Section 3

Throughout, $\|v\|_F = (v^\top F v)^{1/2}$ is a seminorm for $F \succeq 0$ (a norm if $F \succ 0$). We assume that all second moments used below are finite.

Scope of the quadratic cost. The quadratic cost is a local model. For one layer, view the KL as a function of a perturbation δ of that layer's output. If the reference output probabilities are positive and the network above the layer is twice continuously differentiable near the native activations, the KL has value and gradient zero at $\delta = 0$ and Hessian F, so $\frac{1}{2}\delta^\top F\delta$ is its leading term. Hard routing in later layers, however, is smooth only while no routing decision changes. The model therefore has no error bound for the finite perturbations of half skipping. Replacing F by a fixed $\tilde{F}$ changes the cost by

$$\mathbb{E}\,\delta^\top(F-\tilde{F})\delta = \mathbb{E}\,\mathrm{tr}\left((F-\mathbb{E}F)\,\delta\delta^\top\right) + \mathrm{tr}\left((\mathbb{E}F-\tilde{F})\,\mathbb{E}[\delta\delta^\top]\right),$$

that is, the correlation of F with $\delta\delta^\top$ plus the estimation, clipping and truncation error of $\tilde{F}$. The two terms need not vanish, and they can cancel.

Eq. (3) and the readout identity. Substituting $E_a = \bar{E} + \xi_a$ for $a \in T \setminus T'$ and $E_a - S_a = c + \varepsilon_a$ for $a \in T'$ in Eq. (2): $\delta = \sum_{a\in T\setminus T'} g_a(\bar{E}+\xi_a) + \sum_{a\in T'} g_a(c+\varepsilon_a) - m_T c = m_{T\setminus T'}(\bar{E} - c) + \sum_{a\in T\setminus T'} g_a\xi_a + \sum_{a\in T'} g_a\varepsilon_a$, since $m_T = m_{T\setminus T'} + m_{T'}$. This holds for any S_a and any scale s. With a plain sketch $S_a \approx E_a$ and the readout on the full mass m_T, the sketched slot contributes $g_a(E_a - S_a) - g_a c \approx -g_a c$ instead. For the readout identity ($T' = \emptyset$, $\Delta = m_T\bar{E} + r$), $\|\Delta - m_T c\|_F^2 - \|\Delta\|_F^2 = \|m_T c\|_F^2 - 2m_T^2 c^\top F\bar{E} - 2m_T c^\top F r$, and the first two terms equal $\|m_T(\bar{E}-c)\|_F^2 - \|m_T\bar{E}\|_F^2$; taking expectations gives the identity, with no assumption on c. For instance, the readout that minimizes $\mathbb{E}\|\Delta - m_T U\zeta\|_F^2$ over U for fixed V satisfies the normal equations, so it lowers the cost by $\mathbb{E}\|m_T c^\star\|_F^2$. It earns this gain from the whole absent output, specific part included.

Prop. 1. Expand $\|\gamma+r\|_F^2 = \|\gamma\|_F^2 + 2\gamma^\top F r + \|r\|_F^2$ and $\|r\|_F^2 = \sum_{a\in T}\|r_a\|_F^2 + \sum_{a\neq b\in T} r_a^\top F r_b$, and take expectations; (S1) removes the middle term of the first expansion and (S2) the last sum of the second. Only the summed conditions are needed. An additivity ratio formed from these quadratic costs tests the summed (S1); a ratio formed from measured KLs also reflects higher-order and cross-layer terms. □

Prop. 2. For a $\mathcal{G}$-measurable candidate T, the indicators $\mathbf{1}\{a \in T\}$, m_T, the g_a, $\bar{E} - c$ and F are $\mathcal{G}$-measurable, and c does not depend on T. With $T' = \emptyset$, $\delta = m_T(\bar{E} - c) + \sum_{a\in A}\mathbf{1}\{a\in T\}g_a\xi_a$, so $\mathbb{E}[\|\delta\|_F^2 \mid \mathcal{G}] = \kappa m_T^2 + 2m_T\sum_{a\in T} g_a\mathbb{E}[(\bar{E}-c)^\top F\xi_a \mid \mathcal{G}] + \sum_{a,b\in T} g_a g_b\mathbb{E}[\xi_a^\top F\xi_b \mid \mathcal{G}]$; the assumptions remove the middle sum and the off-diagonal terms, which gives J. (i) With $\kappa = 0$, $J(T) = \sum_{a\in T} v_a$ over sets of fixed size $K - k$. If $\max_T v \le \min_{A\setminus T} v$, $J(T)$ is the sum of the $K - k$ smallest v_a, the minimum. Otherwise some $a \in T$, $b \notin T$ have $v_b < v_a$, and exchanging them lowers J. If e is constant on A, $v_a = eg_a^2$ is nondecreasing in $g_a \ge 0$, so router order satisfies the condition. (ii) The swap replaces m_T by $m_T + g_b - g_a$ and $\sum_T v$ by $\sum_T v - v_a + v_b$, so it changes J by $\kappa(g_b - g_a)(2m_T + g_b - g_a) - (v_a - v_b)$. □

The next statement needs none of the assumptions of Prop. 2. It splits what selecting by an imperfect score loses against the best set under a given cost into two terms, neither known at inference.

Proposition 3 (Selection with an imperfect score)**.** *Let J be any real function on the candidate sets $T \subseteq A$ with $|T| = K - k$, for instance $\mathbb{E}[\|\delta\|_F^2 \mid \mathcal{G}]$ without (S1)–(S2), and split it as $J(T) = \sum_{a\in T} v_a + R(T)$ for any per-slot values v_a. Let $\hat{T}$ consist of the $K - k$ slots with the smallest scores $\hat{v}_a$ (ties broken arbitrarily), and let $T^\star$ minimize J. Then for every $\lambda > 0$*

$$0 \le J(\hat{T}) - J(T^\star) \le \sum_{a\in\hat{T}\triangle T^\star} |v_a - \lambda\hat{v}_a| + R(\hat{T}) - R(T^\star),$$

and if $\hat{T}$ also minimizes $\sum_{a\in T} v_a$ (e.g. if $\hat{v}$ and v order the slots alike), the sum can be dropped.

Proof. $\hat{T}$ minimizes $\sum_{a\in T}\hat{v}_a$ over the candidates, so $\lambda\sum_{\hat{T}}\hat{v} \le \lambda\sum_{T^\star}\hat{v}$ and $\sum_{\hat{T}} v - \sum_{T^\star} v \le \sum_{\hat{T}}(v - \lambda\hat{v}) - \sum_{T^\star}(v - \lambda\hat{v})$; the slots in both sets cancel, which leaves at most $\sum_{\hat{T}\triangle T^\star}|v - \lambda\hat{v}|$. Adding $R(\hat{T}) - R(T^\star)$ gives the bound. If $\hat{T}$ minimizes $\sum_T v$, the first difference is ≤ 0. □

Take J to be the conditional quadratic cost, $\hat{v}$ $\mathcal{G}$-measurable and $v_a = g_a^2 e_a$. Then R collects what ranking by v_a ignores: the residual common mode κm_T^2, the coupling and the pairwise terms. The sum is the error of the deployed score on the slots where it disagrees with the best set. This error has

two sources: the gap between predicted and true total energy, and the gap between total and specific energy (the identity after Prop. 2). The free scale λ reflects that $\hat{e}_a$ is a ranking proxy. Both terms are unknown at inference, so the proposition locates the loss rather than bounding it numerically. Table 3 measures part of it: true instead of predicted total energies lower the KL of selection plus absorber on DeepSeek-V2-Lite only from 0.0445 to 0.0407.

