TRAIN ON THE STATES YOU DEPLOY: CLOSED-LOOP HARD ROUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

A 31M learned sparse-attention model with native perplexity 49 reaches perplexity 291 when deployed with the hard routes it was trained to produce. Soft-only training omits the hard deployment-state distribution; because deployed routes are generated layer by layer, that distribution is closed-loop. We introduce *Closed-Loop Hard Routing* (CLHR), which keeps a differentiable soft path for router learning while training the backbone on the hard trajectory induced by its own decisions. In an exact reproduction of a public 31M-parameter protocol, CLHR reduces closed-loop deployment excess from 1.782 to 0.0035 nats across three seeds, changing the hard/native perplexity ratio from $5.94\times$ to $1.0035\times$. With the configuration frozen, the reduction transfers to 300M models on WikiText-103 ($0.796 \to 0.049$) and FineWeb-Edu ($0.513 \to 0.027$); on LAMBADA, standard hardening reduces accuracy from 13.2% to 3.7%, whereas CLHR changes it from 13.9% to 12.9%. A matched causal ladder shows that hard exposure, open-loop replay, and shuffled routing do not suffice. Across the tested settings, successful soft-to-hard deployment requires task supervision on useful, self-induced hard states.

1 INTRODUCTION

Learned sparse attention promises to avoid unnecessary token interactions, but training and deployment often use different computations. Training uses continuous gates so routing parameters receive gradients; deployment requires discrete top-k routes. Soft training therefore omits the hard deployment states, and those states are recursively generated rather than fixed: once a hard decision changes a layer's hidden state, every later router receives a different input.

Aquino-Michaels (2026) shows that end-to-end soft gates can be absorbed by query, key, and value projections, and uses post-hoc decoupling to avoid the resulting co-adaptation. We isolate a further failure inside hard deployment: the non-router backbone has never encountered the state trajectory induced by its own preceding discrete routes. Extracting masks from a completed soft trajectory and replaying them afterward does not reproduce this closed loop. SSA instead jointly trains full and sparse streams with feature alignment (Shen et al., 2026). In our learned-token-router adaptation, its hard-stream task supervision closes the gap, while removing alignment has no measurable effect. CLHR retains the jointly learned router but covers its complete self-induced hard trajectory deterministically, without alignment or a post-hoc replacement stage.

We introduce *Closed-Loop Hard Routing* (CLHR). An ordinary soft branch trains the router and backbone. A second branch selects detached hard routes layer by layer from its own states, exposing the shared backbone to deployed execution. The final method weights the two task losses equally. It neither differentiates through top-k nor freezes and refits the router.

The result transfers across corpus and scale with the configuration frozen:

$$\begin{aligned} &\text{31M WikiText-103}: && 1.782 \to 0.0035, \\ &\text{300M WikiText-103}: && 0.796 \to 0.049, \\ &\text{300M FineWeb-Edu}: && 0.513 \to 0.027. \end{aligned}$$

At 31M, every final CLHR seed also beats every SSA-style seed in absolute deployed NLL L_H.

Anonymous code: `anonymous.4open.science/r/clhr-EFC1/`

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.

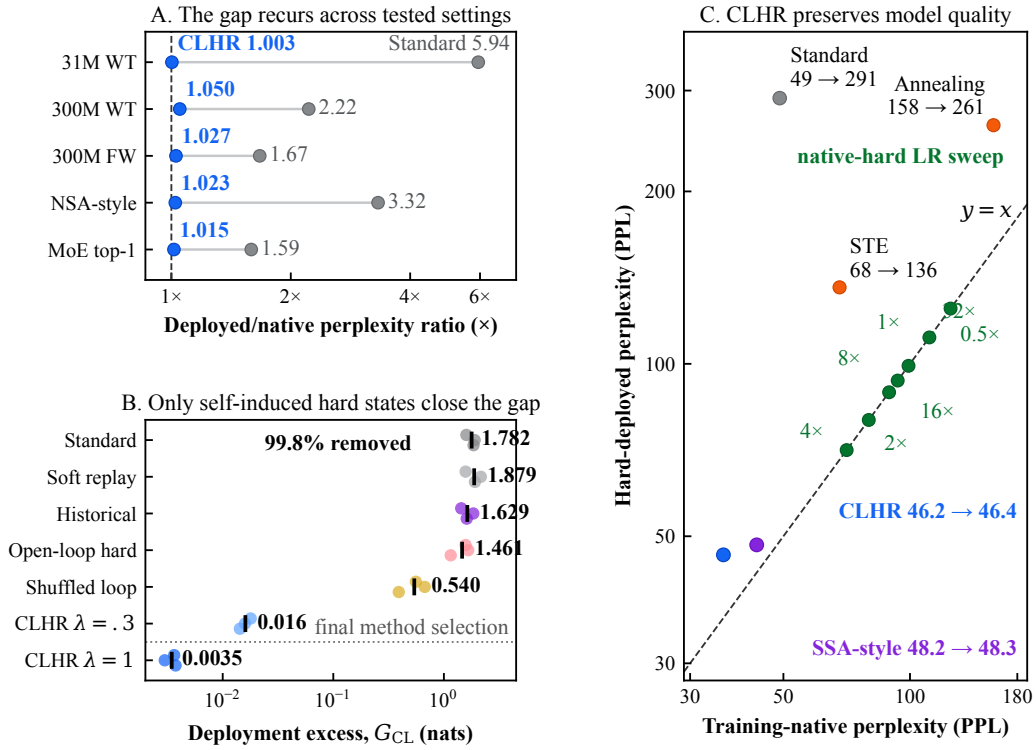


Figure 1: **Closed-loop hard-state training makes learned soft routers directly deployable. A:** Standard soft training leaves a substantial hard-deployment gap across every tested scale, corpus, and routing family; CLHR reduces deployed/native perplexity to within 5% of parity in all five settings. **B:** Controlled interventions show that hard exposure alone and closed-loop exposure without input conditioning are insufficient; contemporaneous, input-conditioned hard states remove 99.8% of the 31M deployment excess. **C:** This robustness does not arise by sacrificing model quality: CLHR retains substantially lower perplexity than native-hard, straight-through, and annealed alternatives.

This study grew from broader routing-context experiments: they motivated us to separate which route trains shared computation from which state distribution generates that route; Appendix F summarizes this lineage.

Our contributions are:

- **Diagnosis.** We isolate missing coverage of recursively generated hard deployment states and show, via a six-condition causal ladder over three seeds, that only contemporaneous, input-conditioned hard-state exposure removes the deployment excess (Section 3.1).
- **Method.** CLHR is a two-path objective with exact local gradient separation: the router learns on the soft path and the backbone on both soft and self-induced hard trajectories, without surrogate gradients, annealing, or alignment (Sections 2.1–4).
- **Evidence.** The deployment excess and remedy transfer across scale, corpus, and tested routing families, and to downstream accuracy; we also identify routing schemes where native-hard training is a viable alternative (Section 6).

2 Soft-to-Hard Routing Is a Closed-Loop Shift

Consider a transformer whose layer transition F_ℓ depends on a hidden state and a learned attention route. Throughout the primary token-routing experiments, *native* means differentiable soft execution and *deployed* means one-pass, self-consistent hard execution. Superscripts S, H, and OL denote soft, closed-loop hard, and open-loop hard execution, respectively. A *route* m_ℓ is the routing object

at layer ℓ: continuous gate weights under S and a binary causal support under H or OL. It is distinct from the resulting normalized attention distribution. Native soft execution follows

$$m_\ell^S = \sigma(g_{\phi,\ell}(h_\ell^S)), \qquad h_{\ell+1}^S = F_\ell(h_\ell^S, m_\ell^S). \tag{1}$$

In cross-family comparisons, *training-native* instead names the execution used to train that arm; it is hard for an explicitly labeled native-hard arm. At deployment, a sparse implementation instead selects a discrete route from the state produced by earlier hard decisions:

$$m_\ell^H = \mathrm{TopK}_k(g_{\phi,\ell}(h_\ell^H)), \qquad h_{\ell+1}^H = F_\ell(h_\ell^H, m_\ell^H). \tag{2}$$

Here $g_{\phi,\ell}$ is the learned router, σ is its continuous gating map, and TopK_k returns the binary causal mask containing the $\min(k, t+1)$ largest eligible scores for a query at zero-indexed position t. Hardening is not an independent approximation at each layer. Once $h_\ell^H \neq h_\ell^S$, subsequent masks are selected from a state trajectory that the soft model did not encounter during training.

An open-loop evaluation does not reproduce this feedback. It first extracts $\bar{m}_\ell = \mathrm{TopK}_k(g_{\phi,\ell}(h_\ell^S))$ from a complete soft rollout and then replays those masks on a separate hard trajectory. The masks are generated from soft teacher states even after the evaluated states have diverged. We distinguish this diagnostic from the one-pass closed-loop execution in Equation 2 throughout. With corrected binary masks, open-loop evaluation nevertheless captures most of the standard model's deployment excess (1.470 versus 1.782 nats). The distinction therefore identifies the deployed computation and state distribution; it does not imply that closed-loop feedback must greatly magnify the loss in every condition.

