LEARNING TOKEN-POPULATION GEOMETRY FOR SPARSE MIXTURE-OF-EXPERTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Routing in sparse mixture-of-experts (MoE) models determines both token-level computation and the populations from which experts learn. Load balancing controls how much traffic each expert receives, but does not specify how these populations are organized in representation space. We introduce the Geometry-Guided Learnable Router (GGLR), which uses expert input means maintained by an exponential moving average as an explicit population reference for a task-trained router. This reference defines population-relative coordinates for expert selection and target directions whose changes guide router updates, while selection-only biases regulate utilization. We analyze how these population-relative coordinates transform pairwise routing logits and how historical averaging affects the accuracy of the population reference. In matched 1.7-billion-parameter experiments on Nemotron-CC-v2, GGLR achieves the lowest observed endpoint loss and global load imbalance, while producing more separated directions of expert input means and more distinct learned router directions. Compared with Loss-Free Balancing with normalized output gains, GGLR reduces global maximum overload by 23.1%, while also lowering validation perplexity.

1 INTRODUCTION

Sparse mixture-of-experts (MoE) models scale model capacity while activating only a small subset of experts for each token, allowing parameter capacity to grow without a proportional increase in per token computation (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022). Repeated router assignments determine which experts process each token and define the input populations used to train each expert. Prior work on routing stability, expert specialization, and representation geometry has highlighted different aspects of this interaction between routing assignments and expert learning (Dai et al., 2022; 2024; Nielsen et al., 2025).

Load balancing addresses an important aspect of this process by regulating how frequently experts are selected and limiting persistent utilization skew (Fedus et al., 2022; Wang et al., 2024). Balanced assignment counts, however, do not determine the structure of the representations received by each expert. Experts with similar loads may nevertheless be trained on populations that differ in location, direction, overlap, and internal variation in hidden space.

Figure 1 previews this design: historical expert-input geometry guides a task-trained router, while selection biases regulate utilization separately.

Standard learned routers optimize expert preferences directly through the language modeling objective, while any structure in their past assignments is represented only implicitly in the learned parameters. Geometric approaches can make such structure explicit. Online K-means, for example, maintains running means of expert inputs and directly selects experts using cosine similarity to these means (Ahrac et al., 2026). This makes population geometry explicit, but uses centroid similarity itself as the routing criterion. We instead seek to incorporate historical population geometry while retaining expert preferences learned from the task objective.

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.

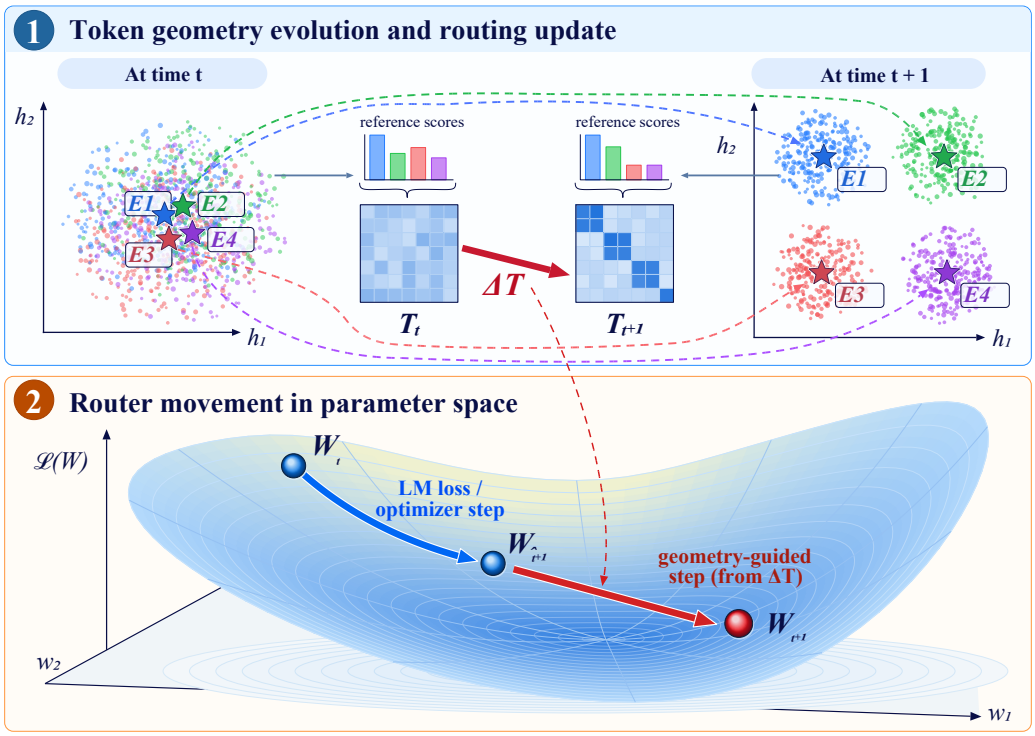


Figure 1: Conceptual overview of GGLR. The upper panel shows mixed expert-input populations being summarized by reference scores and target-direction matrices. The lower panel separates the task-optimizer displacement from the geometry-guided displacement.

We introduce the **Geometry-Guided Learnable Router (GGLR)**, which maintains exponential moving average (EMA) population means as a population reference for a learnable router. Rather than using these means directly as selector centroids, GGLR uses their relative geometry to define the coordinates in which expert preferences are evaluated and to guide how the router evolves over training. The router parameters remain optimized through the language modeling objective, while selection biases separately regulate expert utilization. In this way, GGLR makes historical population structure explicit while retaining task-driven expert preferences.

In matched experiments with a 1.7-billion-parameter MoE on Nemotron-CC-v2, GGLR achieves the lowest observed validation loss and global load imbalance among five routing methods evaluated at the same training endpoint. Compared with Loss-Free Balancing with normalized output gains, GGLR reduces global maximum overload by 23.1% while improving validation perplexity, and the perplexity improvement extends to WikiText-2. These results are accompanied by greater separation between directions of expert input means and lower overlap among learned router directions, consistent with the geometric organization encouraged by GGLR.

2 Related Work

Sparse routing, load control, and representation content. Sparse routing activates a few experts per token; repeated assignments determine both expert traffic and the input populations from which experts learn. Frequency- and probability-based balancing constrain that traffic (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022). Router z-loss stabilizes logits (Zoph et al., 2022), while balanced-assignment rules allocate tokens without that loss (Lewis et al., 2021; Roller et al., 2021). Scaling sparse computation also involves active-compute costs (Du et al., 2022), expert execution (He et al., 2021; Rajbhandari et al., 2022; Hwang et al., 2023; Gale et al., 2022), and initialization or architectural choices across language and vision (Komatsuzaki et al., 2023; DeepSeek-AI et al.,

2024; Riquelme et al., 2021). Recent load-control rules turn expert loads into selection-bias updates (Wang et al., 2024; DeepSeek-AI et al., 2025) or averaged routing probabilities into auxiliary coefficients (Chen et al., 2026); DeepSeek-V3 also pairs its biases with unbiased gains and a sequence loss. Load statistics describe traffic; GGLR also uses historical means of assigned inputs to define population-relative selection coordinates for its task-trained router, while separate biases regulate utilization.

Expert specialization and geometric references. GGLR's geometric reference differs from routes that stabilize assignments or partition capacity. StableMoE freezes distilled assignments, DeepSeekMoE separates fine-grained and shared capacity, and MoEfication partitions feed-forward functions (Dai et al., 2022; 2024; Zhang et al., 2022). Geometric routers shape the selection space through feature rescaling, row orthogonality, or prototype alignment (Nielsen et al., 2025; Omi et al., 2025; Yang, 2025); Online K-means chooses from running centroids plus load biases (Ahrac et al., 2026). GGLR instead uses EMA population means to center and scale a task-trained router, so learned rows set preferences while biases regulate load.

Contextual anisotropy, structural probes, low-dimensional maps, and CKA compare representation geometry, while alignment/uniformity, isotropy corrections, and fine-grained analyses expose structure hidden by aggregate similarity (Ethayarajh, 2019; Hewitt & Manning, 2019; Hernandez & Andreas, 2021; Kornblith et al., 2019; Wang & Isola, 2020; Rajaee & Pilehvar, 2021; Bihani & Rayz, 2021; Liu et al., 2021; Fuster Baggetto & Fresno, 2022; Yang et al., 2023). Our evaluation recomputes expert-input means from each model's assignments and compares the learned router rows.

3 Geometry-Guided Learnable Router

GGLR uses EMA population means to guide a router trained with the language modeling objective.

3.1 Expert Populations Induced by Routing

Consider an MoE layer with E experts $F_e : \mathbb{R}^d \to \mathbb{R}^d$ and input representations $\{\mathbf{h}_i\}_{i=1}^N$, where d is the hidden width and N is the number of input tokens. For each input, the router selects K experts and combines their outputs as

$$\mathcal{A}_i = \mathrm{TopK}_e\{q_e(\mathbf{h}_i) + b_e\}, \qquad \mathbf{y}_i = \sum_{e \in \mathcal{A}_i} g_{i,e} F_e(\mathbf{h}_i), \tag{1}$$

where $q_e(\mathbf{h}_i)$ is the selection affinity, b_e is the selection bias, and $g_{i,e}$ is the output gain. Biased selection scores determine expert participation, whereas gains determine the contributions of the selected experts.

These assignments induce an input population for each expert,

$$\mathcal{B}_e = \{\{\, \mathbf{h}_i : e \in \mathcal{A}_i \,\}\}, \tag{2}$$

where the multiset contains each representation once for every assignment to expert e. A token contributes to every selected expert, so the populations need not form a disjoint partition. Let $\ell_e = |\mathcal{B}_e|$ denote the assignment count and $f_e = \ell_e/N$ the selection frequency per token. Since $\sum_e f_e = K$, the uniform utilization target is K/E.

3.2 Population-Relative Selection

To incorporate population information without prescribing assignments by centroid similarity, GGLR uses EMA population means to define the inputs to a learned selector. Let $\mathbf{c}_e \in \mathbb{R}^d$ denote expert e's EMA population mean. Rather than using these means directly as selector centroids, GGLR uses their collective geometry to define a common coordinate system for learned routing. Specifically, the population center, population radius, and transformed selection input are

$$\mathbf{a} = \frac{1}{E}\sum_{e=1}^{E} \mathbf{c}_e, \qquad R = \sqrt{\frac{1}{E}\sum_{e=1}^{E} \|\mathbf{c}_e - \mathbf{a}\|_2^2}, \qquad \mathbf{x}(\mathbf{h}) = \frac{\mathbf{h} - \mathbf{a}}{\max\{R, \epsilon_r\}}, \tag{3}$$

where $\epsilon_r = 10^{-8}$. The population center $\mathbf{a}$ weights EMA population means equally rather than by assignment count, and the population radius R measures their root mean square distance from that center. The transformation expresses an input relative to the historical populations and at the scale of their spread. Using a shared population radius preserves relative residual lengths, unlike normalizing each residual separately.