Section 3 summarizes the following two consequences of the quadratic cost for sketches.

Proposition 4 (Sketch readout and ceiling)**.** *Let expert a compute $E_a = W_a u$ from its hidden units $u = u(x) \in \mathbb{R}^I$ ($W_a \in \mathbb{R}^{d\times I}$), let $\Sigma = \mathbb{E}[g_a^2 uu^\top]$ over the tokens at which a is absent, and let $F \succeq 0$ be token-independent (such as $\tilde{F}$). (i) Let a sketch keep the units $P \subseteq \{1,\dots,I\}$, $|P| = w$, with $\Sigma_{PP} \succ 0$ ($\Sigma_{:P}$ and $\Sigma_{P:}$: the columns and rows of Σ in P), and read them out linearly, $S_a = Bu_P$. Then $B^\star = W_a\Sigma_{:P}\Sigma_{PP}^{-1}$ minimizes $\mathbb{E}\, g_a^2\|E_a - Bu_P\|_F^2$ for every such F, with minimum* $\operatorname{tr}\left(FW_a(\Sigma - \Sigma_{:P}\Sigma_{PP}^{-1}\Sigma_{P:})W_a^\top\right)$. *(ii) Let y be a target with $\mathbb{E}\, g_a^2\|y\|_F^2 > 0$ ($y = E_a$, or $y = E_a - c_1$ for a residual sketch), and let $M_y = \mathbb{E}[g_a^2 F^{1/2} yy^\top F^{1/2}]$, the second moment of the F-weighted target, have eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots$, extended by zeros. Every sketch $S_a = Bf(x)$ of width w, with $B \in \mathbb{R}^{d\times w}$ and any features $f(x) \in \mathbb{R}^w$ of finite second moment (trained or not), captures a fraction*

$$\rho_a = 1 - \frac{\mathbb{E}\, g_a^2\|y - S_a\|_F^2}{\mathbb{E}\, g_a^2\|y\|_F^2} \leq \frac{\sum_{i\leq w}\lambda_i(M_y)}{\operatorname{tr} M_y}$$

of its target's energy. For $y = E_a$, $M_y = F^{1/2}W_a\Sigma W_a^\top F^{1/2}$.

Proof. (i) Since F is token-independent, the objective $\mathbb{E}g_a^2\|W_a u - Bu_P\|_F^2 = \operatorname{tr} F(W_a\Sigma W_a^\top - 2B\Sigma_{P:}W_a^\top + B\Sigma_{PP}B^\top)$ is a convex quadratic in B with gradient $2F(B\Sigma_{PP} - W_a\Sigma_{:P})$, which vanishes at $B^\star$ for every $F \succeq 0$ (the minimizer is unique if $F \succ 0$); substituting $B^\star$ gives the minimum. (ii) The vector $F^{1/2}S_a = F^{1/2}Bf(x)$ lies in $L = \operatorname{range}(F^{1/2}B)$, a subspace of dimension $\ell \leq w$ that does not depend on the token. With P_L the orthogonal projector onto L, $\|F^{1/2}y - z\|_2 \geq \|(I - P_L)F^{1/2}y\|_2$ for every $z \in L$, so $\mathbb{E}\, g_a^2\|y - S_a\|_F^2 \geq \mathbb{E}\, g_a^2\|(I - P_L)F^{1/2}y\|_2^2 = \operatorname{tr} M_y - \operatorname{tr}(P_L M_y)$. By Ky Fan's maximum principle, $\operatorname{tr}(P_L M_y) \leq \sum_{i\leq\ell}\lambda_i(M_y) \leq \sum_{i\leq w}\lambda_i(M_y)$, as $M_y \succeq 0$. Dividing by $\operatorname{tr} M_y = \mathbb{E}\, g_a^2\|y\|_F^2 > 0$ gives the bound, and $y = W_a u$ gives $M_y = F^{1/2}W_a\Sigma W_a^\top F^{1/2}$. □

The second relaxes the integer sketch widths to real values.

Proposition 5 (Allocating a sketch budget)**.** *Take the separable cost of Prop. 2 with $\kappa = 0$, its assumptions extended to the sketched residuals, so that a sketched slot costs $g_a^2\mathbb{E}[\|\xi_a - S_a\|_F^2 \mid \mathcal{G}]$; at $s = 1$ a residual sketch then targets ξ_a up to an F-null vector. Let a width-w sketch of any absent slot capture a fraction $\rho(w)$ of its specific energy, with ρ continuously differentiable and concave on $[0,\infty)$, $\rho(0) = 0$, $\rho \leq 1$ and $\rho' > 0$. Minimize $\sum_{a\in T} v_a(1 - \rho(w_a))$ over real $w_a \geq 0$ with $\sum_{a\in T} w_a \leq w_{\mathrm{tot}}$, $w_{\mathrm{tot}} > 0$, and let $v_{(1)} \geq v_{(2)}$ be the two largest v_a, with $v_{(1)} > 0$. At every optimum, (i) $v_a > v_b$ implies $w_a \geq w_b$; and (ii) if $v_{(1)} > v_{(2)}$ and $v_{(1)}\rho'(w_{\mathrm{tot}}) > v_{(2)}\rho'(0)$, the whole budget goes to the slot with the largest v_a. The condition in (ii) holds for all small w_{tot} when $v_{(1)} > v_{(2)}$.*

Proof. The objective is convex (each $-v_a\rho$ is convex for $v_a \geq 0$) and the constraints are linear, so the KKT conditions are necessary and sufficient: for some $\lambda \geq 0$ with $\lambda(w_{\mathrm{tot}} - \sum_a w_a) = 0$, $v_a\rho'(w_a) = \lambda$ if $w_a > 0$ and $v_a\rho'(0) \leq \lambda$ if $w_a = 0$. In both cases $v_a\rho'(w_a) \leq \lambda$. Concavity makes ρ' nonincreasing. (i) If $v_a > v_b$ and $w_a < w_b$, then $w_b > 0$ and $v_a\rho'(w_a) \geq v_a\rho'(w_b) > v_b\rho'(w_b) = \lambda$, a contradiction. (ii) If $\lambda = 0$, the top slot would need $v_{(1)}\rho'(w_{(1)}) \leq 0$, which is impossible; so $\lambda > 0$, the budget binds, and by (i) the top slot has the largest, hence a positive, width. Then $\lambda = v_{(1)}\rho'(w_{(1)}) \geq v_{(1)}\rho'(w_{\mathrm{tot}})$, and any other slot b with $w_b > 0$ would give $\lambda = v_b\rho'(w_b) \leq v_{(2)}\rho'(0) < v_{(1)}\rho'(w_{\mathrm{tot}}) \leq \lambda$. Finally, $\rho'(w_{\mathrm{tot}}) \to \rho'(0) > 0$ as $w_{\mathrm{tot}} \to 0$ by continuity, which gives the last claim. □

B FITTING THE COMPENSATORS

We fit all compensators once per model in PyTorch, with the model frozen; inference and every evaluation then run on vLLM. The training text consists of generic WikiText, CodeParrot and GSM8K-train documents, disjoint from the development and test pools and cut into 512-token rows. The first

262,144 tokens calibrate the predictor and the sketches, the next 65,536 estimate the Fisher metric, and the rest (4.3–5.0M tokens, depending on the tokenizer) trains the absorber and the LoRA.