Let L_S, L_H, and L_{OL} denote token-weighted NLL under native soft, closed-loop hard, and open-loop hard execution. Our primary endpoint is closed-loop deployment excess,

$$G_{\mathrm{CL}} = L_H - L_S, \tag{3}$$

and the open-loop diagnostic is

$$G_{\mathrm{OL}} = L_{\mathrm{OL}} - L_S. \tag{4}$$

Its perplexity interpretation is $\mathrm{PPL}_H \,/\, \mathrm{PPL}_S = \exp(G_{\mathrm{CL}})$, where PPL denotes perplexity. We use *deployment excess* for G_{CL} in prose; "gap" in compact figures and tables denotes the same quantity. Router utility is $U_{\mathrm{route}} = L_{\mathrm{random},H} - L_H$, where the random route has the same causal cardinality; $U_{\mathrm{route}} > 0$ means the learned route is better than this balanced-random control.

2.1 Closed-Loop Hard Routing

Let ϕ denote router parameters and θ all non-router backbone parameters. CLHR minimizes

$$\mathcal{L}_{\mathrm{CLHR}}(\theta, \phi) = \frac{\mathcal{L}_S(\theta, \phi) + \lambda \mathcal{L}_H\left(\theta, \mathrm{sg}[m_{\theta,\phi}^H]\right)}{1+\lambda}, \tag{5}$$

where $\lambda \geq 0$ is the hard-loss weight, sg is stop-gradient, and every m_ℓ^H is generated online from h_ℓ^H as in Equation 2. The soft branch is the native gated-attention model, including its sparsity objective. The hard branch recomputes queries, keys, and values from its current hidden state, selects the top k gate scores, applies the binary causal mask immediately, and advances to the next layer. Here and below, calligraphic $\mathcal{L}_S$ and $\mathcal{L}_H$ denote minibatch training objectives, whereas roman L_S and L_H denote evaluated token-averaged NLLs.

The resulting gradients are

$$\nabla_\phi \mathcal{L}_{\mathrm{CLHR}} = \frac{1}{1+\lambda} \nabla_\phi \mathcal{L}_S, \tag{6}$$

$$\nabla_\theta \mathcal{L}_{\mathrm{CLHR}} = \frac{\nabla_\theta \mathcal{L}_S + \lambda \nabla_\theta \mathcal{L}_H}{1+\lambda}. \tag{7}$$

The router is optimized through the differentiable soft branch, while the backbone is optimized jointly on native soft execution and its self-consistent hard deployment trajectory.

No mask is precomputed from another trajectory. We select $\lambda = 1$ on the development seed from a normalized $\{0.3, 0.5, 1, 2\}$ sweep, then freeze it for seeds 123 and 456, FineWeb-Edu, and the 300M replication. Thus the final objective is $(\mathcal{L}_S + \mathcal{L}_H)/2$. Division by $1+\lambda$ keeps the overall loss scale comparable across the sweep.

Closed-Loop Hard Routing for a mini-batch x

1. Compute the native soft loss $\mathcal{L}_S(x; \theta, \phi)$.
2. Initialize the hard-path state h_0^H from the same tokens.
3. For each layer $\ell = 0, \ldots, L-1$:
   (a) $s_\ell \leftarrow g_{\phi,\ell}(h_\ell^H)$,
   (b) $m_\ell^H \leftarrow \mathrm{sg}[\mathrm{TopK}_k(s_\ell)]$,
   (c) $h_{\ell+1}^H \leftarrow F_{\theta,\ell}(h_\ell^H, m_\ell^H)$.
4. Compute $\mathcal{L}_H$ from h_L^H and update with $(\mathcal{L}_S + \lambda\mathcal{L}_H)/(1+\lambda)$.

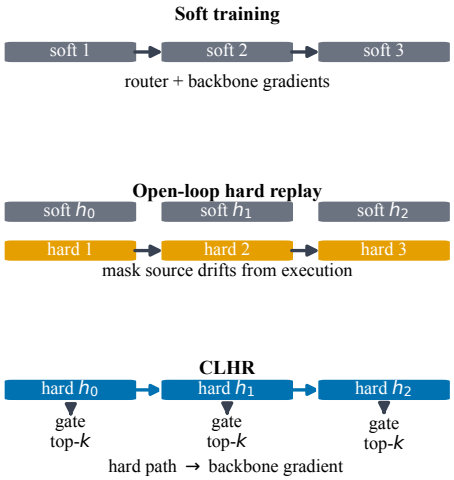


Figure 2: **Algorithm and training trajectories.** The algorithm trains the router through continuous gates and the backbone through both paths. Open-loop replay draws masks from soft teacher states; CLHR selects every detached route from the current hard state.

Hard-path subbatching (CLHR-Lite). Let $B_H \subset B$ be a random hard-path subbatch with $q = |B_H|/|B|$. Every optimizer step evaluates the soft loss on the complete minibatch B and the hard loss on B_H. This separates *cadence* (whether an update contains hard-state supervision) from *breadth* (how many examples receive it). We keep a hard subbatch on every update and vary $q \in \{0.25, 0.5, 0.75, 1\}$; $q = 1$ recovers full CLHR.

Distinctions. Unlike straight-through estimation, CLHR supplies no surrogate derivative through the discrete choice. The native branch remains soft throughout training, so there is no temperature schedule toward hard routing. Each hard mask is selected from the state on which it acts, rather than replayed from a teacher trajectory. The original router and backbone remain jointly trained; no post-hoc replacement stage is used.

3 Experiments

Exact 31M protocol. The primary setting matches the public routing-absorption experiment: a six-layer causal transformer with width 256, four attention heads, feed-forward width 1024, gate width 32, and 512-token sequences. The model uses the GPT-2 50,257-token vocabulary (Radford et al., 2019) and is trained on WikiText-103 (Merity et al., 2017) for 50K steps with AdamW (Loshchilov & Hutter, 2019), learning rate 3×10^{-4}, weight decay 0.1, 1K warmup steps, and batch size 16. Thus each run processes 409.6M training tokens. The soft-gate sparsity penalty is computed exactly as in the reference implementation: sparsity is averaged across layers before applying the squared deviation from target sparsity. All methods deploy with $k = 64$ at length 512. This is 87.5% nominal full-context sparsity; because early causal queries have fewer than 64 eligible keys, the selected causal-pair density is $30{,}752/131{,}328 = 23.4\%$, or approximately 76.6% effective causal-pair sparsity.

True one-pass evaluation. Closed-loop evaluation recomputes the gate from the currently propagated hard state at every layer. We aggregate cross-entropy by token before taking the dataset mean. An independent evaluator reconstructs the forward pass from raw model parameters without importing the training or primary evaluation code. Tests verify that hard masks contain only zeros and ones and that every query selects exactly $\min(k, t+1)$ causally valid keys. Extracting all masks from a soft pass and replaying them later measures G_{OL}, not the primary endpoint in Equation 3.

Comparisons. Our causal ladder crosses two properties of the auxiliary execution: whether masks are generated open- or closed-loop, and whether routes preserve the input-conditioned score ordering. *Contemporary* uses the current router; *historical* uses an earlier checkpoint; *replay* fixes routes

produced by a teacher trajectory; and *shuffled* breaks the input–route pairing while preserving route cardinality. *Standard* denotes the reference soft-gated training recipe without a hard auxiliary trajectory. We additionally compare straight-through estimation (STE), temperature annealing, contemporary soft replay, coherent historical soft routing, balanced random routing, post-hoc Kullback–Leibler (KL) gate fitting, and an SSA-style adaptation to learned token routing. *Native-hard* means hard routing for both training and evaluation from scratch. STE and annealing are selected on seed 42 and frozen for seeds 123 and 456; SSA-style alignment is selected and frozen the same way. Native-hard token routing and the three-arm NSA-style comparison use the standard recipe without learning-rate retuning. Every table states the seed count for each comparison.

Cross-corpus setting. The 324,803,584-parameter FineWeb-Edu test processes 999,948,288 tokens from `HuggingFaceFW/fineweb-edu`, `sample-10BT`, revision `v1.0.0` (HuggingFaceFW, 2024; Penedo et al., 2024), and uses a disjoint held-out shard. Seeds 42 and 123 use the frozen $k = 64$, $\lambda = 1$ method. Appendix H documents packing and additional cross-corpus replication.

Scale setting. The 300M model has 20 layers, width 1024, 16 attention heads, feed-forward width 4096, and gate width 32. It retains sequence length 512 and $k = 64$. It uses the GPT-2 tokenizer and cycles the filtered nonempty text of WikiText-103 raw v1 to four billion processed tokens; the corpus repeats rather than containing four billion unique tokens. This controlled corpus isolates transfer across parameter width, depth, and optimization duration; it is not a claim of corpus-scale transfer to diverse web pretraining. The final standard and equal-weight CLHR runs both process 65,536 tokens per optimizer step for 61,035 updates. We evaluate three seeds per condition and report the single dense seed only as context. Appendix I gives the complete optimizer, packing, checkpoint, precision, and hardware configuration.