GGLR uses these coordinates for selection and the original input direction for output weighting. Both affinities share a learned matrix $\mathbf{W} \in \mathbb{R}^{E\times d}$, with row $\mathbf{w}_e^\mathsf{T}$ associated with expert e,

$$\widehat{\mathbf{h}} = \frac{\mathbf{h}}{\max\{\|\mathbf{h}\|_2, \epsilon_r\}}, \qquad q_e(\mathbf{h}) = \sigma\big(\mathbf{w}_e^\mathsf{T}\mathbf{x}(\mathbf{h})\big), \qquad \gamma_e(\mathbf{h}) = \sigma\Big(\mathbf{w}_e^\mathsf{T}\widehat{\mathbf{h}}\Big), \tag{4}$$

where $\sigma(u) = (1+\exp(-u))^{-1}$. The selection affinity q_e evaluates learned preferences relative to the population reference, whereas the gain affinity γ_e uses the uncentered input direction. Router rows are not normalized, leaving both their directions and norms learnable.

The biased selection scores $q_e(\mathbf{h})+b_e$ determine the selected set through Equation 1. Write $\mathcal{A}(\mathbf{h}_i) = \mathcal{A}_i$ and $g_e(\mathbf{h}_i) = g_{i,e}$. Within that set, we normalize the gain affinities,

$$g_e(\mathbf{h}) = \frac{\gamma_e(\mathbf{h})}{\sum_{j\in\mathcal{A}(\mathbf{h})}\gamma_j(\mathbf{h})}, \qquad e \in \mathcal{A}(\mathbf{h}), \tag{5}$$

so the selected gains sum to one. Experts receive the original representation $\mathbf{h}$, not the transformed input $\mathbf{x}(\mathbf{h})$. The population-relative coordinates and selection biases do not enter the gain affinities explicitly, although they affect the selected set and hence gain normalization. During backpropagation, gradients are computed through the selected gains and the shared router matrix, treating the discrete selected set as fixed.

Following Wang et al. (2024), we regulate utilization through a separate bias update after each optimizer step,

$$b_e \leftarrow b_e + \alpha\ \mathrm{sign}\left(\frac{K}{E} - \bar{f}_e\right), \tag{6}$$

where $\bar{f}_e$ is the mean global selection frequency over that step's microbatches and $\alpha = 0.001$ in our experiments. This update increases the bias of underutilized experts and decreases that of overutilized experts. Biases enter selection after the sigmoid and are excluded from the gain affinities.

3.3 Estimating the Population Reference

To track evolving expert input populations while retaining assignment history, GGLR maintains an EMA of the population mean observed for each expert. Raw input sums and assignment counts accumulate across all gradient-accumulation microbatches and are all-reduced once across processes after the final backward pass, before one EMA update per optimizer interval; per-forward selected-frequency aggregation remains a separate statistic for the bias update. For inputs assigned to expert e in its rth nonempty step, let $\mathcal{B}_{e,r}$ be the input multiset with $n_{e,r} = |\mathcal{B}_{e,r}|$. The corresponding observed population mean and EMA population mean are

$$\mathbf{m}_{e,r} = \frac{1}{n_{e,r}}\sum_{\mathbf{h}\in\mathcal{B}_{e,r}}\mathbf{h}, \qquad \mathbf{c}_{e,r} = \beta\mathbf{c}_{e,r-1} + (1-\beta)\mathbf{m}_{e,r}, \tag{7}$$

with $\beta = 0.99$ in our experiments.

Each nonempty optimizer step contributes one observed population mean to the EMA with coefficient $1-\beta$, regardless of the number of assigned tokens. The EMA therefore weights population observations by recency rather than assigning equal weight to all historical tokens.

3.4 Geometry-Guided Router Learning

GGLR also uses population evolution to guide parameter updates without requiring the learned router rows to match the target directions. After t completed optimizer steps, let $\mathbf{c}_e(t)$ denote the EMA population mean. Assuming the centered EMA population means are nonzero, define the

Algorithm 1 One GGLR optimizer interval

Require: Router $\mathbf{W}_t$, EMA means $\{\mathbf{c}_e\}$, bias $\mathbf{b}$, microbatches $\{\mathcal{M}_s\}_{s=1}^{M}$, and coefficients β, α, λ

1: Save interval-start target $\mathbf{T}_t$ using Equation 8
2: **for** $s = 1, \ldots, M$, keeping $\mathbf{W}_t, \{\mathbf{c}_e\}, \mathbf{b}$ fixed **do**
3: Route $\mathcal{M}_s$ and combine outputs using Equations 1–5
4: Accumulate detached raw hidden sums $\mathbf{S}_e$ and assignment counts n_e
5: Record globally aggregated per-token selection frequencies $f_e^{(s)}$
6: Backpropagate the microbatch task loss divided by M
7: **end for**
8: All-reduce $\mathbf{S}_e$ and n_e across processes
9: Apply Equation 7 once for each $n_e > 0$; leave empty experts unchanged
10: Save $\mathbf{W}_t$; apply the task optimizer to obtain $\mathbf{W}_{t+1}^{\text{opt}}$
11: Form $\mathbf{T}_{t+1}$ from the updated EMA means using Equation 8
12: Compute $\bar{f}_e = M^{-1} \sum_{s=1}^{M} f_e^{(s)}$ and update $\mathbf{b}$ for the next interval using Equation 6
13: Apply the Direct displacement using Equation 9

target-direction matrix by

$$\mathbf{T}_t[e,:] = \left(\frac{\mathbf{c}_e(t) - \mathbf{a}_t}{\|\mathbf{c}_e(t) - \mathbf{a}_t\|_2} \right)^{\mathsf{T}}, \qquad \mathbf{a}_t = \frac{1}{E} \sum_{j=1}^{E} \mathbf{c}_j(t). \tag{8}$$

Each row of $\mathbf{T}_t$ is the centered unit direction of an EMA population mean, using the equally weighted population center from Equation 3.

Let $\mathbf{W}_t$ denote the router after t completed steps and $\mathbf{W}_{t+1}^{\text{opt}}$ its value after the next task optimizer update. The assignments collected during that step produce the updated target-direction matrix $\mathbf{T}_{t+1}$. GGLR combines the optimizer displacement with the change in target directions,

$$\mathbf{W}_{t+1} = \mathbf{W}_t + (1 - \lambda)\big(\mathbf{W}_{t+1}^{\text{opt}} - \mathbf{W}_t\big) + \lambda\big(\mathbf{T}_{t+1} - \mathbf{T}_t\big), \tag{9}$$

with $\lambda = 0.5$ in our experiments.

The geometric contribution depends on changes in the target directions, rather than their current mismatch with the learned router rows. When the target directions are unchanged, it vanishes and only the scaled task optimizer displacement remains. Equation 9 therefore incorporates population evolution without imposing equality between the router rows and the target directions.

Within each optimizer step, population statistics and selection biases remain fixed during the forward and backward passes. The accumulated assignments then update the EMA population means. The geometric write is applied after the task optimizer and bias correction, without gradients or modifications to optimizer moments.

Initialization uses a pass without gradients over the first real training microbatch to obtain initial EMA population means and target directions. The random number generator (RNG) states are then restored and the microbatch is replayed for training. Algorithm 1 summarizes the optimizer interval; Appendix C.1 records initialization and implementation details.

4 Analysis of Population-Relative Routing

4.1 Exposing Population Differences

We first ask how population-relative coordinates change pairwise routing logit differences compared with unit-normalized inputs. Holding the router rows and stored EMA population means fixed isolates the effect of the coordinate transformation.

Proposition 1 (Common-component removal and logit differences) *Let* $\mathbf{d}_{ej} = \mathbf{w}_e - \mathbf{w}_j$, *with* $R > \epsilon_r$ *and* $\|\mathbf{h}\|_2 > \epsilon_r$. *The unit-input and population-relative logit differences,* $\Delta_{ej}^{\text{unit}} =$

$\mathbf{d}_{ej}^{\mathsf{T}}\mathbf{h}/\|\mathbf{h}\|_2$ *and* $\Delta_{ej}^{\mathrm{rel}} = \mathbf{d}_{ej}^{\mathsf{T}}(\mathbf{h}-\mathbf{a})/R$*, satisfy*

$$\Delta_{ej}^{\mathrm{rel}} = \frac{\|\mathbf{h}\|_2}{R}\Delta_{ej}^{\mathrm{unit}} - \frac{\mathbf{d}_{ej}^{\mathsf{T}}\mathbf{a}}{R}. \tag{10}$$

If $\mathbf{d}_{ej}^{\mathsf{T}}\mathbf{a} = 0$*, population-relative coordinates preserve the sign and scale the gap magnitude by* $\|\mathbf{h}\|_2/R$*.*

For general learned rows, subtracting the center can also change that ordering. These statements concern pre-sigmoid logits; expert selection additionally depends on sigmoid affinities and selection biases. Appendix B.1 gives the proof and a shared-component example.

4.2 Estimating the Population Reference

Both selection coordinates and router-update directions depend on EMA population means. With a layer's input distribution and routing rule fixed, we compare the steady-state EMA population mean with a single optimizer-step population mean. Under the stationary-noise assumptions below, this analysis separates within-population variation from mean-estimation error.

For an evaluation token $\mathbf{H}$, let J be uniform over its two selected experts, $p_e = \Pr(J = e)$, and $\boldsymbol{\mu}_e = \mathbb{E}[\mathbf{H} \mid J = e]$. Let $\widetilde{\boldsymbol{\mu}}_e$ be an estimate from reference observations independent of $\mathbf{H}$, either a single-step mean or, for the EMA, $\mathbf{c}_e$. Define $\mathcal{E}_{\mathrm{pop}} = \mathbb{E}\|\mathbf{H} - \widetilde{\boldsymbol{\mu}}_J\|_2^2$, where expectation includes tokens, indices, and estimates. Assume finite second moments and restrict sums to $p_e > 0$. This equal-assignment weighting is analysis-only; actual Top-2 routing is unchanged.

Proposition 2 (Stationary EMA summary error) *For expert* e*, let its* r*th nonempty optimizer-step population mean be* $\mathbf{m}_{e,r} = \boldsymbol{\mu}_e + \boldsymbol{\varepsilon}_{e,r}$*. Assume zero-mean, temporally uncorrelated noise with constant total variance* $S_e = \mathbb{E}\|\boldsymbol{\varepsilon}_{e,r}\|_2^2$*; thus observation-weighted and token-weighted means agree. For fixed initialization and* $0 \le \beta < 1$*, compare a single optimizer-step population mean with the steady-state EMA. Their summary errors are*

$$\mathcal{E}_{\mathrm{batch}} = V_{\mathrm{within}} + \sum_e p_e S_e, \qquad \mathcal{E}_{\mathrm{EMA}} = V_{\mathrm{within}} + \frac{1-\beta}{1+\beta}\sum_e p_e S_e, \tag{11}$$

where $V_{\mathrm{within}} = \mathbb{E}\|\mathbf{H} - \boldsymbol{\mu}_J\|_2^2$ *is variation within the fixed assigned populations.*

Under these assumptions, GGLR's $\beta = 0.99$ scales only the mean-estimation term by $1/199$ after initialization effects vanish (Knoth, 2018). Within-population variation is unchanged. Appendix B.3 separates grouping and estimation error; Appendix B.4 derives the proof and corrections for drift, correlation, and observation weighting.

EMA-mean estimation errors affect selection through the population center and radius and router updates through target directions. With router parameters and selection biases fixed, Appendix B.4 gives a sufficient score-gap condition for recovering the exact-reference Top-2 set. Equation 9 applies target-direction changes to the learned router; Appendix B.5 gives the same-trajectory algebraic decomposition.