Absorber. In each MoE layer the absorber adds $m_T c = s\, m_T U \zeta$. We initialize $U = 0$ and draw the entries of V i.i.d. from $\mathcal{N}(0, 1/d_s)$. We train the absorber at $s = 1$, jointly over layers, by full-vocabulary distillation: the loss is the token-averaged $D_{\mathrm{KL}}(p_{\mathrm{full}} \| p)$ to the frozen model at native routing. Each step draws one of $K/2 + 1$ equally likely skip policies and applies it in every layer. The policies are keep-k' (the top k' router slots) for $k' \in \{K/2, \ldots, K-1\}$, and top-$p$ with $p \sim \mathcal{U}(0.6, 0.95)$, which keeps the shortest router-order prefix whose weight reaches a fraction p of the routed mass. Skipped slots get zero weight, and the kept weights are never renormalized. All policies skip in router order, so at inference the absorber also sees energy-selected absent sets it was not trained on, a train/deploy mismatch like that of the sketches. Training runs 1,200 steps of 4,096 tokens with Adam ($3 \cdot 10^{-4}$), 30 warm-up steps, cosine decay and gradient clipping at 1. We select among checkpoints saved at each quarter of training by development KL at keep-$K/2$.

Equal-distillation-budget LoRA. We attach LoRA (Hu et al., 2022) of rank 64 to the attention output projection and the shared-expert down projection of every MoE layer. It is trained at router-order keep-$K/2$ with the same data, steps, tokens and optimizer settings as the absorber, and its checkpoint (at quarters of training) is selected by development KL on vLLM.

Fisher metric. For each MoE layer and calibration position i, we draw four labels $y_{ij} \sim p_{\mathrm{full}}(\cdot \mid x_i)$ and take the gradients σ_{ij} of $\log p_{\mathrm{full}}(y_{ij} \mid x_i)$ with respect to the layer output. We form $M_i = \frac{1}{4}\sum_j \sigma_{ij}\sigma_{ij}^\top$ and rescale it so that $\operatorname{tr} M_i$ is at most the 99.9% quantile of the traces in its 512-token row. $\hat{F}$ is the mean of the M_i, an uncentered second moment. $\tilde{F} = Q\Lambda Q^\top + \tau(I - QQ^\top)$ keeps the top-256 eigenpairs (Q, Λ) of $\hat{F}$ plus the isotropic rest $\tau = \max\{0, (\operatorname{tr}\hat{F} - \operatorname{tr}\Lambda)/(d - 256)\}$, which preserves the trace. The result is a token-independent, clipped proxy for $\mathbb{E}F$, positive definite when $\tau > 0$ and $\Lambda \succ 0$.

Energy predictor. For every expert, a ridge regression with intercept maps the layer input x to the log energy $\log \|E_a(x)\|_{\tilde{F}}^2$ (energies floored at 10^{-30}), fitted on the tokens routed to that expert. Calibration uses 262,144 tokens per model and holds out the last 10% for validation. On these validation tokens, which also give the R^2 below, we choose the ridge parameter per expert from $10^{-3}, \ldots, 10$ times the mean feature variance. A variant on the 64 leading principal components of x validated worse on every layer of the three models. The R^2 quoted in Section 3 is computed per layer as $1 - \sum_a \mathrm{SSE}_a / \sum_a \mathrm{SST}_a$ on the log energies of the validation tokens routed to each expert a, with SST_a centered at that expert's mean. It ranges over 0.74–0.94, 0.81–0.92 and 0.67–0.88 across the layers of Moonlight, DeepSeek-V2-Lite and Qwen3.6.

Sketches. Calibration uses the router-order absent slots (router ranks $K/2{+}1, \ldots, K$, weighted by g_a^2), not the slots that energy-guided selection skips at inference; this is a deployment approximation. For each expert, we order units greedily by the Euclidean output energy that the readout $B^\star$ of Prop. 4 captures. We copy the gate and up rows of the chosen units and initialize the down projection with $B^\star$ (plain target, also for residual sketches; ridge 10^{-4}) or with the truncated down projection, whichever validates better. Sketches of three nested widths $w_1 < w_2 < w_3$ share unit prefixes ($I/44$, $I/22$, $I/11$, rounded to multiples of 4: 32/64/128 for $I = 1408$, 12/24/48 for $I = 512$). We train their gate, up and down rows on the layer's loss

$$\mathcal{L} = \sum_{j=1}^{3} \omega_j \frac{\sum g_a^2 \|y_a - S_{a,w_j}\|_{\tilde{F}}^2}{\sum g_a^2 \|y_a\|_{\tilde{F}}^2}, \qquad (\omega_1, \omega_2, \omega_3) = (\tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{2}),$$

where the sums run over its absent slots and the target is $y_a = E_a - U\zeta$ (residual sketches) or $y_a = E_a$ (plain sketches, Moonlight and DeepSeek-V2-Lite only). We use Adam (10^{-4}, 4 epochs) and keep the best validation epoch. A sketch is enabled for an expert only if it beats the zero sketch, that is, if its validation residual at the deployed width is below its target energy. The others (0.4%, 0% and 0.9% of the expert–layer pairs) are disabled.

Fitting cost and memory. Table 5 lists the one-time fitting time on a single GPU and the added parameters.

Table 5: One-time fitting cost (GPU-minutes on one RTX PRO 6000) and added parameters (memory in BF16; the energy predictor in FP32).

	Moonlight	DeepSeek-V2-Lite	Qwen3.6
absorber distillation (GPU-min)	22.1	19.8	95.8
Fisher, predictor, sketches (GPU-min)	16.0	15.9	48.1
LoRA distillation (GPU-min)	12.4	12.1	54.4
sketch parameters (GB, BF16)	2.6	2.6	6.0
predictor + absorber parameters (MB)	29.8	29.8	90.5
LoRA parameters (MB)	29.8	29.8	44.6
model weights (GB, BF16)	31.9	31.4	70.0

C Evaluation details

Software and hardware. We use vLLM 0.29.0, PyTorch 2.13 (CUDA 13.0), Transformers 5.16, Triton 3.7 (Tillet et al., 2019) and lm-evaluation-harness 0.4.13 (Sutawika et al., 2026; Biderman et al., 2024). Every run uses one seed (20261021), BF16 weights, the Triton MoE backend with configurations tuned for the GPU, and a single NVIDIA RTX PRO 6000 Blackwell Server Edition GPU.

Anatomy values. Table 6 gives the values behind Figure 2, together with the additivity ratios and the readout cosines.

Table 6: Anatomy of the half-skip error on vLLM: test KL to the full model at half skipping by router weight, and the share removed. "Exact" rows restore one part of the absent output exactly (Eq. (1)). Additivity: the sum of the KLs with one part missing, divided by the KL with both missing, with 95% bootstrap intervals; it is 1 under (S1) of Prop. 1 in the quadratic model. "With the absorber": the same ratio with the absorber standing in for the common mode wherever it is missing, i.e. the KLs of the rows "+ exact common mode" and "+ absorber + exact specific part" over that of "+ absorber". Removed: share of plain skipping's KL. Least-squares readout: a ridge fit of $\bar{E}$ on h, truncated to rank q (see Diagnostics) and applied at the development-chosen scale (1, 0.5, 0.5). Last row: m_T^2-weighted mean of the per-token $\tilde{F}$-cosines between the absorber's output and $\bar{E}$ (Table 10).