Evidence hierarchy. Seed 42 is used for method development; seeds 123 and 456 are frozen confirmations. Unless explicitly marked $n = 1$ or $n = 2$, headline means use three seeds. FineWeb-Edu and the 300M WikiText-103 experiment use the selected configuration without retuning. Unless a caption says otherwise, $\pm$ denotes the sample standard deviation across independent training seeds.

We use 31M and 300M as rounded main-text configuration labels. Where relevant, we give exact parameter counts (for example, 324,803,584 for the FineWeb-Edu model) so these labels are not interpreted as exact counts.

3.1 Exact-Protocol Causal Evidence

Protocol. We reproduce the 31M-parameter WikiText-103 setup of Aquino-Michaels (2026): GPT-2 tokenization, 50K steps, learned token-level gates, $k = 64$, and 87.5% sparsity. We report token-weighted NLL, one-pass closed-loop evaluation, and three matched seeds for every headline comparison.

Stage 1: matched causal identification. The matched $\lambda = 0.3$ controls isolate the ingredients before method optimization. Standard training gives $G_{\mathrm{CL}} = 1.782$ nats and matched CLHR gives 0.016. Open-loop replay leaves 1.461 nats because its masks come from soft teacher states; shuffling key assignments within the hard rollout leaves 0.540. Thus hard exposure helps, but only input-conditioned routes remove the remaining excess.

Historical soft routing leaves 1.629 nats and contemporary soft replay 1.879; neither diversity nor history alone explains the effect. Corrected three-seed $\lambda = 1$ reruns preserve the ordering (Appendix B).

Stage 2: final method selection. After identifying the necessary ingredients, we normalize the objective and select its relative weight on seed 42. Equal-weight CLHR is then frozen. Across three seeds it reduces mean G_{CL} from 1.782 to 0.00346 (individual values 0.00375, 0.00299, and 0.00364), changing the deployed/native perplexity ratio from $5.94\times$ to $1.0035\times$. Mean native NLL L_S and deployed NLL L_H are 3.8323 and 3.8357. We next test dependence on the two-path implementation, hard-path breadth, and corpus.

Why not use native-hard training? A native-hard control has essentially no native-to-deployed NLL change, but its default-rate native PPL is 97.8 (three-seed mean), versus 46.2 for final CLHR. On seed 42, sweeping $0.5/1/2/4/8/16/32\times$ the default rate gives PPL $124.9/99.2/79.8/70.7/89.2/93.5/111.2$. The non-monotonic response brackets a learning-rate sweet spot at the best tested $4\times$ setting, which remains $1.53\times$ CLHR's native PPL. The tradeoff is architecture dependent; native-hard is nearly free in our NSA-style adaptation (Section 6). Appendix Table 10 gives all rates and seeds.

4 Analysis: Bounding Hard–Soft State Drift

The dual-path construction follows from the local geometry of discrete routing. On a neighborhood where the causally eligible top-k support at every layer of the complete hard rollout remains fixed, detaching the hard masks does not approximate the backbone gradient: it gives the exact local gradient of the fixed-support hard objective.

Proposition 1 (Gradient separation)**.** *Let θ parameterize the backbone and ϕ the router. On any parameter-space neighborhood where every layer's causally eligible top-k support along the complete hard rollout remains fixed, $\partial m^H/\partial\theta = \partial m^H/\partial\phi = 0$. Backpropagation through a detached hard route gives the exact local backbone gradient of the hard-path loss, while providing no ordinary hard-path derivative to ϕ.*

The sensitivity of a route is controlled by its selection margin. Let $s_{(k)}$ and $s_{(k+1)}$ denote the scores at the selection boundary among the causally eligible keys for a fixed query and head, and let $\gamma_k(s) = s_{(k)} - s_{(k+1)}$.

Lemma 1 (Top-k stability)**.** *If $\|s - s'\|_\infty < \gamma_k(s)/2$, then $\mathrm{TopK}_k(s) = \mathrm{TopK}_k(s')$.*

Now define $\delta_\ell = \|h_\ell^H - h_\ell^S\|$. Let a_ℓ^S and a_ℓ^H denote the effective causal attention distributions after masking and row normalization; the symbol m_ℓ remains reserved for routing masks. Define $d(a, a')$ as mean row-wise ℓ_1 distance over eligible query–head rows. Suppose each layer is locally Lipschitz on the visited state region, with nonnegative state and attention constants α_ℓ and β_ℓ:

$$\|F_\ell(h, m) - F_\ell(h', m')\| \le \alpha_\ell \|h - h'\| + \beta_\ell d(a, a'), \tag{8}$$

where a and a' are the normalized attention distributions induced by (h, m) and (h', m'), respectively. Because the two executions share the same embedded input, $\delta_0 = 0$. Then

$$\delta_L \le \sum_{\ell=0}^{L-1} \left(\prod_{j=\ell+1}^{L-1} \alpha_j \right) \beta_\ell d(a_\ell^H, a_\ell^S). \tag{9}$$

Early disagreements can influence all remaining layers. This bound describes visited-trajectory drift; it does not assert that drift or deployment loss must increase at every layer, that closed-loop loss must substantially exceed its open-loop counterpart, or that CLHR directly minimizes the bound.

5 What Makes Closed-Loop Training Work?

Independent confirmation from SSA-style training. We adapt SSA's dual-stream recipe to our learned token router; this is not a reproduction of its block-sparse system. The soft or self-induced hard stream is propagated with probability $1/2$, while bidirectional smooth-ℓ_1 alignment matches their attention outputs. Across three seeds it reaches native NLL 3.8747 ± 0.0070, deployed NLL 3.8780 ± 0.0072, and $G_{\mathrm{CL}} = 0.00331 \pm 0.00031$. Equal-weight CLHR has essentially the same deployment excess but improves native and deployed NLL by 0.0424 and 0.0423 nats; every CLHR seed has lower deployed NLL than every SSA-style seed. Thus the adaptation independently validates hard-state exposure, while CLHR gives the better absolute model.

Alignment is not operative in this adaptation. Removing this alignment while retaining 50/50 alternation changes deployed NLL by 0.0009 ± 0.0034 nats across matched seeds, with inconsistent direction. Feature alignment therefore provides no measurable practical benefit in our learned-router adaptation; this does not make a claim about SSA's original block-sparse setting.

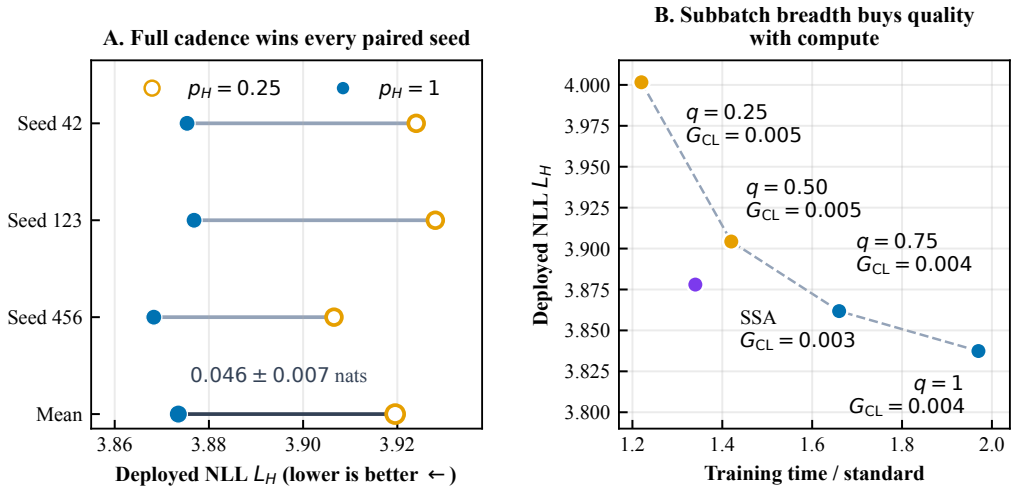


Figure 3: **What makes closed-loop training work. A:** With expected hard-loss contribution fixed, full hard-update cadence wins in every paired seed by 0.046 ± 0.007 nats. **B:** With a hard path on every update, subbatch breadth traces a quality–compute frontier; labels give G_{CL}, and SSA-style is the purple reference point. Full breadth buys the best deployed NLL, 3.837.

Every update matters. In this normalized $\lambda = 0.3$ block, we evaluate the soft loss every batch, sample the hard loss with probability p_H, and multiply it by λ/p_H, holding its expected contribution fixed. If $v_S = \nabla\mathcal{L}_S$, $v_H = \nabla\mathcal{L}_H$, and $\widehat{v}_p = v_S + Z(\lambda/p_H)v_H$, then its stream-sampling variance is $\lambda^2(1-p_H)\|v_H\|_2^2/p_H$. Here the gradients are for a fixed minibatch and $Z \sim \mathrm{Bernoulli}(p_H)$. Deployed NLL improves from 3.9195 ± 0.0114 at $p_H = 0.25$ to 3.8735 ± 0.0046 at $p_H = 1$ across three seeds; all paired differences favor full cadence, by 0.0460 ± 0.0069 nats. Matching the objective in expectation does not match finite-time optimization.