5 Experiments

5.1 Experimental Setup

Model and training. All methods use the same decoder-only Transformer with 1.7 billion parameters, 16 MoE layers, and 32 routed experts per layer, selecting two experts per token. Models use an approximately 100B-token training budget on Nemotron-CC-v2, corresponding to 63,579 optimizer steps, with the same data split, document order, seed, effective batch size, and optimizer configuration. Table 7 and Appendix C provide the full architecture and training details.

Baselines. We compare GGLR with four alternatives covering learned routing and direct routing from historical population means.

- **Auxiliary-loss balancing** uses a learned Top-2 router with the Switch balancing objective and ST-MoE router z-loss (Fedus et al., 2022; Zoph et al., 2022).
- **Loss-Free Balancing (LFB)** uses learned sigmoid affinities with selection-only load biases and weights expert outputs directly by the selected affinities (Wang et al., 2024).
- **LFB + gain normalization** normalizes the selected affinities before combining expert outputs while retaining the LFB selection rule.
- **Online K-means** selects experts using cosine similarity to running expert-input centroids together with load-correcting biases, updating the centroids from assigned inputs rather than task gradients (Ahrac et al., 2026).

Both LFB variants use the original MoE input without router-specific unit normalization. These comparisons evaluate complete routing designs, with implementation details specified in Appendix C.

Evaluation. At step 63,579, we report validation loss and perplexity (PPL) on the same 100.6 million held-out Nemotron-CC-v2 next-token targets, and test PPL on the complete WikiText-2 test set of 318,523 targets. Parameters and routing state are frozen, and all methods use shared scoring protocols. Loss denotes mean next-token negative log-likelihood. Expert utilization is measured using mean absolute relative deviation (MARD) and maximal violation (MaxVio); MARD captures average deviation from uniform utilization, whereas MaxVio captures maximum overload relative to uniform utilization (Wang et al., 2024). Lower values indicate better balance. Global metrics aggregate assignments over the evaluation sample within each layer and then average across layers. Geometry is measured at 8,192 fixed token positions per layer, shared across methods, using each model's own representations and actual routing assignments. Evaluation population means are recomputed from these assignments rather than taken from GGLR's stored EMA state. Appendices A and C provide the detailed metrics, sampling procedures, and evaluation protocols.

5.2 Language Modeling and Load Balance

Prior work reports a quality–balance trade-off in MoE routing (Wang et al., 2024; Ahrac et al., 2026); in our comparison, normalizing LFB's output gains reduces global MaxVio by 92.1% while increasing validation loss by 0.006322 (Table 1). GGLR achieves the lowest observed validation loss, perplexity, global MARD, and global MaxVio among the five methods (Table 1). Compared with LFB + gain normalization, which has the best global balance among the baselines, GGLR reduces MARD by 9.5% and MaxVio by 23.1% while also lowering validation loss. Relative to standard LFB, the strongest baseline in validation loss, GGLR reduces global MaxVio by 94.0% while further lowering validation loss. The quality advantage also extends to WikiText-2, where GGLR attains the lowest test perplexity. The observed gains therefore combine improved aggregate utilization with better language modeling rather than trading one against the other.

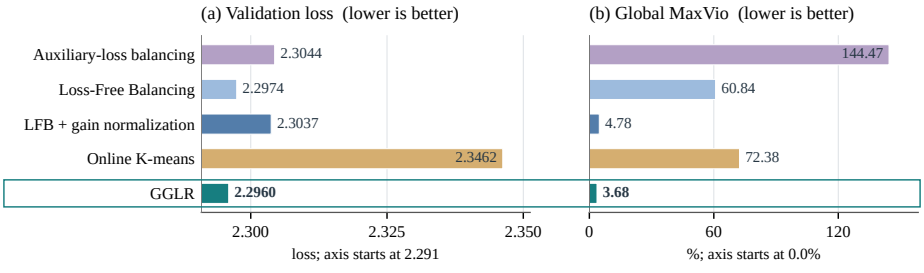


Figure 2: Validation loss and global MaxVio for the five models trained under matched conditions. Each panel compares one measure across all methods; lower is better. Colors identify methods, the box marks GGLR, and bold numbers mark the best values. MaxVio is expressed as percentage overload relative to uniform load. Cropped-axis minima are labeled; bar lengths are not ratios of the underlying values.

Table 1: Matched endpoint results for a 1.7-billion-parameter MoE at step 63,579. Nemotron-CC-v2 uses 100.6 million held-out next-token targets; WikiText-2 uses all 318,523 test targets (Appendix C.5). Global load measures aggregate assignments within each layer and average the 16 layer values. Lower is better in every column. Bold values indicate column-wise best.

	Nemotron-CC-v2				WikiText-2
Method	Loss	PPL	MARD	MaxVio	PPL
Auxiliary-loss balancing	2.304385	10.0180	0.277463	1.444699	11.1226
Loss-Free Balancing	2.297426	9.9485	0.095787	0.608418	11.1513
LFB + gain normalization	2.303748	10.0116	0.016772	0.047822	11.1786
Online K-means	2.346233	10.4461	0.176536	0.723812	11.6657
GGLR	**2.295957**	**9.9339**	**0.015173**	**0.036756**	**11.0892**

The aggregate improvement extends to the most imbalanced layer. GGLR has the lowest worst-layer MARD and MaxVio, including a 74.4% reduction in worst-layer MaxVio relative to LFB + gain normalization (Table 3). Across complete, nonoverlapping 65,536-token evaluation windows, LFB + gain normalization has lower 95th-percentile (p95) MARD and MaxVio (Table 2). GGLR's utilization advantage thus concerns global and worst-layer balance rather than uniformly lower local load variation. Appendix A reports the complete load profiles and entropy measurements.

5.3 Population and Router Geometry

GGLR has the largest population separation among all five methods and the lowest router overlap among the four learned selectors (Figure 3), consistent with the distinct population and router directions it encourages.

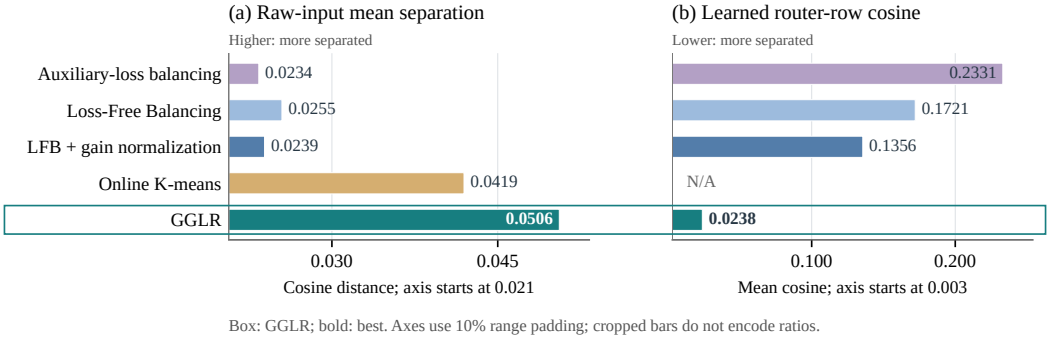


Figure 3: Endpoint input-population and router-weight geometry. Left: mean cosine distance between evaluation population means at 8,192 shared token positions per layer, with larger values indicating more separated mean directions. Right: mean cosine between distinct learned router rows, with smaller values indicating less directional overlap. Both average per-layer statistics. Colors identify methods, the box marks GGLR, and bold numbers mark the best values. The label not applicable (N/A) marks Online K-means, which has no learned router matrix.

To complement directional separation, we measure the fraction of input variation explained by expert identity using the same evaluation assignments. GGLR exceeds all learned-router baselines, while Online K-means explains more variation despite worse language-model quality (Table 5). This comparison is consistent with incorporating population geometry into task-trained routing rather than selecting directly by centroid similarity.

To relate routing to observable input content, we measure normalized mutual information (NMI) (Vinh et al., 2010) between source category and expert identity over the Nemotron-CC-v2 evaluation sample. GGLR attains the highest NMI when assignments receive equal weight. When each document contributes equal total weight, it remains above the learned-router baselines but below Online K-means (Table 4).

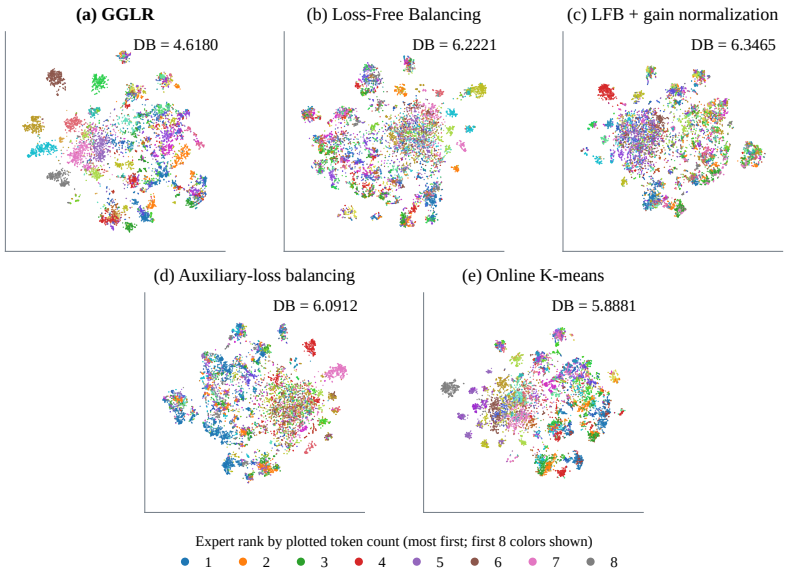


Figure 4: **GGLR and all four controlled baselines.** Final-layer inputs at the same 8,192 token positions use independently fitted t-SNE projections and equally sized panels. Colors identify the first recorded selected expert, ranked within each panel by decreasing plotted token count. One fixed palette maps ranks to colors; equal colors do not identify equivalent experts across models. Axes are model-local t-SNE coordinates. DB uses all original vectors before projection; lower is better, and GGLR has the lowest value. Each token has one visualization label; actual routing still uses two experts. Appendix C.4 gives the protocol.

Figure 4 shows independently fitted final-layer t-distributed stochastic neighbor embedding (t-SNE) projections (van der Maaten & Hinton, 2008). GGLR exhibits more distinct expert-associated local groups than the learned-router baselines and has the lowest Davies–Bouldin (DB) index, computed on the original representations before projection. The DB index compares within-group dispersion with separation between group means. Appendix C.4 gives the visualization protocol, and Appendix A.3 gives public-model structural references.

6 Discussion and Future Directions

The joint results support integrating population geometry into a task-trained router with separate load control. Population geometry describes which inputs experts receive, while load control regulates assignment mass; the local-tail and explained-variation results show that neither balanced counts nor a single geometric statistic alone fully characterizes the outcomes.

GGLR incorporates historical expert-input geometry through population-relative selection coordinates and target-direction changes alongside task-optimizer updates. Future validation should test these relationships at larger model scales, token budgets, and routing settings.

7 Conclusion

In the matched 1.7-billion-parameter, approximately 100B-token Nemotron-CC-v2 setting, GGLR uses historical EMA expert-input means to define population-relative selection coordinates and guide updates of a task-trained router. Among five routing methods, GGLR has the lowest observed held-out loss and global load imbalance, together with more separated evaluation population means and less overlap among learned router directions.