	Moonlight		DeepSeek-V2-Lite		Qwen3.6	
	KL	removed	KL	removed	KL	removed
plain skipping (top-$K/2$)	0.3600	–	0.0551	–	0.0899	–
+ exact common mode	0.0859	76%	0.0482	13%	0.0843	6%
+ exact specific part	0.2050	43%	0.0084	85%	0.0107	88%
+ absorber	0.0852	76%	0.0491	11%	0.0805	10%
+ least-squares readout of $\bar{E}$	0.1071	70%	0.0545	1%	0.0891	1%
+ absorber + exact specific part	0.0174	95%	0.0087	84%	0.0176	80%
additivity, exact parts	0.81 [0.76, 0.85]		1.03 [1.01, 1.04]		1.06 [1.04, 1.08]	
additivity, with the absorber	1.21 [1.19, 1.24]		1.16 [1.14, 1.18]		1.27 [1.24, 1.29]	
cosine of the absorber output and $\bar{E}$	0.17		0.10		0.03	

Pools and bootstrap. Table 7 gives the documents, tokens and KL of each domain; the sources are PG-19 (Rae et al., 2020), WikiText-2 (Merity et al., 2017), GSM8K (Cobbe et al., 2021) and code. The macro average weights the five domains equally, and each domain's KL is token-weighted over its documents. Intervals resample documents with replacement within each domain (1,000 resamples) and apply the same resample to every arm. We compute differences and ratios on the resampled macro averages. PG-19 passages are resampled as documents, not clustered by book, and all timings use BF16 weights.

Routed-unit frontier. Figure 5 places our method against the routed-unit frontier of Table 1.

Table 7: Test pool by domain (documents, tokens) and per-domain test KL of plain skipping, LoRA and ours. PG-19 passages come from books whose windows are also in the test pool; code, math and Wikipedia documents share sources (not documents) with the training text.

model	domain	docs	tokens	plain skip.	LoRA	ours
Moonlight	PG-19 windows	8	4088	0.5083	0.0904	0.0748
	PG-19 passages	32	4096	0.5917	0.0906	0.0750
	code	48	12086	0.1389	0.0677	0.0570
	math	48	6737	0.2345	0.0966	0.0820
	Wikipedia	16	8176	0.3264	0.0788	0.0665
DeepSeek-V2-Lite	PG-19 windows	8	4071	0.0605	0.0570	0.0508
	PG-19 passages	32	4065	0.0575	0.0541	0.0469
	code	48	12203	0.0381	0.0352	0.0269
	math	48	7276	0.0495	0.0366	0.0292
	Wikipedia	16	8168	0.0698	0.0585	0.0450
Qwen3.6	PG-19 windows	8	4077	0.1033	0.0963	0.0843
	PG-19 passages	32	4051	0.0900	0.0831	0.0738
	code	48	12120	0.0825	0.0723	0.0620
	math	48	7207	0.0560	0.0489	0.0400
	Wikipedia	16	8163	0.1177	0.0967	0.0847

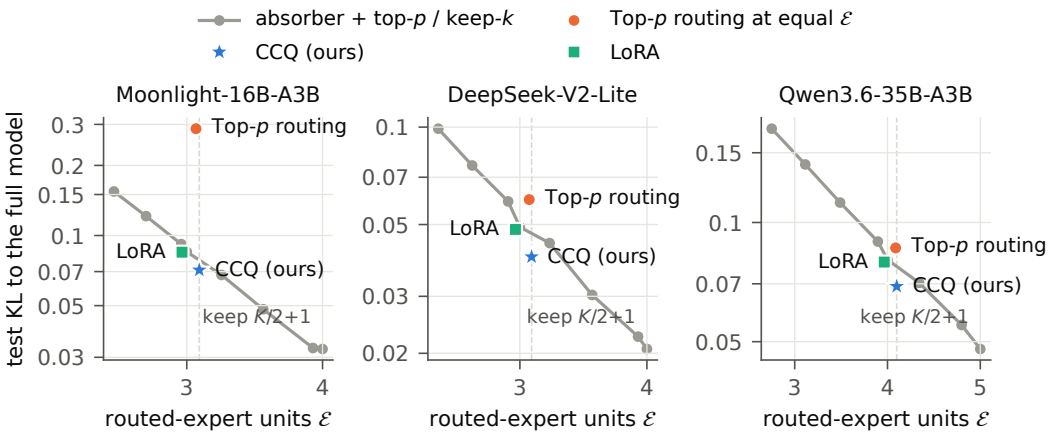


Figure 5: Half skipping against the routed-unit frontier (test KL, log scale). Grey: the absorber with router-order top-p skipping and with keep-$K/2$, keep-$K/2{+}1$. Markers: ours, Top-p routing (no compensation) at our $\mathcal{E}$, LoRA. Vertical line: our $\mathcal{E}$.

Paired intervals. Table 8 lists the 95% intervals of the paired KL differences (arm minus ours).

Table 8: Test KL of each arm minus ours with 95% paired bootstrap intervals ($\times 10^{-3}$).

arm minus ours	Moonlight	DeepSeek-V2-Lite	Qwen3.6
Top-$K/2$ (plain skipping)	+288.88 [+247.91, +327.82]	+15.36 [+14.30, +16.66]	+20.91 [+19.03, +22.98]
Renormalized top-$K/2$	+159.23 [+150.70, +168.54]	+86.04 [+82.61, +89.76]	+137.08 [+129.85, +144.92]
Top-p routing (Huang et al., 2024)	+216.43 [+173.06, +258.87]	+19.98 [+18.31, +21.93]	+17.29 [+14.49, +20.10]
Training-free halving (Chen & Yao, 2026)	+130.72 [+109.52, +152.61]	+15.37 [+14.31, +16.65]	+21.30 [+19.30, +23.48]
LDA (Kim et al., 2026)	+60.62 [+49.26, +73.00]	+16.03 [+14.95, +17.35]	+19.98 [+17.89, +22.17]
MoNE (Zhang et al., 2026)	+368.44 [+337.76, +398.51]	+29.57 [+27.69, +31.71]	+32.81 [+30.57, +35.19]
LoRA (Hu et al., 2022)	+13.72 [+11.84, +15.87]	+8.56 [+7.66, +9.57]	+10.48 [+8.61, +12.39]
absorber only	+14.09 [+12.21, +16.02]	+9.37 [+8.50, +10.38]	+11.58 [+10.30, +12.93]
selection + absorber	+10.69 [+9.30, +12.22]	+4.81 [+3.99, +5.60]	+3.88 [+2.76, +5.03]
absorber + sketch	+1.08 [-0.37, +2.39]	+3.19 [+2.52, +3.81]	+7.01 [+5.69, +8.37]
plain sketch, absorber on the unsketched mass	+2.13 [+0.48, +3.95]	-0.04 [-0.28, +0.18]	–

Baseline tuning. Each tuned setting of the baselines and of CCQ is chosen by its development KL; Table 9 lists every candidate.