Every example does not. CLHR-Lite instead retains a hard path on every update while varying its subbatch fraction q. Across three seeds, G_{CL} remains 0.004–0.005 for every $q \in \{0.25, 0.5, 0.75, 1\}$, whereas deployed NLL improves from 4.002 to 3.837 as breadth increases. On the synchronized L40S benchmark, cost rises smoothly from $1.22\times$ to $1.97\times$ standard. Intermittently omitting entire hard updates harms quality; reducing examples within every hard update preserves deployability but trades absolute quality for compute. Figure 3 summarizes this distinction.

The development sweep selects normalized $\lambda = 1$ by deployed NLL; complete weight sweeps, router utility U_{route}, and earlier estimators appear in Appendix G. For STE and annealing, seed 42 selects the reported setting and seeds 123 and 456 use it unchanged. Main 31M comparisons use three seeds; only the native-hard learning-rate sweep is seed 42 alone.

6 Transfer, Consequences, and Boundaries

Cross-corpus and parameter-scale transfer. Without retuning, CLHR reduces mean G_{CL} from 0.5126 to 0.0270 on 300M FineWeb-Edu ($n = 2$), and from 0.7956 to 0.0488 on 300M WikiText-103 ($n = 3$); Figure 4 shows every seed. On WikiText it also improves native/deployed NLL from 2.9274/3.7230 to 2.9134/2.9623. The older $\lambda = 0.3$ arm has slightly smaller excess but worse absolute quality.

Downstream consequence at 300M. Across seeds 42/123/456 on complete LAMBADA-OpenAI (Paperno et al., 2016) and lm-eval WikiText-2, standard hardening loses 9.5 accuracy points and raises word perplexity 153.6%; CLHR changes them by 1.0 point and 5.5% (Table 2A). Appendix D specifies this distinct evaluation protocol.

Table 1: **Closed-loop deployment excess across settings.** Means use the stated seeds; "–" denotes an unevaluated quantity, not zero.

Setting	Size/budget	n	Std. G_{CL}	CLHR G_{CL}	Reduction	U_{route}: Std. → CLHR
Token, WikiText-103	31M/0.41B	3	1.782	0.0035	99.8%	$0.47 \rightarrow 0.89^{*}$
Token, WikiText-103	300M/4B	3	0.796	0.049	93.9%	–
Token, FineWeb-Edu	300M/1B	2	0.513	0.027	94.7%	$2.73 \rightarrow 3.04$
MoE top-1, WikiText-103	57M/20K steps	3	0.465	0.015	96.8%	$2.61 \rightarrow 3.41$
NSA-style adaptation, WikiText-103	31M/50K steps	3	1.201	0.023	98.1%	–
Budget-conditioned $k = 32$–$256^{\dagger}$	31M/50K steps	3	–	0.003–0.026	–	0.10–1.18

n: seeds; Std.: standard soft training. *Utility uses matched $\lambda = 0.3$; quality uses final $\lambda = 1$. †Ranges are over routing budgets, not Standard→CLHR.

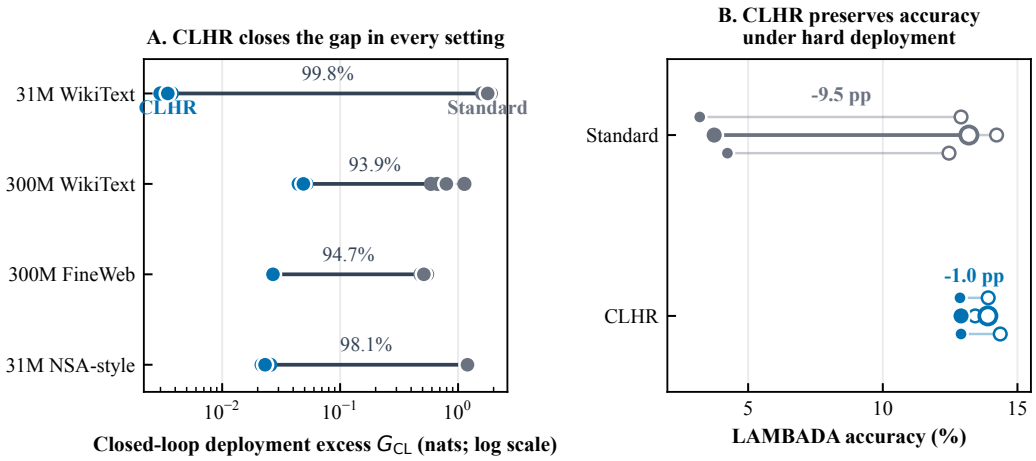


Figure 4: **Transfer and downstream consequence.** The gap and remedy transfer across scale, corpus, and our NSA-style adaptation (**A**). Standard hard deployment drops mean LAMBADA accuracy from 13.2% to 3.7%; CLHR loses one point (**B**; open: soft, filled: hard).

External routing schemes and native-hard training. In a three-seed NSA-style adaptation (Yuan et al., 2025; Wang, 2025), CLHR reduces excess from 1.201 to 0.023 nats; native-hard is competitive (Table 2).

Router utility rises under CLHR at 31M, in top-1 MoE (Shazeer et al., 2017; Fedus et al., 2022; Roller et al., 2021), and on FineWeb-Edu (Table 1); shuffled routing is negative. At $k = 256$, however, utility falls to 0.097: small excess alone does not establish useful selection.

7 Mechanism and Early Deployment Screening

Mean tokenwise angular drift reaches 0.455/0.348/0.136/0.009 for standard/open-loop/shuffled/CLHR. Open-loop evaluation captures most excess ($G_{\mathrm{CL}}/G_{\mathrm{OL}} \leq 1.8$), so we do not claim multiplicative amplification. Across eight runs at 5K/50K updates, early G_{CL} best screens final G_{CL} ($\rho = 0.976$, $R^2 = 0.996$); maximum drift is the best target-loss-free rank signal ($\rho = 0.952$, $R^2 = 0.980$). Thus early excess diagnoses failure while drift explains it (Appendices K and M).

8 Related Work

Routing Transformers learn content-based sparse patterns (Roy et al., 2021). Gradient estimators alter credit, not state coverage (Bengio et al., 2013; Jang et al., 2017; Louizos et al., 2018; Xie et al., 2020; Sander et al., 2023). Routing absorption decouples post hoc (Aquino-Michaels, 2026); SSA aligns streams (Shen et al., 2026); NSA/MoBA train hard routes natively (Yuan et al., 2025; Lu

Table 2: **Operational consequences and an NSA-style boundary.** Three-seed means; NSA-style arms are not retuned. Per-seed values and protocols: Appendix D.

A. Downstream consequence at 300M (3-seed mean)

Condition	WikiText-2 word PPL	LAMBADA accuracy
Standard	33.08 → 83.88	13.2% → 3.7%
CLHR	32.97 → 34.79	13.9% → 12.9%

B. NSA-style routing adaptation

Arm	Training-native PPL	Deployed PPL	Excess NLL
Soft standard	47.16	156.75	1.201
CLHR	46.06	47.14	0.023
Native-hard	49.30	48.86	−0.009

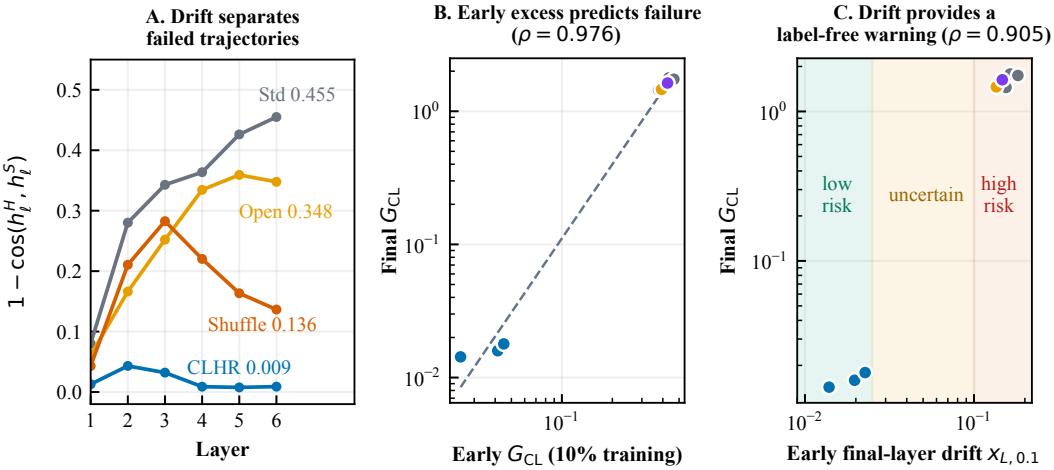


Figure 5: **Drift explains; early deployment excess diagnoses. A:** Three-seed layerwise drift. **B:** G_{CL} at 10% best screens final excess. **C:** Early final-layer drift with empirical risk bands.

et al., 2025). SeerAttention-R and SpargeAttention2 instead adapt or distill sparse execution (Gao et al., 2026; Zhang et al., 2026); SeerAttention, QUEST, and UNIQUE emphasize execution (Gao et al., 2024b; Tang et al., 2024; Deng et al., 2026). Scheduled sampling and DAgger cover states across time, not depth (Bengio et al., 2015; Ross et al., 2011).