Reproducibility Statement

We provide an anonymized supplementary package containing the implementation and configuration templates, training and validation launchers, plotting scripts, a result ledger, and compact de-identified table and coordinate data. Appendix C specifies the training and evaluation procedures, including initialization, random-state handling, optimizer and numerical settings, data processing, and evaluation protocols, while Appendix B contains the assumptions and proofs for the theoretical results. The included plotting commands reproduce the ledger-based and compact-coordinate figures without model loading. Training, perplexity evaluation, and checkpoint-dependent validation require the external dataset, tokenizer, runtime environment, and model checkpoints identified in Appendix C and the supplementary README.

Artificial intelligence (AI) use statement

Large language model (LLM) assistants were used to organize, draft, and revise the manuscript and to help develop and check idealized mathematical analyses. LLM assistants also helped search for and identify relevant literature. The experimental measurements come from existing model evaluations, not generated data. Responsibility for the manuscript's claims, citations, analysis, and reported results rests with the authors.

References

Sagi Ahrac, Noya Hochwald, and Mor Geva. Routers learn the geometry of their experts: Geometric coupling in sparse mixture-of-experts, 2026. URL https://arxiv.org/abs/2605.12476.

Arindam Banerjee, Inderjit S. Dhillon, Joydeep Ghosh, and Suvrit Sra. Clustering on the unit hypersphere using von Mises-Fisher distributions. *JMLR*, 2005. URL https://www.jmlr.org/beta/papers/v6/banerjee05a.html.

Geetanjali Bihani and Julia Rayz. Low anisotropy sense retrofitting (LASeR): Towards isotropic and sense enriched representations. In *Proceedings of Deep Learning Inside Out (DeeLIO)*, pp. 81–95, 2021. doi: 10.18653/v1/2021.deelio-1.9. URL https://aclanthology.org/2021.deelio-1.9/.

Lizhang Chen, Jonathan Li, Qi Wang, et al. ϕ-balancing for mixture-of-experts training, 2026. URL https://arxiv.org/abs/2605.15403.

Damai Dai, Li Dong, Shuming Ma, et al. StableMoE: Stable routing strategy for mixture of experts. In *ACL*, 2022. doi: 10.18653/v1/2022.acl-long.489. URL https://aclanthology.org/2022.acl-long.489/.

Damai Dai, Chengqi Deng, Chenggang Zhao, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2401.06066.

DeepSeek-AI, Aixin Liu, Bei Feng, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

DeepSeek-AI, Aixin Liu, Bei Feng, et al. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Nan Du, Yanping Huang, Andrew M. Dai, et al. Glam: Efficient scaling of language models with mixture-of-experts, 2022. URL https://arxiv.org/abs/2112.06905.

Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 55–65, 2019. doi: 10.18653/v1/D19-1006. URL https://aclanthology.org/D19-1006/.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2022. URL https://arxiv.org/abs/2101.03961.

Alejandro Fuster Baggetto and Víctor Fresno. Is anisotropy really the cause of BERT embeddings not being semantic? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 4271–4281, 2022. doi: 10.18653/v1/2022.findings-emnlp.314. URL https://aclanthology.org/2022.findings-emnlp.314/.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts, 2022. URL https://arxiv.org/abs/2211.15841.

Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. Fastmoe: A fast mixture-of-expert training system, 2021. URL https://arxiv.org/abs/2103.13262.

Evan Hernandez and Jacob Andreas. The low-dimensional linear geometry of contextualized word representations. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pp. 82–93, 2021. doi: 10.18653/v1/2021.conll-1.7. URL https://aclanthology.org/2021.conll-1.7/.

John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4129–4138, 2019. doi: 10.18653/v1/N19-1419. URL https://aclanthology.org/N19-1419/.

Changho Hwang, Wei Cui, Yifan Xiong, et al. Tutel: Adaptive mixture-of-experts at scale, 2023. URL https://arxiv.org/abs/2206.03382.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, et al. Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

Sven Knoth. The steady-state behavior of multivariate exponentially weighted moving average control charts, 2018. URL https://arxiv.org/abs/1808.05069.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, et al. Sparse upcycling: Training mixture-of-experts from dense checkpoints, 2023. URL https://arxiv.org/abs/2212.05055.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 3519–3529. PMLR, 2019. URL https://proceedings.mlr.press/v97/kornblith19a.html.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, et al. Gshard: Scaling giant models with conditional computation and automatic sharding, 2020. URL https://arxiv.org/abs/2006.16668.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models, 2021. URL https://arxiv.org/abs/2103.16716.

Weiyang Liu, Rongmei Lin, Zhen Liu, Li Xiong, Bernhard Schölkopf, and Adrian Weller. Learning with hyperspherical uniformity. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, pp. 1180–1188. PMLR, 2021. URL https://proceedings.mlr.press/v130/liu21d.html.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843.

Stefan K. Nielsen, Rachel S. Y. Teo, Laziz U. Abdullaev, and Tan M. Nguyen. Tight clusters make specialized experts. In *ICLR*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/854a9ab0f323b841955e70ca383b27d1-Abstract-Conference.html.

NVIDIA, Aarti Basant, Abhijit Khairnar, et al. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model, 2025. URL https://arxiv.org/abs/2508.14444.

Nabil Omi, Siddhartha Sen, and Ali Farhadi. Load balancing mixture of experts with similarity preserving routers. *arXiv preprint arXiv:2506.14038*, 2025. URL https://arxiv.org/abs/2506.14038.

Sara Rajaee and Mohammad Taher Pilehvar. How does fine-tuning affect the geometry of embedding space: A case study on isotropy. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 3042–3049, 2021. doi: 10.18653/v1/2021.findings-emnlp.261. URL https://aclanthology.org/2021.findings-emnlp.261/.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, et al. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale, 2022. URL https://arxiv.org/abs/2201.05596.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, et al. Scaling vision with sparse mixture of experts, 2021. URL https://arxiv.org/abs/2106.05974.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. Hash layers for large sparse models, 2021. URL https://arxiv.org/abs/2106.04426.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, et al. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL https://arxiv.org/abs/1701.06538.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL https://jmlr.org/papers/v9/vandermaaten08a.html.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11 (95):2837–2854, 2010. URL https://jmlr.org/papers/v11/vinh10a.html.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts, 2024. URL https://arxiv.org/abs/2408.15664.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 9929–9939. PMLR, 2020. URL https://proceedings.mlr.press/v119/wang20k.html.

Forest Yang and Sanmi Koyejo. On the consistency of top-k surrogate losses. In *ICML*, 2020. URL https://proceedings.mlr.press/v119/yang20f.html.

Jiajie Yang. Latent prototype routing: Achieving near-perfect load balancing in mixture-of-experts. *arXiv preprint arXiv:2506.21328*, 2025. URL https://arxiv.org/abs/2506.21328.

Yongyi Yang, Jacob Steinhardt, and Wei Hu. Are neurons actually collapsed? on the fine-grained structure in neural representations. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 39453–39487. PMLR, 2023. URL https://proceedings.mlr.press/v202/yang23m.html.

Biao Zhang and Rico Sennrich. Root mean square layer normalization, 2019. URL https://arxiv.org/abs/1910.07467.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Moefication: Transformer feed-forward layers are mixtures of experts, 2022. URL https://arxiv.org/abs/2110.01786.

Barret Zoph, Irwan Bello, Sameer Kumar, et al. St-moe: Designing stable and transferable sparse expert models, 2022. URL https://arxiv.org/abs/2202.08906.

A Supplementary Results

These tables supplement the five-method comparison in Table 1. All controlled Nemotron-CC-v2 results use the protocol in Appendix C; public-model measurements use their separate native samples.

A.1 Load and Source Association

For layer ℓ in the evaluation sample, let $p_{\ell e}$ be expert e's share of all actual Top-2 assignments, with $\sum_{e=1}^{E} p_{\ell e} = 1$. We define

$$\mathrm{MARD}_\ell = \frac{1}{E}\sum_{e=1}^{E} |Ep_{\ell e} - 1|, \qquad \mathrm{MaxVio}_\ell = \max_{e=1,\ldots,E}(Ep_{\ell e} - 1). \tag{12}$$

Here, E is the number of experts per layer. MARD averages absolute relative deviations from uniform assignment, whereas MaxVio measures the largest overload (Wang et al., 2024); lower values indicate better balance. Global values aggregate all evaluation assignments within each layer and then average across layers. By contrast, training selection frequencies divide per-microbatch assignment counts by the number of tokens and sum to K across experts. For local load tails, we compute MARD and MaxVio in every complete, nonoverlapping 65,536-token window, take

each measure's linearly interpolated p95 within each layer, and average the 16 percentiles. Lower values indicate smaller local imbalance. Normalized load entropy is $-\sum_e p_{\ell e} \log p_{\ell e} / \log E$, using aggregate assignment shares and setting zero-mass terms to zero. Its 16-layer mean reaches one under uniform utilization.

Table 2 supplements the quality and global-load comparison in Table 1. Both tail measures take the p95 over complete 65,536-token windows within each layer and then average the 16 layer values. Entropy uses the same aggregate assignment shares as the global deviations.

Table 2: Supplementary load results on the shared evaluation sample. Tail measures are per-layer window p95 values averaged across layers; lower is better. Normalized entropy measures aggregate uniformity; higher is better. Bold values indicate column-wise best.

Method	Tail MARD	Tail MaxVio	Entropy
Auxiliary-loss balancing	0.343330	1.711739	0.980388
Loss-Free Balancing	0.262151	1.766937	0.984779
LFB + gain normalization	**0.175599**	**1.050526**	0.999895
Online K-means	0.320822	1.543625	0.980506
GGLR	0.211381	1.296055	**0.999938**

Table 3 reports the maximum of each global-load measure across the 16 layers, together with its zero-indexed layer. These maxima expose concentrated imbalance that layer averages can dilute.

Table 3: Worst-layer global load on the shared evaluation sample; lower is better. Each maximum is taken after aggregating all evaluation assignments within each layer. Bold values indicate column-wise best.

	Largest MARD		Largest MaxVio	
Method	Value	Layer	Value	Layer
Auxiliary-loss balancing	0.469462	15	2.521183	15
Loss-Free Balancing	0.616244	1	4.989818	1
LFB + gain normalization	0.072329	0	0.182296	0
Online K-means	0.862501	0	5.891560	0
GGLR	**0.018927**	13	**0.046696**	3

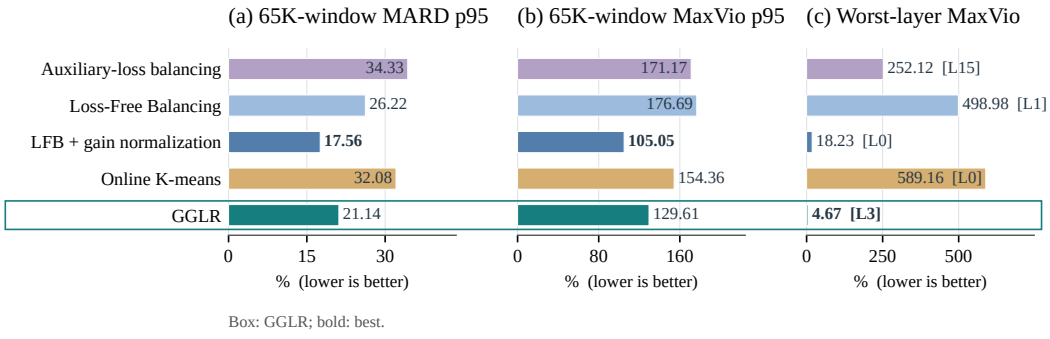


Figure 5: Local-window and worst-layer load comparisons; lower is better. Mean absolute relative deviation (MARD) is the mean absolute percentage deviation from uniform expert load; maximal violation (MaxVio) is the maximum percentage overload. In the panel labels, 65K denotes a 65,536-token window and L denotes the layer index. Window tails average per-layer p95 values over 65,536-token windows; worst-layer MaxVio takes the maximum after global aggregation within each layer. Colors identify methods, the box marks GGLR, and bold numbers mark the best values. GGLR has the lowest worst-layer overload; LFB with gain normalization has the lower local tails.