Table 9: Development KL of every tuning candidate (bold: chosen; –: not a candidate for that model). For Top-p routing, the threshold p is matched to CCQ's development compute $\mathcal{E}$ (in parentheses).

tuned setting	candidate	Moonlight	DeepSeek-V2-Lite	Qwen3.6
training-free halving: reference mass k_2	3	0.3053	**0.0564**	–
	4	**0.1714**	0.0686	**0.0926**
	5	0.1744	0.0949	0.0980
	6	0.2177	0.1268	0.1239
	7	–	–	0.1610
	8	–	–	0.2028
LDA: alignment	applied	**0.1290**	**0.0552**	**0.0914**
	not applied	0.3053	0.0567	0.0923
absorber only: scale s	0.5	0.1105	0.0499	0.0824
	0.75	0.0840	0.0484	**0.0823**
	1.0	**0.0781**	**0.0474**	0.0837
CCQ: scale s	0.5	0.0838	0.0390	0.0724
	0.75	0.0671	**0.0385**	**0.0722**
	1.0	**0.0655**	0.0388	0.0730
LoRA: checkpoint	step 300	0.0835	0.0487	0.0829
	step 600	0.0799	0.0474	0.0834
	step 900	0.0780	0.0468	0.0822
	step 1200	**0.0774**	**0.0465**	**0.0816**
Top-p routing: p (dev $\mathcal{E}$)	matched to CCQ	0.696 (3.09)	0.627 (3.09)	0.622 (4.10)

Diagnostics. Table 10 summarizes the readout and selection diagnostics on the test pool, computed on the full-model path. Here m_T is the router-order absent mass and $c = sU\zeta$ is the absorber's readout at its development-chosen scale. Each statistic is computed per layer and averaged over layers, except $R^2_{\bar{E}}$, which pools all token-layers. The least-squares readout of $\bar{E}$ in Table 6 first solves, on calibration tokens, the ridge problem

$$W_\lambda = \arg\min_W \Big\{ \sum m_T^2 \|\bar{E} - Wh\|_2^2 + \lambda \|W\|_{\mathrm{Fro}}^2 \Big\}, \qquad \lambda = 10^{-3}\,\frac{\operatorname{tr} H}{d_s}, \quad H = \sum m_T^2 hh^\top,$$

and then truncates W_λ to the absorber's rank q by an SVD. This fit-then-truncate readout is in general not the best rank-q least-squares readout. Because $\tilde{F}$ may be singular, we use the stabilized cosine $\cos_{\tilde{F}}(u,v) = u^\top \tilde{F} v / \max\{\|u\|^2_{\tilde{F}} \|v\|^2_{\tilde{F}},\, 10^{-30}\}^{1/2}$.

Table 10: Diagnostics. $R^2_{\bar{E}} = 1 - \sum m_T^2 \|\bar{E} - c\|^2_{\tilde{F}} / \sum m_T^2 \|\bar{E}\|^2_{\tilde{F}}$, an uncentered score relative to the zero readout; cosine: $\sum m_T^2 \cos_{\tilde{F}}(c, \bar{E}) / \sum m_T^2$ over tokens, a weighted mean of per-token cosines (not the cosine of pooled inner products); agreement: fraction of token-layers whose selected k-sets coincide; τ: Kendall τ_a between $\hat{v}_a$ and the true $v_a^{\mathrm{out}} = g_a^2 \|E_a\|^2_{\tilde{F}}$ over the K slots, averaged over tokens. "True energy" and "true v" denote this total-energy score, not the specific-energy v_a of Prop. 2.

	Moonlight	DeepSeek-V2-Lite	Qwen3.6
$R^2_{\bar{E}}$ of the readout (pooled)	0.05	-0.05	-4.37
mass-weighted $\tilde{F}$-cosine	0.17	0.10	0.03
agreement predicted / true energy	0.88	0.82	0.66
agreement router / true energy	0.82	0.58	0.44
agreement predicted / router	0.84	0.60	0.48
Kendall τ predicted vs true v	0.92	0.89	0.84

D PER-BENCHMARK SCORES

The twelve benchmarks are ARC-Easy and ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), OpenBookQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2020), BoolQ (Clark et al., 2019), RTE from GLUE (Wang et al., 2019), SciQ (Welbl et al., 2017), MMLU (Hendrycks et al., 2021a), GSM8K (Cobbe et al., 2021) and MATH-500 (Lightman et al., 2024), a 500-problem subset of MATH (Hendrycks et al., 2021b). Tables 11–13 give the per-benchmark scores and Table 14 the paired differences. The average weights the twelve benchmarks equally, with MMLU pooling the items of its subjects. Retention is an arm's average divided by the full model's. Each of 2,000 paired bootstrap replicates resamples items with replacement within each benchmark, uses the same items for both arms and averages the twelve per-benchmark differences equally. Intervals are the 2.5th and 97.5th percentiles; they do not cover training variance.

GSM8K generation budget. Under lm-evaluation-harness's default budget of 256 generated tokens, our first GSM8K run on Qwen3.6 placed ours 6.6 points below the absorber alone. The decode kernels were not the cause. A rerun of the first 300 problems (flexible-extract accuracy) gave 82.7% with the decode kernels and 82.3% with the general path at decode sizes (absorber alone 87.3%, selection + absorber 85.7%, full model 83.0%). Without a chat template, Qwen3.6 opens every answer with a `<think>` block. It either closes the block at once and answers directly, or fills it with a long reasoning trace that the 256-token budget truncates before the answer. On the first 40 problems, a long trace occurred 1 time for the absorber alone, 7 times for selection + absorber and 10 times for ours. Accuracy was 0–14% on those problems and 90%, 94% and 90% on the directly answered ones. We therefore rerun GSM8K for every arm of all three models with a budget of 4,096 tokens, and all tables report these scores. At this budget, ours remains 1.7 points below the absorber alone on Qwen3.6's GSM8K (95% interval $[-2.8, -0.5]$, Table 14). Qwen3.6's low MATH-500 scores (11.0% for the full model) likely reflect the same truncation under the unchanged default budget. MATH-500 and the other benchmarks keep their first run.

Table 11: Per-benchmark primary scores (%), Moonlight-16B-A3B.

	Science QA				Commonsense			Reading/NLI		Knowl.	Math		
arm	ARC-E	ARC-C	OBQA	SciQ	HS	PIQA	WG	BoolQ	RTE	MMLU	GSM8K	MATH	avg.
Full model	82.7	58.0	45.4	96.0	78.4	80.7	71.1	80.3	66.1	69.8	75.7	42.4	70.5
Top-$K/2$ (plain skipping)	78.0	52.0	41.2	96.4	68.3	75.7	60.5	75.9	61.4	65.7	58.8	37.2	64.3
Renormalized top-$K/2$	77.9	53.7	42.0	94.0	75.8	79.4	66.5	69.3	60.3	64.6	58.9	20.6	63.6
Top-p routing (Huang et al., 2024)	79.4	53.0	40.4	96.6	69.3	77.0	66.9	76.5	62.5	66.2	63.5	38.2	65.8
Training-free halving (Chen & Yao, 2026)	79.2	54.4	43.0	95.2	73.6	78.8	66.4	74.4	66.4	66.4	63.5	34.8	66.3
LDA (Kim et al., 2026)	80.9	55.2	42.0	96.4	73.8	79.3	69.9	78.3	66.4	67.0	69.7	38.0	68.1
MoNE (Zhang et al., 2026)	77.3	50.9	39.4	94.7	67.5	78.2	67.8	75.0	65.0	57.3	57.2	32.0	63.5
LoRA (Hu et al., 2022)	82.6	56.4	43.2	96.5	77.2	80.3	70.9	78.8	65.7	67.2	72.9	37.2	69.1
Absorber only	82.2	56.7	43.0	96.2	77.1	80.7	70.7	78.9	65.7	67.2	72.8	38.2	69.1
CCQ (ours)	82.0	56.7	43.6	96.5	77.0	80.4	71.4	79.9	65.3	67.3	71.3	40.0	69.3
Keep $K/2$+1 + absorber	82.1	57.3	46.4	96.1	77.8	80.5	71.3	79.8	65.3	68.7	74.9	41.6	70.2
s.e. (ours)	0.8	1.4	2.2	0.6	0.4	0.9	1.3	0.7	2.9	0.4	1.2	2.2	

Table 12: Per-benchmark primary scores (%), DeepSeek-V2-Lite.