9 Limitations

Tests stop at 300M and length 512; 128K block routing has only 0.09–0.10 utility. Full 31M CLHR costs ≈ 1.96× standard, using 14.1 vs. 9.2 GiB. A separate block-sparse inference microbenchmark reaches 3.2–4.9× speedup at 32K and 14.7–15.2× at 128K, but the corresponding learned block router has low utility; we therefore do not claim an end-to-end quality–speed result (Appendix J). Native-hard was tuned over learning rate only, on one seed, and incurs almost no deployment-quality penalty in our NSA-style adaptation; frozen-Qwen is null. CLHR fixes deployment mismatch, not routing quality.

10 Conclusion

Learned routers may use differentiable approximations, but shared computation must train on useful discrete deployment states. CLHR provides this coverage while retaining the learned router, turning catastrophic hardening into near-lossless deployment across the tested settings.

Reproducibility Statement

Section 3 specifies the model, data, training, evaluation, and development–confirmation protocols. Appendix E documents the exact configurations, per-seed results, data processing, software and hardware environments, and raw-to-paper provenance; Appendix L audits the independent one-pass closed-loop evaluator. The supplementary archive contains source code, training and evaluation commands, per-seed result records, mask and evaluator tests, and scripts that reproduce the paper's tables and figures from those records.

AI Use Statement

Generative AI tools assisted with proposing and refining hypotheses, conceptual framing, experimental and methodological design, implementation and debugging, formulation and checking of mathematical arguments, interpretation of experimental results, literature discovery and summarization, preparation of figures and tables, artifact documentation, and organization, drafting, and editing of the manuscript. No synthetic training or evaluation data reported in this work were generated by AI, and no human-subject qualitative analysis or manuscript translation was performed. The authors independently ran the experiments, inspected the underlying raw records, reviewed and tested AI-assisted code, and checked the mathematical arguments, citations, numerical claims, and final text. The authors take full responsibility for the content of the paper and its artifacts.

References

Keston Aquino-Michaels. Routing absorption in sparse attention: Why random gates are hard to beat. *arXiv preprint arXiv:2603.02227*, 2026.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2015.

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

Keqi Deng, Shaoshi Ling, Ruchao Fan, and Jinyu Li. UNIQUE: Universal top-k sparse attention for training-free inference and sparsity-aware training. *arXiv preprint arXiv:2605.27740*, 2026.

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. arXiv:2101.03961.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 2024a. URL `https://zenodo.org/records/12608602`.

Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden Kwok-Hay So, Ting Cao, Fan Yang, and Mao Yang. SeerAttention: Learning intrinsic sparse attention in your LLMs. *arXiv preprint arXiv:2410.13276*, 2024b.

Yizhao Gao, Shuming Guo, Shijie Cao, Yuqing Xia, Yu Cheng, Lei Wang, Lingxiao Ma, Yutao Sun, Tianzhu Ye, Li Dong, Hayden Kwok-Hay So, Yu Hua, Ting Cao, Fan Yang, and Mao Yang. SeerAttention-R: Sparse attention adaptation for long reasoning. In *International Conference on Learning Representations*, 2026.

HuggingFaceFW. FineWeb-Edu, sample-10bt, revision v1.0.0. `https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu/tree/v1.0.0/sample/10BT`, 2024. Dataset revision accessed September 2026.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. arXiv:1711.05101.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018. arXiv:1712.01312.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, et al. MoBA: Mixture of block attention for long-context LLMs. *arXiv preprint arXiv:2502.13189*, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. arXiv:1609.07843.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 1525–1534, 2016. doi: 10.18653/v1/P16-1144.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems: Datasets and Benchmarks Track*, volume 37, 2024. doi: 10.52202/079017-0970.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. Hash layers for large sparse models. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 627–635, 2011.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021. doi: 10.1162/tacl_a_00353.

Michael E. Sander, Joan Puigcerver, Josip Djolonga, Gabriel Peyré, and Mathieu Blondel. Fast, differentiable and sparse top-k: A convex analysis perspective. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 29919–29936. PMLR, 2023.

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

Zhenyi Shen, Junru Lu, Lin Gui, Jiazheng Li, Yulan He, Di Yin, and Xing Sun. Ssa: Sparse sparse attention by aligning full and sparse attention outputs in feature space. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. arXiv:2511.20102.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47901–47911. PMLR, 2024.

Phil Wang. native-sparse-attention-pytorch. `https://github.com/lucidrains/native-sparse-attention-pytorch`, 2025. Software implementation accessed September 2026.

Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. Differentiable top-k with optimal transport. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1126.

Jintao Zhang, Kai Jiang, Chendong Xiang, Weiqi Feng, Yuezhou Hu, Haocheng Xi, Jianfei Chen, and Jun Zhu. SpargeAttention2: Trainable sparse attention via hybrid top-k+top-p masking and distillation fine-tuning. *arXiv preprint arXiv:2602.13515*, 2026.

A Formal Details

A.1 Proof of gradient separation

Fix an input and suppose every layer's causally eligible kth and $(k+1)$th gate scores along the complete hard rollout are distinct. Continuity and the finite number of routing decisions give a parameter-space neighborhood over which every selected support remains unchanged. The hard masks are constant on this neighborhood, so their derivatives with respect to both parameter blocks vanish. Differentiating the resulting smooth computation with the masks held fixed is exact locally. Under a generic non-tie assumption, such as continuously distributed scores, the exceptional routing-boundary surfaces have measure zero.

A.2 Proof of top-k stability

Every selected score exceeds every unselected score by at least $\gamma_k(s)$. An ℓ_∞ perturbation smaller than $\gamma_k(s)/2$ can lower a selected score and raise an unselected score by less than $\gamma_k(s)$ in total. Their ordering cannot reverse, so the selected support is unchanged.

A.3 Proof of the state-drift bound

Apply the local Lipschitz inequality to the soft and hard transitions:

$$\delta_{\ell+1} \le \alpha_\ell \delta_\ell + \beta_\ell d(a_\ell^H, a_\ell^S). \tag{10}$$

Unrolling from the shared embedding, for which $\delta_0 = 0$, yields Equation 9.

B Full Exact-Protocol Results

Table 3: All exact-protocol conditions. Means are over three seeds. The open-loop column uses the corrected binary-mask evaluator.

Condition	Native NLL L_S	G_{OL}	G_{CL}
Learned soft gate	3.891	1.470	1.782
Contemporary soft replay	3.865	1.616	1.879
Shuffled historical	3.891	2.419	2.502
Coherent historical	3.880	1.331	1.629
Open-loop current hard replay	3.839	0.825	1.461
Shuffled closed-loop hard	3.892	0.417	0.540
CLHR, $\lambda = 0.3$	3.849	0.016	0.016
CLHR, $\lambda = 1$	**3.832**	**0.006**	**0.0035**

The causal controls above use the original $\lambda = 0.3$ objective. The 0.0035 row uses the final dedicated three-seed CLHR block. A distinct three-seed causal-ladder training block, evaluated with the same corrected one-pass metric, gives mean deployment excess 3.118 for contemporary soft replay, 1.489 for open-loop current hard replay, 0.436 for shuffled closed-loop routing, and 0.00417 for input-conditioned CLHR. These values come from the corrected one-pass evaluator in `results/ladder_lam10_eval/`; older seed-42 values produced by the mask-bug evaluator are excluded.

C PER-SEED HEADLINE RESULTS

Table 4: Closed-loop deployment excess by seed.

Scale and condition	Seed 42	Seed 123	Seed 456	Mean
31M standard	1.853	1.897	1.597	1.782
31M CLHR, $\lambda = 0.3$	0.014	0.016	0.018	0.016
31M CLHR, $\lambda = 1$	0.00375	0.00299	0.00364	0.00346
131M FineWeb-Edu standard	1.398	1.299	1.104	1.267
131M FineWeb-Edu CLHR	0.02375	0.02203	0.02344	0.02307
300M standard	1.131	0.588	0.667	0.796
300M CLHR, $\lambda = 0.3$	0.0450	0.0460	0.0443	0.0451
300M CLHR, $\lambda = 1$	0.0445	0.0519	0.0501	0.0488
300M FineWeb-Edu standard	0.4789	0.5463	–	0.5126
300M FineWeb-Edu CLHR	0.0280	0.0259	–	0.0270

The 300M FineWeb-Edu cells use two seeds; the dash records that seed 456 was not run. The corresponding U_{route} values are $2.7541/2.7125$ for standard and $3.2388/2.8328$ for CLHR (seeds 42/123). The top-1 MoE standard deployment excess values are 0.4889, 0.4662, and 0.4400 and its CLHR values are 0.0159, 0.0153, and 0.0131 for seeds 42, 123, and 456. Router utilities U_{route} are $2.6050/2.6888/2.5338$ and $3.4467/3.3984/3.3839$, respectively.