Source association uses NMI between Nemotron-CC-v2 source category and expert. Assignment weighting gives every actual Top-2 assignment equal mass; document weighting gives each document unit total mass. Results are averaged over 16 layers. Higher values indicate stronger association, not correctness of expert choice. The document weighting removes direct document-length weighting. The shuffle reference permutes source-category labels across documents once using seed 20260814, leaves expert assignments unchanged, and uses the same permutation for both weighting schemes. Table 4 reports NMI under assignment and document weighting together with the shuffle reference. NMI divides mutual information by the geometric mean of category and expert entropies. For all five methods, NMI exceeds the corresponding shuffled-label value under both weighting schemes.

Table 4: Source-category/expert NMI, averaged across 16 layers; higher NMI indicates stronger source association. Each shuffle column uses the same fixed document-label permutation and is not ranked. Bold values indicate column-wise best.

	Assignment weighting		Document weighting	
Method	NMI	Shuffle	NMI	Shuffle
Auxiliary-loss balancing	0.028399	0.000023	0.021265	0.000006
Loss-Free Balancing	0.033878	0.000035	0.025945	0.000008
LFB + gain normalization	0.029069	0.000026	0.022351	0.000006
Online K-means	0.037418	0.000031	**0.031820**	0.000007
GGLR	**0.037944**	0.000037	0.030188	0.000008

A.2 Input and Router Geometry

At each of 16 MoE layers, actual Top-2 memberships group raw pre-router hidden states at 8,192 fixed positions. Each token contributes equally to both selected experts. We compute an evaluation population mean $\overline{\mathbf{h}}_e$, the arithmetic mean of the assigned sample, for each active expert, independently of any stored EMA population mean. Input separation averages cosine distance over distinct expert-mean pairs and then over layers. The explained fraction averages the 16 ratios $V_{\text{between}}/V_{\text{total}}$, computed using the same-sample assignment shares and means. It measures the fractional reduction in squared summary error from conditioning on expert identity. Both measures apply to all five routers through their actual assignments.

Each router-weight row supplies an expert-specific direction and scale for selection. To measure directional overlap, we normalize the rows, exclude self-pairs, average pairwise cosines within each layer, and then average layers. This statistic covers the four learned selectors; Online K-means has no learned router matrix.

Table 5 connects the assigned-input analysis to the summary-error decomposition. Evaluation population means use the shared 8,192-position sample. Explained is the layer-mean fraction of same-sample squared variation removed by expert-specific means. Each checkpoint supplies its own representation, so the fraction provides a scale-normalized comparison rather than a ranking of absolute input variances. Router-row values are shown directly in Figure 3.

Table 5: Assigned-input geometry. Input separation is the mean pairwise cosine distance between evaluation population means. Explained averages the same-sample squared-error reduction fraction across layers. Larger values indicate greater separation or explained variation, respectively. Bold values indicate column-wise best.

Method	Input separation	Explained
Auxiliary-loss balancing	0.02339	6.72%
Loss-Free Balancing	0.02548	6.17%
LFB + gain normalization	0.02391	6.36%
Online K-means	0.04195	**11.83%**
GGLR	**0.05057**	9.30%

A.3 Public-Model Structural References

DeepSeek-V4-Flash and MiniMax-M2.7 provide a separate structural comparison. Their final-layer projections show expert-associated local groups (Figure 6), alongside separated directions of expert input means and learned router-row geometry across layers (Table 6). For the evaluation population mean statistics, we retain the first two recorded expert indices per token from their native Top-6 and Top-8 routes. We view this organization in independently trained models as a structural parallel to the geometry GGLR encourages explicitly. Each public model is measured in its own representation space and native-tokenizer sample; Figure 4 reports GGLR's lowest final-layer DB among the five controlled methods. Appendix C specifies layer sets and sampling.

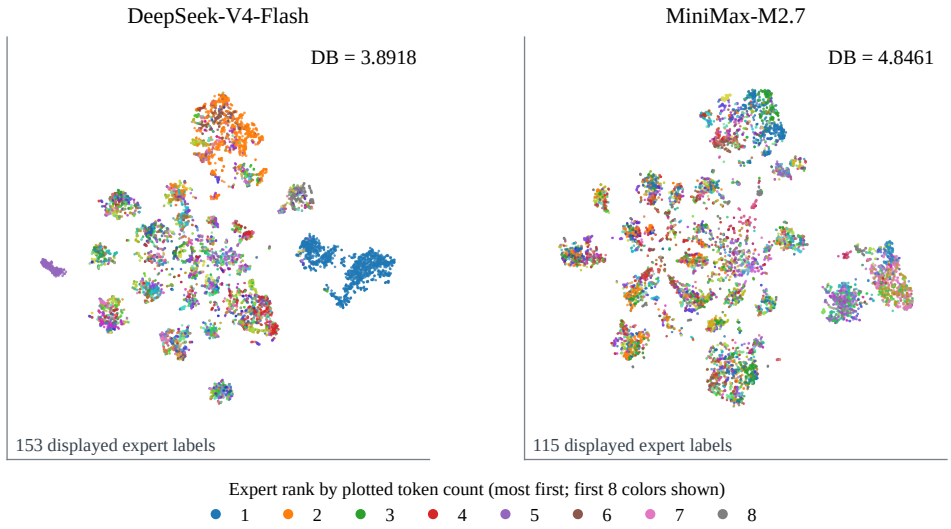


Figure 6: **Public-model structural comparisons.** Independently fitted final-layer t-SNE projections show 8,192 inputs per model. Colors label the first recorded selected expert, ranked by plotted token count with the same palette as Figure 4. The samples contain 153 and 115 distinct visualization labels, respectively. Axes are model-local t-SNE coordinates. DB uses original vectors and the same definition as Figure 4; lower is better. Different native-tokenizer samples and representation spaces make these structural comparisons, not a controlled ranking against GGLR.

Table 6: Input and router geometry in two public MoE models. Evaluation population means use the first two recorded expert indices per token from native Top-6 routing in DeepSeek and Top-8 routing in MiniMax. Input separation is the layer-mean cosine distance between these means; router cosine is the layer-mean cosine between learned rows. Different native samples and coordinates make these model-local descriptions, not a cross-model ranking.

Model	Input separation	Router-row cosine
DeepSeek-V4-Flash	0.57539	0.16722
MiniMax-M2.7	0.55623	0.37073

B Population Geometry and Historical Estimation

B.1 Population-Relative Coordinates and Common Components

For a shared-component example, write $\mathbf{h} = \mathbf{a} + \mathbf{u}$ and $\mathbf{c}_e = \mathbf{a} + \mathbf{v}_e$, where $\sum_e \mathbf{v}_e = 0$ and the nonzero residuals $\mathbf{u}, \mathbf{v}_e$ are orthogonal to $\mathbf{a}$. At the target direction $\mathbf{w}_e = \mathbf{v}_e / \|\mathbf{v}_e\|_2$, the orthogonality condition in Proposition 1 holds for every pair.

For Proposition 1, expand the centered pairwise logit difference:

$$\frac{\mathbf{d}_{ej}^{\mathsf{T}}(\mathbf{h}-\mathbf{a})}{R} = \frac{\|\mathbf{h}\|_2}{R}\frac{\mathbf{d}_{ej}^{\mathsf{T}}\mathbf{h}}{\|\mathbf{h}\|_2} - \frac{\mathbf{d}_{ej}^{\mathsf{T}}\mathbf{a}}{R}. \tag{13}$$

When the two rows respond equally to the population center, the last term vanishes. In the shared-component model, each target direction is orthogonal to that center, which supplies this condition without constraining residual directions to be mutually orthogonal. The amplification concerns nonzero pairwise logit differences; a zero difference remains zero.

For example, take two EMA population means $(10, 1)$ and $(10, -1)$, their center $(10, 0)$ and radius 1, target directions $(0, 1)$ and $(0, -1)$, and token $(10, 0.2)$. Unit-input logits have difference $0.4/\sqrt{100.04}$; relative-input logits have difference 0.4. The factor $\sqrt{100.04}$ comes from the shared component dominating the raw norm. This is an illustrative coordinate calculation, not a measurement from the trained model. For arbitrary learned rows the removed center can change the ordering, and larger logit differences do not universally imply larger sigmoid differences. Actual selection uses both transformed affinities and expert-dependent biases.

Shared radius scaling preserves residual-length information: for a fixed center and positive radius, the selection-input distance between two tokens is $\|\mathbf{h}_i - \mathbf{h}_j\|_2/R$. Subtracting the center alone preserves that distance exactly. The useful change is the reference direction and scale at which the router compares inputs, rather than creation of new information in the hidden representation.

B.2 Equal Loads, Different Population Organization

Consider four equally likely token populations on the unit circle, with population $Z = e$ assigned a predefined specialist e. Their centers are $\alpha_e = e\pi/2$ for $e \in \{0, 1, 2, 3\}$, and $\mathbf{X} = (\cos\Theta, \sin\Theta)$. For common concentration $\kappa > 0$, define the periodic density

$$f_\kappa(u) = \frac{\exp(\kappa\cos u)}{D_\kappa}, \qquad D_\kappa = \int_{-\pi}^{\pi}\exp(\kappa\cos u)\,du, \qquad p(\theta \mid Z = e) = f_\kappa(\theta - \alpha_e). \tag{14}$$

This is the two-dimensional von Mises–Fisher model (Banerjee et al., 2005); its marginal density is $p_\Theta(\theta) = \frac{1}{4}\sum_{e=0}^{3} f_\kappa(\theta-\alpha_e)$. Specialist identity is a modeling premise, independent of the evaluated router, unlike the assignment-derived population means in Section 4.

Rotate every router direction by the same angle ϕ, with common positive row norm τ and common bias b:

$$\mathbf{w}_e^\phi = \tau(\cos(\alpha_e + \phi), \sin(\alpha_e + \phi)), \qquad S_\phi(\theta) = \mathrm{Top2}_e\{\sigma((\mathbf{w}_e^\phi)^{\mathsf{T}}\mathbf{X}) + b\}. \tag{15}$$

These scores preserve the ordering of inner products. Two opposite direction pairs make Top-2 select the two positive inner products, except at zero-probability boundaries. Each expert's region spans a half-turn. The marginal density repeats every quarter-turn, giving

$$\Pr\{e \in S_\phi(\Theta)\} = \int_{\alpha_e+\phi-\pi/2}^{\alpha_e+\phi+\pi/2} p_\Theta(\theta)\,d\theta = \frac{1}{2}. \tag{16}$$

Each adjacent expert pair also occupies a quarter-turn with probability one quarter. Thus all rotations preserve both expert loads and expert-pair frequencies.