	Science QA				Commonsense			Reading/NLI		Knowl.	Math		
arm	ARC-E	ARC-C	OBQA	SciQ	HS	PIQA	WG	BoolQ	RTE	MMLU	GSM8K	MATH	avg.
Full model	76.2	48.7	43.4	94.8	78.0	80.4	71.0	79.8	61.7	58.0	40.6	16.8	62.5
Top-$K/2$ (plain skipping)	73.3	45.9	42.6	94.6	76.2	79.7	69.2	76.2	60.6	55.6	34.2	13.2	60.1
Renormalized top-$K/2$	71.8	45.5	43.8	92.9	76.1	79.2	67.1	70.2	55.6	54.2	30.4	12.2	58.2
Top-p routing (Huang et al., 2024)	74.3	45.6	42.4	94.6	76.7	80.3	70.4	77.1	60.3	56.5	35.0	16.2	60.8
Training-free halving (Chen & Yao, 2026)	73.3	45.9	42.6	94.6	76.2	79.7	69.2	76.2	60.6	55.6	34.2	13.2	60.1
LDA (Kim et al., 2026)	73.2	45.6	42.4	94.9	75.8	79.3	69.1	77.0	61.4	55.5	32.8	14.0	60.1
MoNE (Zhang et al., 2026)	73.1	45.1	41.6	94.9	74.9	79.3	70.7	77.2	61.7	55.0	30.9	13.2	59.8
LoRA (Hu et al., 2022)	74.0	46.4	43.0	94.7	76.5	79.7	69.9	76.4	61.4	55.7	37.5	15.2	60.9
Absorber only	73.8	46.8	43.4	94.7	76.6	79.7	69.0	76.8	61.0	55.7	35.4	15.6	60.7
CCQ (ours)	74.8	45.6	44.4	95.1	76.2	79.3	70.3	77.8	61.4	56.3	38.1	15.4	61.2
Keep $K/2$+1 + absorber	75.5	47.2	43.0	95.0	77.4	80.1	71.0	78.2	62.8	57.3	38.1	16.2	61.8
s.e. (ours)	0.9	1.5	2.2	0.7	0.4	0.9	1.3	0.7	2.9	0.4	1.3	1.6	

Table 13: Per-benchmark primary scores (%), Qwen3.6-35B-A3B.

	Science QA				Commonsense			Reading/NLI		Knowl.	Math		
arm	ARC-E	ARC-C	OBQA	SciQ	HS	PIQA	WG	BoolQ	RTE	MMLU	GSM8K	MATH	avg.
Full model	72.1	55.5	44.0	96.9	83.5	82.6	74.4	85.0	76.2	82.3	94.2	11.0	71.5
Top-$K/2$ (plain skipping)	72.3	54.4	42.4	96.8	81.6	82.2	71.9	81.4	75.1	80.5	93.5	10.2	70.2
Renormalized top-$K/2$	66.9	49.1	41.2	92.3	81.4	81.1	65.4	77.5	54.9	76.1	92.7	3.2	65.2
Top-p routing (Huang et al., 2024)	73.3	53.2	42.8	96.5	82.0	82.7	73.6	80.1	74.4	80.1	91.9	9.6	70.0
Training-free halving (Chen & Yao, 2026)	72.3	54.4	42.4	96.8	81.6	82.2	71.9	81.4	75.1	80.5	93.5	10.2	70.2
LDA (Kim et al., 2026)	71.8	53.3	42.8	96.5	82.0	81.8	73.0	80.9	68.2	79.9	92.4	9.2	69.3
MoNE (Zhang et al., 2026)	73.1	53.2	41.6	96.5	81.1	81.8	74.3	81.8	77.3	79.7	93.3	12.0	70.5
LoRA (Hu et al., 2022)	74.0	55.2	43.2	96.6	81.8	82.1	74.6	85.5	75.8	80.6	92.9	10.8	71.1
Absorber only	73.4	53.5	41.6	96.5	81.5	82.2	74.1	83.1	75.5	79.6	94.0	11.0	70.5
CCQ (ours)	73.5	54.9	42.0	96.5	81.5	82.3	73.3	85.5	77.6	79.4	92.3	12.2	70.9
Keep $K/2{+}1$ + absorber	72.6	54.2	43.2	96.4	82.4	82.6	73.5	85.0	76.5	80.5	91.4	11.6	70.8
s.e. (ours)	0.9	1.5	2.2	0.6	0.4	0.9	1.2	0.6	2.5	0.3	0.7	1.5	

Table 14: Ours minus the absorber alone, LoRA and the full model: macro average and GSM8K/MATH-500 (points), paired item bootstrap 95% intervals. Before rounding, the average against the absorber alone is $+0.53$ $[0.01, 1.04]$ on DeepSeek-V2-Lite, the only average interval against the absorber alone or LoRA that excludes zero, and $+0.41$ $[-0.04, 0.85]$ on Qwen3.6.

model	ours minus	12-task average	GSM8K	MATH-500
Moonlight	absorber only	+0.2 [-0.3, +0.7]	-1.4 [-3.3, +0.5]	+1.8 [-1.8, +5.4]
	LoRA	+0.2 [-0.3, +0.7]	-1.5 [-3.5, +0.5]	+2.8 [-0.6, +6.2]
	full model	-1.2 [-1.8, -0.7]	-4.4 [-6.4, -2.4]	-2.4 [-6.4, +1.6]
DeepSeek-V2-Lite	absorber only	+0.5 [+0.0, +1.0]	+2.7 [+0.5, +4.9]	-0.2 [-3.0, +2.4]
	LoRA	+0.4 [-0.2, +0.9]	+0.6 [-1.6, +2.9]	+0.2 [-2.6, +2.8]
	full model	-1.2 [-1.8, -0.6]	-2.4 [-4.8, -0.2]	-1.4 [-4.2, +1.4]
Qwen3.6	absorber only	+0.4 [-0.0, +0.9]	-1.7 [-2.8, -0.5]	+1.2 [-0.6, +3.2]
	LoRA	-0.2 [-0.6, +0.3]	-0.5 [-1.7, +0.6]	+1.4 [-0.8, +3.4]
	full model	-0.6 [-1.0, -0.1]	-1.9 [-3.1, -0.7]	+1.2 [-0.8, +3.2]

E DECODE PATH VERSUS GENERAL PATH

KL and lm-evaluation-harness's loglikelihood tasks run prefill-sized batches through the general path; generation (GSM8K, MATH-500) and the timing run decode steps through the decode kernels. Both paths compute the same function up to floating-point summation order. Routing and selection are discrete, however, so summation order can flip near-ties. To measure this effect, we feed test-pool chunks of 8 and 48 tokens, one chunk per engine step, through both paths and compute the full-vocabulary KL between them (Table 15). For ours, the two paths differ by $6.6 \cdot 10^{-4}$ to $7.3 \cdot 10^{-3}$. This is 1–11% of its KL to the full model on the same steps, and of the same order as for the full model itself (up to $7.6 \cdot 10^{-3}$). The difference thus reflects floating-point reordering rather than a different function.