Table 5: Per-seed NSA-style adaptation evidence used in Tables 1–2. Here deployment excess is deployed minus training-native NLL and equals G_{CL} for the soft-native arms. All three arms use seeds 42, 123, and 456.

Setting	Seed	Training-native NLL	Deployed NLL	Excess NLL
NSA standard	42	3.8701	5.0903	1.2202
NSA standard	123	3.8461	5.0470	1.2009
NSA standard	456	3.8443	5.0266	1.1823
NSA standard	Mean	3.8535	5.0546	1.2011
NSA CLHR	42	3.8300	3.8521	0.0221
NSA CLHR	123	3.8307	3.8563	0.0256
NSA CLHR	456	3.8293	3.8510	0.0216
NSA CLHR	Mean	3.8300	3.8531	0.0231
NSA native-hard	42	3.8926	3.8852	−0.0074
NSA native-hard	123	3.9116	3.8895	−0.0221
NSA native-hard	456	3.8896	3.8919	0.0024
NSA native-hard	Mean	3.8979	3.8889	−0.0090

The MoBA dense-to-sparse excess values (`dense_switch` condition) are 0.06856 (seed 42) and 0.05254 (seed 456). The CLHR arm was regenerated from saved result files across three seeds after fixing a configuration bug (top-k selection equal to the full block count, which silently made the "hard" path dense by construction). The regenerated CLHR excess values are 0.06745 (seed 42; 1.62% reduction from 0.06856), 0.0602 (seed 123; no matched `dense_switch` baseline), and 0.0372 (seed 456; 29.2% reduction from 0.0525). With only two seeds having a matched baseline, we treat this as two-seed boundary evidence rather than a claim that CLHR reliably reduces MoBA's dense-to-sparse excess.

The original downstream record (seed 42 only, a custom 100-sequence WikiText-103 / 200-example LAMBADA harness) and the full-dataset, three-seed protocol now used in the main text (Table 2A) are both given in Appendix D.

D DOWNSTREAM EVALUATION PROTOCOL AND PER-SEED RESULTS

Table 2A follows the official lm-eval-harness (Gao et al., 2024a) task definitions for LAMBADA-OpenAI and WikiText. The latter task uses WikiText-2 (`wikitext-2-raw-v1`) and reports word

perplexity; it is distinct from the WikiText-103 token-level NLL used in our training-domain evaluations. Because the harness aggregation was unreliable for this model adapter, an independent evaluator computes the published task formulas directly from per-document log likelihoods. The supplement contains the evaluator and per-seed outputs. Table 2A takes arithmetic means of the three seed-level metrics (42, 123, 456).

Table 6: Per-seed downstream results underlying Table 2A.

Condition	Seed	WikiText-2 word PPL	LAMBADA PPL	LAMBADA acc.
Standard	42	$32.95 \to 116.72$	$2402 \to 16077$	$12.5\% \to 4.2\%$
Standard	123	$33.22 \to 64.67$	$2406 \to 15578$	$14.2\% \to 3.8\%$
Standard	456	$33.07 \to 70.26$	$2172 \to 13853$	$12.9\% \to 3.2\%$
CLHR	42	$32.93 \to 34.72$	$2063 \to 2403$	$14.4\% \to 12.9\%$
CLHR	123	$32.98 \to 34.88$	$1955 \to 2163$	$13.4\% \to 12.9\%$
CLHR	456	$32.99 \to 34.77$	$1929 \to 2159$	$13.9\% \to 12.9\%$

For reference, the original single-seed (42) custom harness (100 WikiText-103 sequences, 200 LAMBADA examples, a different protocol from the above) gave: standard WikiText-103 NLL $2.8551 \to 4.0455$ and LAMBADA accuracy $0.11 \to 0.03$; CLHR WikiText-103 NLL $2.8605 \to 2.9177$ and LAMBADA accuracy $0.12 \to 0.11$. These subset values should not be numerically equated with the full-dataset values above.

Table 7: Contemporary CLHR-Lite. Means and standard deviations use three seeds; timing is the synchronized L40S benchmark.

q	Native NLL L_S	Deployed NLL L_H	G_{CL}	Time	Memory
0.25	3.9963	4.0016	0.00532 ± 0.00048	$1.22\times$	9.71 GiB
0.50	3.8996	3.9043	0.00469 ± 0.00069	$1.42\times$	10.14 GiB
0.75	3.8577	3.8618	0.00407 ± 0.00043	$1.66\times$	12.11 GiB
1.00	3.8332	3.8374	0.00423 ± 0.00074	$1.97\times$	14.09 GiB

The Lite runs are a fresh reconstruction of contemporary equal-weight CLHR; their $q = 1$ endpoint closely matches, but is not numerically substituted for, the frozen headline runs. Historical-router Lite controls are reported in the artifact and are not used for the main frontier.

E Reproducibility and Result Provenance

The 31M exact-protocol runs use WikiText-103, the GPT-2 tokenizer, sequences of length 512, 50K optimization steps, $k = 64$, and an 87.5% sparsity target. The 300M models are trained for four billion processed tokens. Except for the explicitly labeled arithmetic means of downstream metrics in Table 2A, likelihood means are computed in NLL space and perplexity ratios are obtained by exponentiating NLL differences. The supplementary archive submitted with this manuscript contains per-seed JSON outputs, one-pass closed-loop evaluation code, mask-cardinality tests, training commands, and scripts mapping raw records to every table and figure. The archive contains, for every NSA and MoBA record, the saved result file and (where applicable) the evaluated checkpoint or training log used to produce it. We use the three-seed NSA-style adaptation as cross-family evidence and the two-seed matched MoBA comparison as explicitly small-n boundary evidence.

Table 8: Comparison to the closest optimization strategies. "Self-induced" means that each hard route is computed from the current state of that stream.

Method	Router-learning path	Hard states	Extra loss/stage
STE/annealing	Surrogate or relaxation	Self-induced	None
SSA-style	Selected soft/alignment path	Self-induced	Feature alignment
Post-hoc	Fit after backbone freeze	Deployment only	Separate fit
CLHR	**Differentiable soft path**	**Self-induced**	**None**

F Broader Routing Co-Adaptation Evidence

The sparse-attention study grew from a broader investigation of routing–value compatibility. We include only the three observations needed to motivate the present distinction between routing context and deployment state; the complete cross-checkpoint, temporal-null, fork, intervention, and MoE studies are reserved for separate work. None of these observations is required to define or evaluate the primary closed-loop endpoint.

Table 9: Headline evidence motivating the routing-context view. RCA denotes coherent routing-context augmentation. The interaction multiplier M_{int} is post-intervention interaction divided by pre-intervention interaction, so lower is better.

Observation	Replication	Headline result
Dense routing–value compatibility	5 seeds	Matched advantage 2.7 ± 0.3 points; oldest-first RCA reduces interaction by 88.6%
Independent optimization forks	5 parents	RCA reduces branch-specific compatibility by 30–44%
Routing-granularity reversal	3 seeds/cell	Categorical M_{int}: $0.877/0.537$; fine-grained M_{int}: $0.697/0.917$ (coherent/shuffled)

These observations support a restrained lineage: shared computation can become specialized to its routing trajectory; lower measured coupling can instead arise because a categorical router becomes less relevant; and fine-grained routing must preserve coherent input-conditioned structure. They do not imply generic component interchangeability, metric-invariant behavior, or broad benefits for merging, continual learning, or pruning. Sparse attention adds the further requirement that every route be generated from the hard-induced state created by earlier routes, which is the aspect targeted by CLHR.

G Baseline Tuning and Additional Controls

G.1 Native-hard learning-rate control

Table 10: Native-hard learning-rate sweep at 31M (seed 42), plus three default-rate seeds. LR is relative to the 3×10^{-4} default used elsewhere in this paper.

A. Learning-rate sweep (seed 42)

	$0.5\times$ LR	$1\times$ LR	$2\times$ LR	$4\times$ LR	$8\times$ LR	$16\times$ LR	$32\times$ LR
LR	1.5e−4	3e−4	6e−4	1.2e−3	2.4e−3	4.8e−3	9.6e−3
Native PPL	124.9	99.2	79.8	**70.7**	89.2	93.5	111.2

B. Default-rate seeds

	Seed 42	Seed 123	Seed 456
Native PPL	99.19	96.03	98.15

G.2 SSA-style learned-router adaptation

This experiment adapts SSA's training recipe to the exact 31M learned-router setting; it is not a reproduction of SSA's original block-sparse system. The soft and hard attention outputs are computed from each visited state, one stream is selected once per batch for propagation, and a bidirectional smooth-ℓ_1 alignment loss is applied between the outputs. Its weight is selected on seed 42 and fixed at one for seeds 123 and 456.

Table 11: Held-out comparison to SSA-style training. "Native" is L_S and "deployed" is L_H; parentheses contain G_{CL}.