The probability of excluding the predefined specialist, $\mathcal{R}(\phi)$, nevertheless depends on orientation. Translating by each population center yields

$$\begin{aligned}\mathcal{R}(\phi) &:= \Pr\{Z \notin S_\phi(\Theta)\} = 1 - \int_{\phi-\pi/2}^{\phi+\pi/2} f_\kappa(u)\,du,\\ \mathcal{R}'(\phi) &= \frac{\exp(\kappa\sin\phi) - \exp(-\kappa\sin\phi)}{D_\kappa} > 0, \qquad 0 < \phi < \pi.\end{aligned} \tag{17}$$

Bayes' rule gives $\Pr(Z = e \mid \mathbf{X} = \mathbf{x}) = \exp(\kappa\mathbf{u}_e^{\mathsf{T}}\mathbf{x})/\sum_j \exp(\kappa\mathbf{u}_j^{\mathsf{T}}\mathbf{x})$, with $\mathbf{u}_e = (\cos\alpha_e, \sin\alpha_e)$. Selecting the two largest probabilities minimizes conditional exclusion probability (Yang & Koyejo, 2020). The aligned linear router therefore minimizes the probability of excluding the predefined specialist. Numerical integration at $\kappa = 4$ gives $\mathcal{R}(0) = 0.00755941$ and $\mathcal{R}(\pi/4) = 0.07121642$, with selection probability 0.5 for every expert in both cases. The calculation isolates geometric information that balanced counts alone do not specify.

B.3 Population Summaries and Observable Geometry

Fix the input distribution and routing rule, with finite second moments. The auxiliary assignment index gives each of a token's two actual assignments half its mass:

$$J \mid \mathbf{H} \sim \mathrm{Uniform}(\mathcal{A}(\mathbf{H})), \qquad p_e = \Pr(J = e), \qquad \boldsymbol{\mu}_e = \mathbb{E}[\mathbf{H} \mid J = e]. \tag{18}$$

p_e is an assignment share and $\boldsymbol{\mu}_e$ its token-weighted conditional population mean; sums include only $p_e > 0$. Independent reference observations determine the reference mean estimates $\widetilde{\boldsymbol{\mu}}_e$. The expected squared summary error decomposes as

$$\underbrace{\mathbb{E}\|\mathbf{H} - \widetilde{\boldsymbol{\mu}}_J\|_2^2}_{\mathcal{E}_{\mathrm{pop}}} = \underbrace{\mathbb{E}\|\mathbf{H} - \boldsymbol{\mu}_J\|_2^2}_{V_{\mathrm{within}}} + \underbrace{\sum_e p_e \mathbb{E}\|\widetilde{\boldsymbol{\mu}}_e - \boldsymbol{\mu}_e\|_2^2}_{V_{\mathrm{estimate}}}. \tag{19}$$

The expectation covers the token, assignment index, and reference estimates. The two terms measure within-population variation and mean-estimation error, respectively. Balanced loads alone do not determine population organization, as Appendix B.2 demonstrates. For the input distribution in Equation 18, the overall mean is $\boldsymbol{\mu} = \mathbb{E}[\mathbf{H}] = \sum_e p_e \boldsymbol{\mu}_e$. Under squared Euclidean error, it is the best common constant summary. Replacing it with the expert-specific conditional means gives

$$\begin{aligned} V_{\mathrm{total}} &:= \mathbb{E}\|\mathbf{H} - \boldsymbol{\mu}\|_2^2, \qquad V_{\mathrm{within}} := \mathbb{E}\|\mathbf{H} - \boldsymbol{\mu}_J\|_2^2, \\ V_{\mathrm{total}} - V_{\mathrm{within}} &= V_{\mathrm{between}} := \sum_e p_e \|\boldsymbol{\mu}_e - \boldsymbol{\mu}\|_2^2 \\ &= \frac{1}{2} \sum_{e,j} p_e p_j \|\boldsymbol{\mu}_e - \boldsymbol{\mu}_j\|_2^2. \end{aligned} \tag{20}$$

Thus $V_{\mathrm{between}}/V_{\mathrm{total}}$, for positive total variation, is the fractional reduction in squared summary error. Combining this identity with Equation 19, estimated means improve on the true common mean when $V_{\mathrm{estimate}} < V_{\mathrm{between}}$.

For the fixed distribution and routing rule in Equation 18, expanding $\mathbf{H} - \boldsymbol{\mu} = (\mathbf{H} - \boldsymbol{\mu}_J) + (\boldsymbol{\mu}_J - \boldsymbol{\mu})$ gives $V_{\mathrm{total}} = V_{\mathrm{within}} + V_{\mathrm{between}}$. The cross term vanishes because $\mathbb{E}[\mathbf{H} - \boldsymbol{\mu}_J \mid J] = 0$. To express the resulting error reduction through mean distances, use $\sum_e p_e = 1$:

$$\begin{aligned} V_{\mathrm{between}} &= \sum_e p_e \|\boldsymbol{\mu}_e\|_2^2 - \left\| \sum_e p_e \boldsymbol{\mu}_e \right\|_2^2 \\ &= \frac{1}{2} \sum_{e,j} p_e p_j \left(\|\boldsymbol{\mu}_e\|_2^2 + \|\boldsymbol{\mu}_j\|_2^2 - 2\boldsymbol{\mu}_e^{\mathsf{T}} \boldsymbol{\mu}_j \right) \\ &= \frac{1}{2} \sum_{e,j} p_e p_j \|\boldsymbol{\mu}_e - \boldsymbol{\mu}_j\|_2^2. \end{aligned} \tag{21}$$

The same identity holds exactly on a finite input sample using its assignment shares and evaluation population means $\overline{\mathbf{h}}_e$, without token independence. That identity measures in-sample organization; independent evaluation inputs measure how estimated summaries describe new content.

Let independent reference data determine the random reference mean estimates $\{\widetilde{\boldsymbol{\mu}}_e\}$ for the same populations. Expand the evaluation error around each true conditional mean:

$$\begin{aligned} \mathbb{E}\|\mathbf{H} - \widetilde{\boldsymbol{\mu}}_J\|_2^2 = \mathbb{E}\|\mathbf{H} - \boldsymbol{\mu}_J\|_2^2 + \mathbb{E}\|\boldsymbol{\mu}_J - \widetilde{\boldsymbol{\mu}}_J\|_2^2 \\ + 2\mathbb{E}[(\mathbf{H} - \boldsymbol{\mu}_J)^{\mathsf{T}} (\boldsymbol{\mu}_J - \widetilde{\boldsymbol{\mu}}_J)]. \end{aligned} \tag{22}$$

Independence of reference and evaluation data implies $\mathbb{E}[\mathbf{H} - \boldsymbol{\mu}_J \mid J, \{\widetilde{\boldsymbol{\mu}}_e\}] = 0$, so the cross term vanishes. Averaging the final squared term over expert probabilities yields Equation 19. In particular, each true conditional mean minimizes error among constant summaries of its fixed expert population.

Assume at least two experts have positive assignment probability and every such expert has a nonzero mean. Set $\nu_e = \|\boldsymbol{\mu}_e\|_2$, $\nu_{\min} = \min_e \nu_e$, $d_{ej} = 1 - \boldsymbol{\mu}_e^{\mathsf{T}} \boldsymbol{\mu}_j / (\nu_e \nu_j)$, and $d_e = \min_{j \neq e} d_{ej}$. The distance expansion

$$\|\boldsymbol{\mu}_e - \boldsymbol{\mu}_j\|_2^2 = (\nu_e - \nu_j)^2 + 2\nu_e \nu_j d_{ej} \tag{23}$$

and nonnegativity give

$$V_{\text{between}} \geq \nu_{\min}^2 \sum_{e \neq j} p_e p_j d_{ej} \geq \nu_{\min}^2 \sum_e p_e(1-p_e)d_e. \tag{24}$$

For positive V_{total}, the fractional reduction in squared summary error therefore satisfies

$$\frac{V_{\text{between}}}{V_{\text{total}}} \geq \frac{\nu_{\min}^2}{V_{\text{total}}} \sum_e p_e(1-p_e)d_e. \tag{25}$$

Nearest cosine distances give a lower bound on the error reduction when combined with mean lengths, assignment weights, and input scale. A small minimum mean length weakens this bound even when the directions are separated. Unweighted cosine averages alone cannot rank within-population error across routers; the complete variance ratio measures fractional reduction in squared summary error.

B.4 EMA Error, Drift, and Observation Weights

For the single-expert calculations below, suppress the expert index; it is restored when comparing population centers and target directions. In the stationary model of Section 4, $\boldsymbol{\mu}$ is both the token-weighted population mean and the expected nonempty optimizer-step population mean. Subtract $\boldsymbol{\mu}$ from Equation 7 and let $\mathbf{e}_0 = \mathbf{c}_0 - \boldsymbol{\mu}$. Recursive substitution yields

$$\mathbf{c}_r - \boldsymbol{\mu} = \beta^r \mathbf{e}_0 + (1-\beta)\sum_{s=1}^{r} \beta^{r-s}\boldsymbol{\varepsilon}_s. \tag{26}$$

Fixed initialization and zero-mean, uncorrelated observations remove all cross terms. With equal noise variance S,

$$\begin{aligned} \mathbb{E}\|\mathbf{c}_r - \boldsymbol{\mu}\|_2^2 &= \beta^{2r}\|\mathbf{e}_0\|_2^2 + (1-\beta)^2 S \sum_{s=1}^{r} \beta^{2(r-s)} \\ &= \beta^{2r}\|\mathbf{e}_0\|_2^2 + S\frac{1-\beta}{1+\beta}(1-\beta^{2r}). \end{aligned} \tag{27}$$

Applying these estimators to independent new tokens and averaging by expert shares gives Equation 11.

Let $\boldsymbol{\mu}_r = \boldsymbol{\mu}_0 + r\mathbf{v}$, retaining the noise assumptions above. For error against the current mean, $\mathbf{e}_r = \mathbf{c}_r - \boldsymbol{\mu}_r$, subtraction gives $\mathbf{e}_r = \beta\mathbf{e}_{r-1} - \beta\mathbf{v} + (1-\beta)\boldsymbol{\varepsilon}_r$. Expanding this recurrence gives

$$\mathbb{E}\|\mathbf{e}_r\|_2^2 = \left\|\beta^r \mathbf{e}_0 - \frac{\beta(1-\beta^r)}{1-\beta}\mathbf{v}\right\|_2^2 + S\frac{1-\beta}{1+\beta}(1-\beta^{2r}). \tag{28}$$

Taking the limit gives

$$\lim_{r\to\infty} \mathbb{E}\|\mathbf{c}_r - \boldsymbol{\mu}_r\|_2^2 = \frac{1-\beta}{1+\beta}S + \frac{\beta^2}{(1-\beta)^2}\|\mathbf{v}\|_2^2. \tag{29}$$

The terms represent retained noise and tracking lag; $\mathbf{v}$ is the mean's change per nonempty observation. For $S > 0$ and $0 < \beta < 1$, comparison with the current optimizer-step population mean yields

$$\lim_{r\to\infty} \mathbb{E}\|\mathbf{e}_r\|_2^2 < S \quad \Longleftrightarrow \quad \frac{\|\mathbf{v}\|_2}{\sqrt{S}} < (1-\beta)\sqrt{\frac{2}{\beta(1+\beta)}}. \tag{30}$$

At GGLR's $\beta = 0.99$, the noise coefficient is $1/199$ and the current-mean lag is $99\mathbf{v}$. Predicting the next observation's mean changes the lag to $100\mathbf{v}$. These coefficients follow from the configured momentum; actual noise and drift must be measured to evaluate the tradeoff during training.