Table 15: KL between the decode path and the general path of the same arm on decode-sized steps. For scale, parentheses give the KL between the full model and the arm (general path). "Plugin, $k = K$": our kernels with no expert skipped.

arm	chunk	Moonlight	DeepSeek-V2-Lite	Qwen3.6
full model	8	1.4e-03	7.7e-04	7.6e-03
full model	48	0.0e+00	0.0e+00	0.0e+00
plugin, $k = K$	8	1.4e-03 (0.000)	7.6e-04 (0.000)	7.6e-03 (0.000)
plugin, $k = K$	48	1.3e-03 (0.000)	9.1e-04 (0.000)	6.1e-04 (0.000)
plain skipping	8	3.8e-03 (0.156)	1.8e-03 (0.046)	1.1e-02 (0.091)
plain skipping	48	8.6e-03 (0.313)	3.2e-03 (0.074)	1.0e-03 (0.117)
CCQ (ours)	8	2.9e-03 (0.059)	1.4e-03 (0.030)	7.3e-03 (0.069)
CCQ (ours)	48	3.0e-03 (0.072)	2.2e-03 (0.050)	6.6e-04 (0.083)

F KERNELS AND TIMING

All kernels are CUDA kernels, written in Triton or CUDA C++ (nvcc for sm_120). We check each against PyTorch references on real weights. The per-row relative error against fp32 references is below $5 \cdot 10^{-3}$ on average and 10^{-2} at most, the level of the bf16 reference paths. Routing is bit-exact to vLLM given the logits, and CUDA-graph replay is bitwise equal to eager execution. We time kernels as CUDA graphs, with weights cycled through more distinct copies than the 128 MB L2 cache holds. The GPU is an RTX PRO 6000 Blackwell Server Edition (GDDR7, 188 SMs). The largest weight-streaming GEMV in the model, the LM head, reaches 1.53 TB/s, which we use as the achievable DRAM bound (the nominal peak is 1.79 TB/s). Table 16 lists the kernels.

Table 16: Kernels on the Server Edition GPU (cold weights, CUDA graphs): reference implementation, shape, time of the reference and of the kernel (μs), and speedup. Moonlight shapes unless noted ($d = 2048$, $I = 1408$, $K = 6$, $k = 3$); S: sketch, A: absorber term (fused into the down kernel in this benchmark; deployed, it runs on the second stream).

kernel	reference	shape	ref.	kernel	speedup
energy selection (Triton)	PyTorch	Moonlight, $T = 1$	57.3	4.8	11.94×
		Moonlight, $T = 2048$	130.1	14.5	8.97×
		Qwen3.6, $T = 1$	57.6	4.6	12.52×
		Qwen3.6, $T = 2048$	177.8	22.5	7.90×
sketch (Triton)	vLLM fused MoE	$T = 1$	15.7	4.7	3.34×
		$T = 8192$	197.5	125.3	1.58×
absorber (Triton)	cuBLAS + PyTorch	Moonlight, $T = 1$	36.1	12.3	2.93×
		Moonlight, $T = 32$	40.2	12.6	3.19×
		Qwen3.6, $T = 1$	23.8	5.6	4.25×
		Qwen3.6, $T = 32$	30.2	5.8	5.21×
routing + selection (CUDA)	vLLM router + general selection	Moonlight, $T = 1$	7.3	4.3	1.68×
		Qwen3.6, $T = 1$	9.4	5.7	1.65×
decode MoE (CUDA/Triton)	general-path kernels	Moonlight, $k = 3$+S+A, $T = 1$	48.8	37.4	1.31×
		Moonlight, $k = 3$+S+A, $T = 8$	260.8	247.5	1.05×
		Qwen3.6, $k = 4$+S+A, $T = 1$	28.7	19.6	1.47×
		Moonlight, $k = K$, $T = 1$	76.1	71.9	1.06×
small GEMM	cuBLAS	LM head, Moonlight, $T = 1$	450.4	438.9	1.03×
		LM head, Qwen3.6, $T = 1$	680.9	664.8	1.02×
		shared gate/up, Moonlight, $T = 1$	18.0	16.5	1.09×
		shared down, Qwen3.6, $T = 1$	4.3	3.0	1.45×
KL fwd+bwd (Triton)	PyTorch	V=163,840, $T = 4096$	264.5 ms	47.8 ms	5.53×
fused layer (not deployed)	separate kernels	Moonlight, $T = 1$	72.7	63.3	1.15×
		Qwen3.6, $T = 1$	41.5	28.5	1.46×

General regime (prefill and large batches). *Energy selection*: one Triton kernel replaces about fifteen PyTorch operations. Per token, it computes the K predictor dot products, the top-k of $\hat{v}$ (ties to the lower slot, returned in router order), the sketch slot and the absent mass. *Sketch*: instead of a second fused-MoE call, a one-slot SwiGLU kernel runs a GEMV path for few tokens and a grouped path that keeps the sketch intermediate on chip. *Absorber*: the shared expert's down projection and $\zeta = V^\top h$ share one GEMM on the stacked weight $[W_{\text{down}}; V^\top]$, and one kernel adds $s\, m_T U \zeta$ to the shared output. *Routed experts*: vLLM's Triton fused MoE kernel with configurations tuned for this GPU (19 token buckets, per model shape). The tuned kernels are deterministic and faster than vLLM's FlashInfer backend at small batch, so every arm uses them, stock vLLM included. *KL*: a fused LM-head and online-softmax kernel computes the full-vocabulary KL and NLL, forward and backward for distillation, without materializing the logits.

Decode regime. *Routing*: one CUDA C++ kernel computes the gate logits, the model's own routing (grouped sigmoid top-K with correction bias for Moonlight, softmax top-K for the others) and the energy selection. *Routed experts and sketch*: two launches, without an alignment pass. A persistent gate/up kernel groups the (token, slot) pairs by expert in shared memory and streams each active expert's rows once. A down kernel, launched with programmatic dependent launch, combines the routed slots and the sketch. Dispatch is per shape and fixed inside a CUDA graph: Triton at one token (CUDA C++ down kernel for $I = 1408$), CUDA C++ up to 4–8 tokens, and the tuned fused MoE above that. *Shared expert and absorber*: both run on vLLM's shared-expert stream, as the shared expert does in stock vLLM, with one cross-stream event for the absent mass. *Small GEMMs*

($M \leq 16$): a table of CUDA C++ and Triton GEMV kernels serves the LM head and the dense projections and is never slower than cuBLAS on the decode shapes. Stock vLLM (the full model and top-$K/2$ in Table 18) uses cuBLAS instead, so part of our lead over stock top-$K/2$ at small batch sizes comes from this table and the decode MoE kernels, not from the method. Figure 6 shows one MoE layer at decode.