Method	Seed 42	Seed 123	Seed 456
CLHR $\lambda = 1$, native	3.8318	3.8339	3.8312
CLHR $\lambda = 1$, deployed	3.8355 (0.00375)	3.8369 (0.00299)	3.8348 (0.00364)
SSA-style, native	3.8698	3.8827	3.8717
SSA-style, deployed	3.8727 (0.00295)	3.8862 (0.00353)	3.8752 (0.00344)

Removing feature alignment changes deployed NLL by $+0.00273$, -0.00306, and $+0.00303$ nats for seeds 42, 123, and 456 when reported as no-alignment minus alignment. The mean is $+0.00090 \pm 0.00343$ nats. Corresponding U_{route} remains 0.820 ± 0.006 without alignment and 0.827 ± 0.032 with alignment.

Table 12: Seed-42 frequency and alignment factorial. Deployed NLL L_H is the primary quantity; each cell also shows G_{CL} in parentheses.

Hard-stream probability	No alignment	Alignment weight 1
$p_H = 0.231$	3.9071 (0.0172)	3.9217 (0.0116)
$p_H = 0.500$	3.8755 (0.0037)	3.8727 (0.0029)

In a separate frequency intervention, the soft loss is evaluated every batch and a sampled hard loss is scaled by λ/p_H, holding its expected coefficient fixed. Table 13 reports the replicated endpoints. Full update cadence ($p_H = 1$) improves deployed NLL for every seed, with paired mean improvement 0.0460 ± 0.0069 nats. The development-seed midpoint has deployed NLL 3.9029 at $p_H = 0.5$, between its endpoint values 3.9240 and 3.8754. Thus the endpoints support the replicated coverage effect, while the complete three-point dose response remains development-seed evidence.

Table 13: Constant-expected-objective frequency endpoints. Δ is deployed NLL at $p_H = 0.25$ minus deployed NLL at $p_H = 1$.

Seed	$p_H = 0.25$	$p_H = 1$	Δ
42	3.9240	3.8754	0.0486
123	3.9281	3.8768	0.0512
456	3.9066	3.8683	0.0383
Mean $\pm$ sample SD	3.9195 ± 0.0114	3.8735 ± 0.0046	0.0460 ± 0.0069

These are optimization-dynamics results, not a claim that the expected gradients differ. At $p_H = 1$, this procedure is exactly normalized $\lambda = 0.3$ CLHR. Its difference from the original 3.8634 deployed NLL is consistent with sensitivity to the original run's unnormalized loss scale, subject to ordinary run-to-run training variation. The held-out endpoint runs use L40S GPUs; the seed-42 $p_H = 1$ run used an A10G. Hardware timing is therefore not compared across the three seeds.

More precisely, for a fixed minibatch let $v_S = \nabla \mathcal{L}_S$ and $v_H = \nabla \mathcal{L}_H$, and let $\widehat{v}_p = v_S + Z(\lambda/p_H)v_H$, where $Z \sim \mathrm{Bernoulli}(p_H)$. Then

$$\mathbb{E}_Z \widehat{v}_p = v_S + \lambda v_H, \qquad \mathbb{E}_Z \left[\|\widehat{v}_p - \mathbb{E}_Z \widehat{v}_p\|_2^2 \right] = \lambda^2 \frac{1 - p_H}{p_H} \|v_H\|_2^2.$$

Equal expected gradients therefore do not imply equal stream-selection variance or finite-time optimizer trajectories.

Table 14: Development-seed normalized weight sweep. The normalized $\lambda = 0.3$ result is the $p_H = 1$ frequency run above. The original $\lambda = 0.3$ reference used the earlier unnormalized objective.

Setting	Native NLL L_S	Deployed NLL L_H	G_{CL}
Original $\lambda = 0.3$	3.8491	3.8634	0.01431
Normalized $\lambda = 0.3$	3.8428	3.8754	0.03253
Normalized $\lambda = 0.5$	3.8357	3.8600	0.02430
Normalized $\lambda = 1$	**3.8321**	**3.8377**	0.00559
Normalized $\lambda = 2$	3.8375	3.8420	**0.00449**

G.3 EARLIER SOFT-TO-HARD ESTIMATORS

Straight-through and temperature-annealing hyperparameters were selected on seed 42 and frozen before evaluation on seeds 123 and 456. The full held-out results are shown below. Both methods leave substantially more deployment excess than CLHR, and both degrade native language-modeling quality.

Table 15: Per-seed results for tuned soft-to-hard alternatives.

Method	Seed	Native NLL	G_{CL}
STE	42	4.223	0.694
STE	123	4.220	0.995
STE	456	4.213	0.669
Annealing	42	5.067	0.394
Annealing	123	5.001	0.710
Annealing	456	5.083	0.412
Dual STE–CLHR	42	4.300	0.123
Dual STE–CLHR	123	4.336	0.154
Dual STE–CLHR	456	4.278	0.082

Router-utility controls replace the learned hard route at evaluation with a balanced random route or a local-window route at the same sparsity. For CLHR, the learned route improves NLL by 0.894 nats over the balanced random route, compared with 0.472 nats for standard training. Shuffled closed-loop training instead yields negative U_{route}. Thus the near-zero deployment excess is not explained by making all admissible masks equivalent. These U_{route} figures use the original $\lambda = 0.3$ experiment; the SSA-style value above directly verifies that the closest baseline also retains useful routing.

H CROSS-CORPUS AND SCALE DETAILS

The FineWeb-Edu experiment uses the 131M configuration: 12 layers, width 768, 12 heads, feed-forward width 3072, and 131,087,616 parameters including the GPT-2 embedding and language-model head. We construct 25 shards of 100M GPT-2 tokens from `HuggingFaceFW/fineweb-edu`, `sample-10BT`, revision `v1.0.0`. Training consumes 1,999,962,112 tokens from this 2.5B-token pool over 30,517 updates. Evaluation uses a separate 100M-token shard beyond the training set. Text is concatenated without inserted document separators.

Table 16: FineWeb-Edu cross-corpus results. The selected contemporary $\lambda = 1$, $k = 64$ configuration is not retuned.

Method	Seed	Native NLL L_S	Deployed NLL L_H	G_{CL}
Standard	42	3.6669	5.0647	1.3978
Standard	123	3.6656	4.9647	1.2991
Standard	456	3.6555	4.7594	1.1039
CLHR	42	3.6463	3.6700	0.02375
CLHR	123	3.6406	3.6627	0.02203
CLHR	456	3.6305	3.6539	0.02344

I 300M TRAINING CONFIGURATION

The 300M experiments use the `Salesforce/wikitext wikitext-103-raw-v1` release. Preprocessing removes empty text records, encodes each remaining record with the GPT-2 tokenizer (50,257-token vocabulary), and concatenates the resulting token IDs without inserting an additional document separator. Contiguous nonoverlapping 513-token chunks produce 512-token inputs and next-token targets. No additional filtering or deduplication is applied. The training loader reshuffles chunks whenever the finite WikiText-103 stream is exhausted, so the reported four billion tokens include repeated passes. Validation uses the corresponding official validation split and token-weighted NLL over 100 batches.

The architecture has 20 pre-normalized transformer blocks, model width 1024, 16 heads, head width 64, feed-forward width 4096, learned positional embeddings, gate width 32, and dropout 0.1. Token embeddings and the language-model head are tied. AdamW uses peak learning rate 3×10^{-4}, betas $(0.9, 0.95)$, weight decay 0.1, 2,000 linear-warmup steps, cosine decay to zero, and global gradient clipping at 1.0. The standard and final normalized $\lambda = 1$ runs process 65,536 tokens per optimizer step and train for 61,035 steps, totaling 3,999,989,760 processed tokens. The earlier $\lambda = 0.3$ closed-loop runs use a memory-constrained 57,344-token batch and 69,754 steps.

Training uses bfloat16 autocast, activation checkpointing, `torch.compile`, and NVIDIA Collective Communications Library (NCCL) distributed data parallelism. Runs were executed on NVIDIA L40S, RTX 5090, and RTX PRO 6000 Blackwell systems. The timing comparison in Table 17 uses four GPUs for standard training and eight for closed-loop training; it is therefore reported in GPU-hours rather than as a direct wall-clock speed ratio. Early checkpoints at 250M and 500M were retained for seeds 123 and 456. Checkpoints at 1B, 2B, and the final 4B state were retained for all three seeds. A legacy file named `tokens_3B.pt` records 3.500B processed tokens at step 61,036 for each original $\lambda = 0.3$ CLHR seed, as verified from its serialized metadata. The current implementation materializes dense score tensors and applies a binary mask; it does not use a training-time block-sparse kernel.

J Compute Accounting

Training currently uses dense-emulated masked attention rather than a sparse kernel, so model-based active FLOPs do not predict measured runtime directly.

Table 17: Canonical synchronized 31M training benchmark on one NVIDIA L40S. Times are relative to standard training.

Method	Steps/s	Relative time	Peak memory
Standard	11.43	1.00×	9.19 GiB
CLHR-Lite, $q = 0.25$	9.36	1.22×	9.71 GiB
SSA-style	8.55	1.34×	10.27 GiB
CLHR-Lite, $q = 0.50$	8.07	1.42×	10.14 GiB
CLHR-Lite, $q = 0.75$	6.89	1.66×	12.11 GiB
Full CLHR, $q = 1$	5.80	1.97×	14.09 GiB

The benchmark uses three repetitions, identical fixed batches, 100 warm-up steps, 500 measured updates, rotated method order, and explicit CUDA synchronization. It is the canonical main-text cost comparison.