Correlated observations change the amount of removable noise. For stationary zero-mean noise around a fixed mean, define $\gamma_k = \mathbb{E}[\boldsymbol{\varepsilon}_r^{\mathsf{T}}\boldsymbol{\varepsilon}_{r-k}]$ and assume $\sum_{k=0}^{\infty}|\gamma_k| < \infty$. Taking the long-run limit and grouping squared-noise terms by lag gives the steady-state EMA noise variance

$$V_{\text{EMA,noise}} = \frac{1-\beta}{1+\beta}\left(\gamma_0 + 2\sum_{k=1}^{\infty}\beta^k\gamma_k\right). \tag{31}$$

Zero lag contributes once and positive lags twice. γ_0 is the single-observation total noise variance. The $1/199$ factor at $\beta = 0.99$ is recovered when all nonzero-lag covariances vanish; measured temporal correlations provide the corresponding correction.

GGLR assigns the same new-observation coefficient to every nonempty optimizer-step population mean, rather than equal weight to every historical token. Conditional on positive counts $n_1, \dots, n_r$, assume independent observations containing independent, identically distributed population tokens with covariance $\mathbf{\Sigma} = \mathrm{Cov}(\mathbf{H} \mid J = e)$. For fixed initialization,

$$\mathrm{Cov}(\mathbf{c}_r \mid n_1, \dots, n_r) = (1-\beta)^2 \sum_{s=1}^{r} \frac{\beta^{2(r-s)}}{n_s} \mathbf{\Sigma}. \tag{32}$$

Counts determine the precision of each observed mean. More generally, after initialization effects vanish, the stationary EMA expectation equals $\mathbb{E}[\mathbf{m}]$. If observation content and count covary, weighting by tokens gives a different population mean:

$$\boldsymbol{\mu}_{\text{token}} = \frac{\mathbb{E}[n\mathbf{m}]}{\mathbb{E}[n]} = \mathbb{E}[\mathbf{m}] + \frac{\mathrm{Cov}(n, \mathbf{m})}{\mathbb{E}[n]}. \tag{33}$$

The covariance vector contains count/mean-coordinate covariances. The main unbiased-observation model assumes the two population means agree. Any offset between these population means contributes a squared bias to the mean-estimation term in Equation 19; evaluating EMA therefore requires specifying the mean estimate's weighting.

Normalization translates raw estimation error into directional error. For nonzero $\mathbf{c}$ and $\boldsymbol{\mu}$, insert $\mathbf{c}/\|\boldsymbol{\mu}\|_2$ and apply the triangle inequality:

$$\begin{aligned} \left\| \frac{\mathbf{c}}{\|\mathbf{c}\|_2} - \frac{\boldsymbol{\mu}}{\|\boldsymbol{\mu}\|_2} \right\|_2 &\le \left\| \frac{\mathbf{c}}{\|\mathbf{c}\|_2} - \frac{\mathbf{c}}{\|\boldsymbol{\mu}\|_2} \right\|_2 + \frac{\|\mathbf{c} - \boldsymbol{\mu}\|_2}{\|\boldsymbol{\mu}\|_2} \\ &= \frac{\big|\|\boldsymbol{\mu}\|_2 - \|\mathbf{c}\|_2\big|}{\|\boldsymbol{\mu}\|_2} + \frac{\|\mathbf{c} - \boldsymbol{\mu}\|_2}{\|\boldsymbol{\mu}\|_2} \\ &\le \frac{2\|\mathbf{c} - \boldsymbol{\mu}\|_2}{\|\boldsymbol{\mu}\|_2}. \end{aligned} \tag{34}$$

The last step uses the reverse-triangle inequality. Mean length controls error amplification, so estimation measurements should retain it when opposing input directions largely cancel.

For all E expert populations with positive probability, let $\mathbf{a} = E^{-1} \sum_e \mathbf{c}_e$ and $\mathbf{a}^* = E^{-1} \sum_e \boldsymbol{\mu}_e$. These centers weight experts equally, whereas the overall input mean $\boldsymbol{\mu} = \sum_e p_e \boldsymbol{\mu}_e$ in Appendix B.3 is assignment-weighted. With time indices omitted, write $\mathbf{T}_e$ for the target direction. Apply Equation 34 to their nonzero centered residuals and use the triangle inequality to obtain

$$\left\| \mathbf{T}_e - \frac{\boldsymbol{\mu}_e - \mathbf{a}^*}{\|\boldsymbol{\mu}_e - \mathbf{a}^*\|_2} \right\|_2 \le \frac{2\big(\|\mathbf{c}_e - \boldsymbol{\mu}_e\|_2 + \|\mathbf{a} - \mathbf{a}^*\|_2\big)}{\|\boldsymbol{\mu}_e - \mathbf{a}^*\|_2}. \tag{35}$$

If $\boldsymbol{\delta}_e = \mathbf{c}_e - \boldsymbol{\mu}_e$, Jensen's inequality gives $\mathbb{E}\|\mathbf{a} - \mathbf{a}^*\|_2^2 \le E^{-1} \sum_e \mathbb{E}\|\boldsymbol{\delta}_e\|_2^2$, without independence between experts. This connects mean-estimation precision to the population center and target directions used by GGLR.

The population radius is also stable under errors in the means. Let $R^* = (E^{-1} \sum_e \|\boldsymbol{\mu}_e - \mathbf{a}^*\|_2^2)^{1/2}$. The reverse-triangle inequality for the centered matrix and contraction under centering give

$$|R - R^*| \le \left(\frac{1}{E} \sum_e \|\boldsymbol{\delta}_e - (\mathbf{a} - \mathbf{a}^*)\|_2^2 \right)^{1/2} \le \left(\frac{1}{E} \sum_e \|\boldsymbol{\delta}_e\|_2^2 \right)^{1/2}. \tag{36}$$

Write $\rho = \max\{R, \epsilon_r\}$ and $\rho^* = \max\{R^*, \epsilon_r\}$. Comparing the same token in estimated and true coordinates yields

$$\left\| \frac{\mathbf{h} - \mathbf{a}}{\rho} - \frac{\mathbf{h} - \mathbf{a}^*}{\rho^*} \right\|_2 \le \frac{\|\mathbf{a} - \mathbf{a}^*\|_2}{\rho} + \frac{\|\mathbf{h} - \mathbf{a}^*\|_2 \, |\rho - \rho^*|}{\rho \rho^*}. \tag{37}$$

Since the clamp is 1-Lipschitz, $|\rho - \rho^*| \le |R - R^*|$. For a fixed learned row, the corresponding selection affinity error is at most $\|\mathbf{w}_e\|_2/4$ times the right-hand side of Equation 37. With fixed biases, a true-coordinate second–third score gap exceeding twice the largest such error preserves the selected Top-2 set. This is a conditional stability statement about estimating a given population reference, not a convergence result for the coupled training dynamics.

B.5 Algebraic Decomposition of GGLR

Let $t_0 = 0$ denote the state after first-batch initialization, and define the actual GGLR optimizer displacement $\mathbf{D}_s = \mathbf{W}_s^{\text{opt}} - \mathbf{W}_{s-1}$. Starting from $\mathbf{W}_0, \mathbf{T}_0$, successive target-direction matrix differences in Equation 9 telescope in real arithmetic. For $t \geq 0$,

$$\begin{aligned} \mathbf{W}_t &= \mathbf{W}_{t_0} + (1-\lambda) \sum_{s=t_0+1}^{t} \mathbf{D}_s + \lambda \sum_{s=t_0+1}^{t} (\mathbf{T}_s - \mathbf{T}_{s-1}) \\ &= \mathbf{W}_{t_0} + (1-\lambda) \sum_{s=t_0+1}^{t} \mathbf{D}_s + \lambda(\mathbf{T}_t - \mathbf{T}_{t_0}). \end{aligned} \tag{38}$$

For $0 \leq \lambda < 1$, define $\mathbf{U}_t = (\mathbf{W}_{t_0} - \lambda \mathbf{T}_{t_0})/(1-\lambda) + \sum_{s=t_0+1}^{t} \mathbf{D}_s$. Then

$$\mathbf{W}_t = (1-\lambda)\mathbf{U}_t + \lambda \mathbf{T}_t. \tag{39}$$

The displacements $\mathbf{D}_s$ follow the actual GGLR trajectory, including its earlier geometry-guided updates. Thus $\mathbf{U}_t$ is an algebraic reparameterization of that same trajectory. Its inner-product contribution and the target-direction matrix's contribution add before sigmoid; selection bias enters afterward. For any fixed target-direction matrix $\mathbf{T}$ and desired matrix $\mathbf{W}^\star$, the choice $\mathbf{U}^\star = (\mathbf{W}^\star - \lambda \mathbf{T})/(1-\lambda)$ satisfies $(1-\lambda)\mathbf{U}^\star + \lambda \mathbf{T} = \mathbf{W}^\star$. The decomposition preserves the algebraic set of representable router matrices. GGLR guides parameter changes through target-direction matrix displacements while retaining this algebraic freedom; the actual optimizer determines which weights are reached.

The implementation evaluates the GGLR parameter update in 32-bit floating-point arithmetic (FP32) and casts its result to the 16-bit Brain Floating Point format (BF16) for router storage. Let $\boldsymbol{\rho}_s$ be the difference between each stored update and its real-arithmetic expression. With the stored first-batch initialization as the starting point, the trajectory obeys

$$\mathbf{W}_t = \mathbf{W}_{t_0} + (1-\lambda) \sum_{s=t_0+1}^{t} \mathbf{D}_s + \lambda(\mathbf{T}_t - \mathbf{T}_{t_0}) + \sum_{s=t_0+1}^{t} \boldsymbol{\rho}_s. \tag{40}$$

The residual sum includes arithmetic and storage rounding. Equation 39 describes the real-arithmetic rule, while empirical parameter and target-direction analyses use stored values.

C Reproducibility Details

C.1 Complete Training Procedure

Algorithm 1 gives the GGLR optimizer interval. Before the first interval, the trainer saves the random-number-generator (RNG) states, routes the first real microbatch without gradients to initialize EMA population means, initializes router rows from the resulting target directions, then restores the RNG states and replays that microbatch. Within each interval, it accumulates assignments over all gradient-accumulation microbatches, all-reduces their raw-input sums and counts, and updates each nonempty expert's EMA mean once; empty experts remain unchanged. It clips task gradients, then applies the task optimizer, selection-bias update, direct router update, and learning-rate scheduler only when the optimizer step is applied. If the optimizer step is skipped, the EMA update is retained while the bias and direct router updates are skipped.

C.2 Geometry of Linear Scores

Consider nearest-neighbor selection for a query $\mathbf{x} \in \mathbb{R}^d$ and references $\{\mathbf{z}_j\}_{j=1}^n$. For any distance D and strictly decreasing function f, the scores $s_j(\mathbf{x}) = f(D(\mathbf{x}, \mathbf{z}_j))$ satisfy $\arg\max_j s_j(\mathbf{x}) = \arg\min_j D(\mathbf{x}, \mathbf{z}_j)$. This statement gives only ranking equivalence, including tied optima: it neither equates score values with distances nor says that an arbitrary distance has a linear realization.