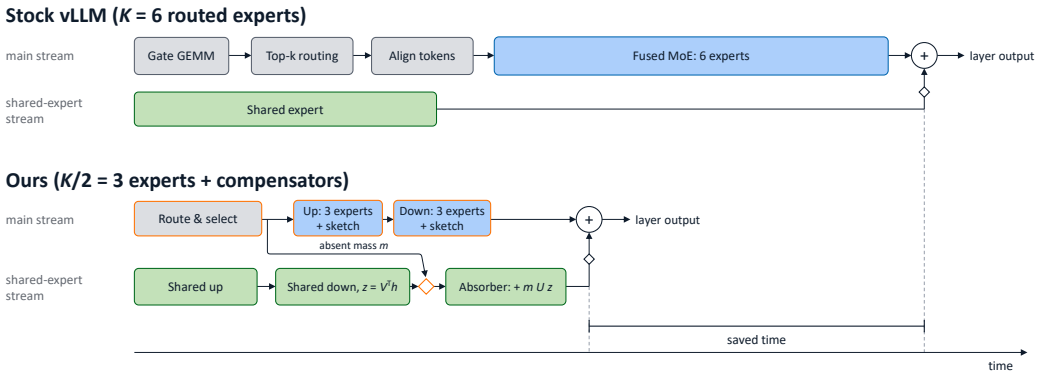


Figure 6: One MoE layer at decode in vLLM (schematic, not to scale). Stock vLLM overlaps the shared expert with gate, routing, alignment and the fused MoE over K experts. The plugin fuses gate, routing and selection into one kernel, streams the $K/2$ executed experts and the sketch in two kernels, and runs the shared expert and the absorber on the second stream; the absorber waits for the absent mass m (in the figure $m = m_T$, $z = \zeta$ and the scale s is omitted).

Where the time goes. At batch size 1 on Moonlight, the decode step is GPU-bound. We profile one run per engine (1,024-token context, CUDA graphs, asynchronous scheduling). Kernels run for the whole step, both for stock vLLM (5.13 ms per token under the profiler; summed kernel time over both streams 5.29 ms per step) and for the method (4.18 ms; 4.29 ms). Per MoE layer, the method streams 88.7 MB (three routed experts, the shared expert, the sketch and the absorber) in about 78 μs (1.1 TB/s); stock vLLM streams 138.4 MB in about 116 μs (1.2 TB/s). Layer time is the routed-expert kernels on the main stream plus the time when only the second stream is busy (0.69 ms per step for stock vLLM, 0.06 ms for the method). The MoE layers thus account for 0.98 ms of the 1.00 ms per step by which the GPU busy time falls. Attention (vLLM's two-kernel Triton decode for multi-head latent attention, 21 μs per layer), the dense projections and the LM head (0.44 ms) are nearly the same for every arm (ours uses the small-GEMM table) and bound the end-to-end speedup (Amdahl's law).

Limit. The next level of fusion is not faster end to end. We tested a single-stream MoE layer in which the gate/up kernel also streams the shared expert's rows and the down kernel also computes the shared down projection and ζ (routed output bitwise equal, shared output within $2.1 \cdot 10^{-3}$ of vLLM's). In isolation it runs at 92% of the byte bound (Moonlight, one token: 63.3 μs vs 72.7 for the separate kernels, bound 58.0 at 1.53 TB/s). End to end, however, it is not faster than the two-stream schedule (in a separate paired run at batch size 1: Moonlight 242 vs 242 tokens/s, Qwen3.6 197 vs 205), because the second stream already hides the shared expert. The deployed plugin therefore keeps the two-stream schedule. Further gains would need kernels that span attention and the MoE layer, or several layers.

Timing protocol. Throughput is measured with vLLM's official benchmark (`vllm bench throughput`, vLLM 0.29, V1 engine, CUDA graphs, torch.compile, asynchronous scheduling, prefix caching off, a single GPU, no tensor parallelism): 256 random prompts of 256 tokens, each generating exactly 256 tokens (`ignore_eos`), are submitted at once, and the engine's batch limit `-max-num-seqs` is set to $b \in \{1, 4, 16, 64, 256\}$. We report the output tokens per second of one run per setting, on a GPU that runs nothing else. The full model and top-$K/2$ skipping run on stock vLLM with the tuned fused-MoE configurations, the latter with the model's number of routed experts per token set to $K/2$ (on Moonlight and Qwen3.6, whose routers renormalize the top-K

Table 17: Kernel latency (μs; the values of Figure 4) of the routed part of one MoE layer at decode (cold weights, CUDA graphs): routing and routed experts, plus selection, sketch and absorber term (fused into the down kernel here) for ours; the shared expert runs in parallel on a second stream in all three and is excluded. Full model and top-$K/2$: vLLM's routing and Triton fused MoE (tuned for this GPU). DeepSeek-V2-Lite (Moonlight's expert shapes, different routing) is not shown. Bold: fastest.

	Moonlight		Qwen3.6	
	batch 1	batch 8	batch 1	batch 8
full model (K experts)	80.8	381.1	47.6	256.6
top-$K/2$ skipping	47.1	264.2	29.6	**141.2**
CCQ (ours)	**41.7**	**252.2**	**25.3**	147.0

Table 18: Output throughput on one GPU, the values of Figure 3 (vLLM's official benchmark; 256 prompts of 256 tokens, 256 generated tokens each): tokens/s of the full model on stock vLLM and speedups over it at batch limit b. Top-$K/2$ runs on stock vLLM with $K/2$ routed experts per token (renormalized on Moonlight and Qwen3.6, hence its KL); training-free halving and a merged LoRA cost the same, LDA and MoNE add one vector operation per layer, and Top-p routing was not timed.

model		test KL	$b=1$	$b=4$	$b=16$	$b=64$	$b=256$
Moonlight	full model (stock vLLM), tokens/s	0	203	461	1214	3205	8128
	top-$K/2$ (stock vLLM)	0.2303	1.20×	1.31×	1.23×	1.13×	1.07×
	CCQ (ours)	0.0711	1.22×	1.42×	1.32×	1.20×	1.10×
DeepSeek-V2-Lite	full model (stock vLLM), tokens/s	0	210	445	1036	2716	7594
	top-$K/2$ (stock vLLM)	0.0551	1.21×	1.38×	1.39×	1.26×	1.04×
	CCQ (ours)	0.0397	1.23×	1.48×	1.43×	1.23×	1.06×
Qwen3.6	full model (stock vLLM), tokens/s	0	162	377	806	1674	2582
	top-$K/2$ (stock vLLM)	0.2060	1.12×	1.26×	1.36×	1.27×	1.21×
	CCQ (ours)	0.0690	1.19×	1.35×	1.40×	1.31×	1.15×

weights, this computes renormalized top-$K/2$ at the cost of plain skipping). Ours runs on the same vLLM with our plugin, decode MoE kernels and small-GEMM table.

Multi-GPU serving. Our plugin runs on a single GPU. To check whether skipping half of the routed experts still pays under tensor and expert parallelism, we run the same benchmark on stock vLLM for DeepSeek-V2-Lite with tensor parallelism TP$=2$ and 4, without and with expert parallelism (EP, `-enable-expert-parallel`), and compare the full model with top-$K/2$ (Table 19; one run per setting, default fused-MoE configurations for the sharded shapes). Skipping still pays in every setting: top-$K/2$ is 1.14–1.29× as fast as the full model at $b=16$ and 1.03–1.11× at $b=256$ (1.39× and 1.04× on one GPU); on this server, four GPUs are slower than two at $b=256$. CCQ was not run in these settings.

Table 19: Multi-GPU scaling of half skipping on stock vLLM (DeepSeek-V2-Lite, workload of Table 18): full-model tokens/s and top-$K/2$ speedup at batch limit b.

	$b=16$		$b=256$	
GPUs	full model	top-$K/2$	full model	top-$K/2$
1 GPU	1036	1.39×	7594	1.04×
TP=2	1571	1.29×	8810	1.11×
TP=2 + EP	1287	1.23×	9389	1.07×
TP=4	1593	1.14×	5908	1.03×
TP=4 + EP	1558	1.16×	5429	1.11×