As an independent device replication, we time matched seed-456 31M runs on one RTX PRO 6000 Blackwell. Standard training takes 2,810 seconds (0.781 GPU-hours), 17.79 steps/s, and 145.7k nominal input tokens/s. SSA-style alternation takes 3,448 seconds (0.958 GPU-hours), 14.50 steps/s, and 118.8k nominal input tokens/s, or $1.23\times$ standard time. Full-coverage CLHR takes 5,458 seconds (1.516 GPU-hours), 9.16 steps/s, and 75.0k nominal input tokens/s, or $1.94\times$ standard and $1.58\times$ SSA-style time. The CLHR timing uses the original $\lambda = 0.3$ objective, whose two-trajectory workload is identical to normalized $\lambda = 1$; changing the weight does not change the executed branches. The independently retrained standard checkpoint follows a different numerical optimization trajectory from the primary L40S seed-456 checkpoint, so we use it only for timing. These measurements support a quality–compute frontier, not a claim that CLHR dominates SSA-style on cost.

We verify these full-run ratios with a synchronized repeated benchmark. Each method uses the same fixed batches in three repetitions, with 100 untimed warm-up updates followed by 1,000 measured updates. Method order is rotated across repetitions, and CUDA is synchronized immediately before and after each measurement. Standard takes 55.91 ± 0.43 seconds, SSA-style takes 68.90 ± 0.10 seconds, and CLHR takes 109.67 ± 0.12 seconds (sample standard deviations). Thus SSA-style is $1.232\times$ standard, while CLHR is $1.961\times$ standard and $1.592\times$ SSA-style. Instrumented peak CUDA allocations are 9.26, 10.27, and 14.09 GiB, respectively. The repeated results agree with the complete-training measurements and remove evaluation, checkpointing, warm-up, and asynchronous-kernel timing from the comparison.

Realized latency gains require a compatible sparse kernel and are not projected here. The long-context numbers quoted in the limitations come from the field `end_to_end_speedup_block_router_direct` in: `bench_flex_real_selection_k8_l40s_end_to_end_bs128.json` (32K L40S: 3.169/3.187), `bench_flex_real_selection_k8_end_to_end_bs128.json` (32K RTX 5090: 4.731/4.854), `bench_flex_real_selection_k8_64k_end_to_end_bs128.json` (64K: 9.062/9.231), and `real_selection_128k_k8/bench_flex_real_selection_k8_128k_end_to_end_bs128.json` (128K: 14.746/15.180). Each pair is the two measured configurations retained in the corresponding immutable JSON record.

K Additional State-Divergence Diagnostics

For terminal states after the final LayerNorm, define $\cos_L^{\mathrm{LN}} = \cos(h_L^{H,\mathrm{LN}}, h_L^{S,\mathrm{LN}})$. As in the main text, cosine similarity is computed per token across the hidden dimension, then averaged over tokens and evaluation batches. A local quadratic loss expansion motivates the empirical relation

$$\widehat{G_{\mathrm{CL}}} = 3.17(1 - \cos_L^{\mathrm{LN}}). \tag{11}$$

It is a compact mechanistic summary, not a law beyond the tested settings.

The one-parameter coefficient is fit in natural-log space on 24 adapted-protocol condition–seed observations. Resampling the five seed identities as blocks, keeping all observations belonging to each sampled seed, gives a 95% bootstrap interval $[2.73, 3.44]$ from 20,000 resamples. The exact-protocol predictor comparison is:

Table 18: Evaluation of the terminal-divergence relation. The coefficient is fit only on the adapted 31M block. $R^2_{\log}$ evaluates log-transformed predicted and observed positive excess; R^2 and mean absolute error (MAE, in nats) are on the original scale.

Block	n	$R^2_{\log}$	R^2	MAE
31M adapted (fit)	24	0.962	0.838	0.216
31M exact (transfer)	15	0.966	0.844	0.231
300M (transfer)	6	0.857	0.957	0.066

Table 19: Predicting final G_{CL} after 10% of training on the eight-run diagnostic set. R^2 is from a univariate linear fit; thresholds are not used.

Early predictor	Spearman ρ	R^2	Role
Early G_{CL}	**0.976**	**0.996**	Practical screen
Maximum layerwise drift	0.952	0.980	Label-free rank signal
Final-layer drift	0.905	0.983	Mechanistic endpoint
Mean routing margin	-0.881	0.959	Router confidence
Mean support Jaccard	-0.643	0.825	Weaker support statistic
Retained mass	-0.690	0.276	Poor predictor

The empirical final-layer drift bands $[0, 0.025)$, $[0.025, 0.10)$, and $[0.10, \infty)$ are labeled low observed risk, uncertain, and high observed risk. They are descriptive thresholds selected after inspecting outcomes. The broader ledger contains zero false-safe declarations and two conservative false warnings; we do not claim prospective calibration.

We measure terminal states after the final LayerNorm so that the diagnostic matches the representation consumed by the language-model head. As a robustness check on the adapted-protocol fit block, the corresponding pre-normalization relation has coefficient 3.20, $R^2_{\log} = 0.963$, and original-scale $R^2 = 0.843$. The pre- and post-normalization angular distances have correlation $r = 0.999981$, with a mean post/pre distance ratio of 1.010. Final normalization therefore has little effect on the fitted relationship.

Table 20: Simple predictors of exact-protocol deployment excess. Each value is original-scale R^2 on the held-out exact block.

Predictor	R^2
Native NLL	-1.397
Open-loop excess G_{OL}	0.929
Support disagreement ($1-$support Jaccard)	0.573
Condition identity	0.827
Final post-LayerNorm angular divergence	**0.844**

To test whether predictiveness appears only immediately before the language model head, we repeat the same log-space coefficient fit separately at every layer and transfer it to the exact-protocol block. Original-scale R^2 values for layers 1–6 are -0.430, -0.917, -0.523, 0.023, 0.760, and 0.844. Thus penultimate-layer divergence is already strongly predictive, although the final layer is best; early-layer distance alone is insufficient.

Table 21: All 300M diagnostic predictions. These evaluations use the diagnostic checkpoint/evaluator records and therefore differ slightly from rounded headline values.

Run	Predicted G_{CL}	Observed G_{CL}
Standard, seed 42	0.936	1.114
Standard, seed 123	0.575	0.587
Standard, seed 456	0.608	0.662
CLHR $\lambda = 0.3$, seed 42	0.095	0.045
CLHR $\lambda = 0.3$, seed 123	0.097	0.046
CLHR $\lambda = 0.3$, seed 456	0.095	0.044

L Closed-Loop Evaluator Audit

Closed-loop evaluation is implemented as a single forward rollout. At layer ℓ, the evaluator computes scores from the current hard-path state, selects exactly k entries per eligible query, applies that mask immediately, and passes the resulting state to layer $\ell + 1$. It never caches masks from a soft or teacher trajectory. A separately implemented evaluator reproduces the same per-token losses on fixed checkpoints.

Unit tests check binary mask values, exact cardinality after causal-prefix handling, absence of future-token attention, and equality between accumulated token-weighted NLL and a direct full-corpus calculation. Fresh-process tests guard against cached routing state, and explicit open-loop tests confirm that precomputed soft-state masks reproduce G_{OL} rather than G_{CL}. The artifact maps each table row to its evaluator mode, checkpoint, result record, and aggregation command.

M Negative and Boundary Results

Deployment under unseen sparsity budgets. The original $k = 64$-trained $\lambda = 0.3$ CLHR checkpoints remain deployable at $k = 32$: deployment excess values are 0.0462, 0.0512, and 0.0644 across seeds 42, 123, and 456 (mean 0.054 ± 0.008, population SD), while learned routes improve deployed NLL over balanced random routes by 1.14–1.19 nats. On seed 42, unseen denser budgets $k = 128$ and $k = 256$ instead give excess values 0.119 and 0.352. The result supports transfer to one more aggressive cutoff, not arbitrary multi-budget flexibility.

Coarse block routing. At length 2048, an average-pooled 64-token block router trained with historical closed-loop supervision has negative U_{route} in every seed (-0.146, -0.113, and -0.037). CLHR reduces the standard model's very large deployment excess but leaves a mean excess of 1.65 nats. Because context length, granularity, pooling, and architecture change together, this is an architectural boundary rather than evidence that long context itself causes greater absorption.

The positive claim is restricted to from-scratch settings with a large baseline hard-deployment excess. In a frozen pretrained Qwen adaptation, the baseline excess was already small and CLHR

supplied no reliable benefit. The available headroom and representation plasticity may both affect whether closed-loop training is useful.

Broader component-substitution experiments in merging, continual learning, pruning, and generic module replacement also did not establish a consistent external utility benefit. Lower routing–value specialization did not make independently trained modules interchangeable in these experiments. The claim for CLHR is limited to deployment-state mismatch in end-to-end learned sparse attention.