For squared Euclidean distance,

$$\|\mathbf{x} - \mathbf{z}_j\|_2^2 = \|\mathbf{x}\|_2^2 - 2\mathbf{z}_j^\mathsf{T}\mathbf{x} + \|\mathbf{z}_j\|_2^2. \tag{41}$$

The affine score with row $\boldsymbol{\omega}_j = 2\mathbf{z}_j$ and intercept $\eta_j = -\|\mathbf{z}_j\|_2^2$ is therefore $s_j(\mathbf{x}) = \boldsymbol{\omega}_j^\mathsf{T}\mathbf{x} + \eta_j = \|\mathbf{x}\|_2^2 - D(\mathbf{x}, \mathbf{z}_j)$. Stacking these rows and intercepts exactly realizes the squared-Euclidean ranking. A purely linear map omits η_j and has this property only when all reference norms are equal. The distance-derived intercept is fixed by reference geometry; the separately updated bias in Equation 1 instead changes selection to regulate load.

For nonzero vectors, write $\widehat{\mathbf{x}} = \mathbf{x}/\|\mathbf{x}\|_2$ and $\widehat{\mathbf{z}}_j = \mathbf{z}_j/\|\mathbf{z}_j\|_2$. Cosine distance and similarity satisfy

$$D_{\cos}(\mathbf{x}, \mathbf{z}_j) = 1 - \widehat{\mathbf{z}}_j^\mathsf{T}\widehat{\mathbf{x}}, \qquad s_j^{\cos}(\mathbf{x}) = \widehat{\mathbf{z}}_j^\mathsf{T}\widehat{\mathbf{x}}. \tag{42}$$

Unit reference directions can thus serve as linear weight rows for cosine ranking. GGLR uses each expert's EMA population mean as a one-vector summary and its centered unit direction as a target direction. Replacing the complete assigned-input multiset with its mean discards within-population variation and does not reproduce a full nearest-neighbor reference set. The learned router rows retain task-dependent direction and scale beyond that summary.

C.3 Training and Evaluation Details

The configuration uses root mean square normalization (RMSNorm) and rotary position embeddings (RoPE) (Zhang & Sennrich, 2019; Su et al., 2023).

Table 7: Configuration of the controlled five-method comparison. All methods share the architecture and training settings. The GGLR rows specify the selected method's additional settings.

Setting	Value
Model / MoE layers	1.7 billion parameters / 16
Hidden width / expert intermediate width	1,024 / 1,024
Query heads / key-value heads	16 / 4
Head dimension	64
RMSNorm epsilon	10^{-6}
RoPE base	10,000
Configured maximum positions	32,768
Vocabulary size	129,280
Routed experts / selected per token	32 / 2
Training steps / sequence length	63,579 / 4,096
Effective global batch / training seed	384 sequences / 42
Hardware / parameter storage	Eight H100 graphics processing units / BF16
Per-device microbatch / accumulation	24 sequences / 2 microbatches
Optimizer groups	Muon and AdamW; router matrices use Muon
Peak learning rate / weight decay	0.000713 / 0.1
Muon momentum	0.95
Learning-rate warmup	1,000 steps, followed by a constant peak rate
Final learning-rate decay	Linear over the final 10%, to 10% of peak
GGLR EMA momentum	0.99
GGLR geometric-update coefficient	0.5
GGLR initialization	First-real-microbatch means and replay
GGLR selection input	Centered and scaled by population radius
GGLR gain input	Unit-normalized original hidden state
GGLR selection-bias step	0.001

All five methods use a decoder-only Transformer with a Mixtral-style block structure (Jiang et al., 2024). Each block applies RMSNorm before attention and before its MoE sublayer, with a residual connection around each sublayer. A final RMSNorm precedes the language-model output projection. RMSNorm uses learned gains without additive offsets (Zhang & Sennrich, 2019). Every block contains routed experts, with no dense-only blocks or shared experts.

Attention uses grouped-query attention (GQA), with four query heads sharing each key–value head pair. Queries and keys receive RoPE (Su et al., 2023). Attention is fully causal across the input

sequence, with no sliding-window restriction or query/key normalization. Table 7 lists the RoPE base and configured maximum position count separately from the 4,096-token training sequence length.

Each expert uses a Swish-gated linear unit (SwiGLU) feedforward network with separate gate and value projections. A sigmoid linear unit activates the gate, whose elementwise product with the value branch is projected back to the hidden width. The expert intermediate width in Table 7 denotes the width of each branch. Attention, expert, and language-model output projections have no additive biases. Input token embeddings and the language-model output projection share weights.

GGLR performs the first-batch initialization and replay described in Appendix C.1 before step 1; restored checkpoints use their saved routing state. Each step's two accumulation microbatches contribute to one cross-rank EMA population mean update. EMA population means, population center/radius calculations, and geometric-update arithmetic use FP32; router matrices are stored in BF16. Selection inputs and unit gain inputs are cast to the hidden-state data type before the learned projection. Sigmoid and selected-score division use FP32. Both the shared-radius floor and the unit-input normalization floor are 10^{-8}. Assuming nonzero centered EMA population means, target directions divide these rows by their norms without a floor. Selected gains divide by their selected sum without the radius or bias. AdamW groups use momentum coefficients $(0.9, 0.95)$ and numerical-stability constant $\epsilon = 10^{-15}$; router matrices use Muon.

The corpus is the public Hugging Face dataset `https://huggingface.co/datasets/nvidia/Nemotron-CC-v2` (NVIDIA et al., 2025). A file ordering randomized with seed 20260720 supplies 176 complete Parquet files, totaling 201,645,377,529 bytes and 172,428,988 document rows. Downloads used mutable `main`-branch URLs, so their upstream revision was not recorded. As of September 18, 2026 (PDT), the official Nemotron-CC-v2 repository's last observable revision was `2669787c66d18601c0e91167fd520be6c1245865` (last modified July 7, 2026 PDT). The fixed training and held-out shard manifests record the input file order used for these runs. Offline preprocessing deterministically assigns each document to one of 64 groups using its source-file path, row index, document identifier, and seed 42. Groups are ordered with seed 42; the first group in that order is held out, and documents are shuffled within each group using group-derived seeds. The resulting manifests contain 2,646 training shards and 42 held-out shards. All runs use these fixed orders with runtime shuffling disabled.

Training and Nemotron-CC-v2 evaluation construct `LlamaTokenizerFast` from the `deepseek-ai/DeepSeek-V3-0324` tokenizer files, with automatic special-token insertion disabled. Document token streams are concatenated without added boundary tokens and cut into 4,096-token sequences, so a sequence can span documents. The held-out manifest defines the evaluation source; the reported measurements consume a fixed sample from it rather than the entire held-out corpus.

All learned baselines receive the MoE input after the model's multilayer perceptron (MLP) normalization, before any router-specific transformation. Auxiliary-loss balancing implements learned Top-2 dispatch with the SonicMoE library and adds the Switch load-balancing objective with coefficient 0.01 (Fedus et al., 2022). It adds the ST-MoE router z-loss (Zoph et al., 2022), the mean squared log-sum-exp of expert logits, with coefficient 0.001. Auxiliary terms are averaged across contributing MoE layers. This baseline uses neither persistent selection biases nor centroid updates.

For both LFB configurations, define the affinity $s_e(\mathbf{h}) = \sigma(\mathbf{w}_e^\top \mathbf{h})$ on that original MoE input, without router-specific unit normalization. Selection takes the two largest $s_e(\mathbf{h}) + b_e$. Loss-Free Balancing uses $g_e = s_e$ for selected experts, following the original formulation (Wang et al., 2024); LFB + gain normalization uses $g_e = s_e / \sum_{j \in \mathcal{A}(\mathbf{h})} s_j$. Both update selection-only biases once per optimizer step with sign-update coefficient 0.001 and add no balancing objective.

Online K-means uses centroids $\mathbf{k}_e$, their unit directions $\widehat{\mathbf{k}}_e$, and the affinity $s_e(\mathbf{h}) = \widehat{\mathbf{k}}_e^\top \widehat{\mathbf{h}}$, selects the largest $s_e + b_e$, and combines the selected expert outputs with $g_e = \exp(s_e) / \sum_{j \in \mathcal{A}(\mathbf{h})} \exp(s_j)$. Its stored centroids are raw assigned-input means updated with momentum 0.99 after each training microbatch, using globally aggregated sums and counts. Bias updates occur once per optimizer step with coefficient 0.001. Its selector has no trainable router rows. GGLR instead aggregates all accumulation microbatches before one EMA population mean update per optimizer interval and uses the learned selection and gain inputs in Algorithm 1.

The evaluation protocol uses each checkpoint at training step 63,579 with eight ranks, each processing 48 sequences per forward batch for 64 batches, giving 100,638,720 shifted next-token targets. Gradients and routing-state updates are disabled. Global load and source association aggregate actual assignments over this sample. Window measures use complete 65,536-token windows and discard incomplete terminal windows. The raw-input geometry probe samples 8,192 positions without replacement from the first three global validation sequences, using seed 20260814. These positions are fixed across methods and layers; each model supplies its own raw hidden states and actual Top-2 memberships. Means are computed only for experts with assigned samples. Probe sizes are global sample counts, not per-batch caps. All five evaluations follow this protocol.

Each public model uses its native tokenizer and its own 8,192-row input sample. Coordinates and token identities are not shared across models. Statistics of evaluation population means retain the first two recorded expert indices per token from native Top-6 selection in DeepSeek and Top-8 selection in MiniMax. They cover all 43 DeepSeek MoE layers and all 62 MiniMax MoE layers. DeepSeek-V4-Flash layers 0–2 select by fixed token-identifier lookup and are excluded from learned-row statistics; layers 3–42 use learned rows. MiniMax-M2.7 uses learned selector rows in all 62 layers. Both models have 256 routed experts.

The public checkpoints are `deepseek-ai/DeepSeek-V4-Flash`, revision `60d8d70770c6776ff598c94bb586a859a38244f1`, and `MiniMaxAI/MiniMax-M2.7`, revision `d494266a4affc0d2995ba1fa35c8481cbd84294b`. Table 6 reports their layer-averaged input and learned-row statistics.

C.4 Final-Layer Visualization and Clustering Measures

Figure 4 uses the saved raw router inputs at layer index 15 for all five controlled models. Figure 6 uses layer 42 for DeepSeek-V4-Flash and layer 61 for MiniMax-M2.7. These are their respective final MoE layers, with input dimensions 1,024, 4,096, and 3,072. Each model contributes all 8,192 saved vectors. The five controlled models share held-out token positions; the public captures use a separate Nemotron-CC-v2 sample with each model's native tokenizer. Expert labels are local to each model.

For this clustering analysis, each token is labeled by the first column of its recorded selected-expert array. This defines disjoint groups without duplicating token vectors. It is distinct from the overlapping evaluation population means used in Table 5 and Table 6. The label denotes a recorded selected expert, not uniformly the highest-scoring expert: MiniMax's native Top-8 operation does not sort its selected indices. All 32 labels occur in the controlled samples; 153 and 115 of the 256 expert labels occur in the DeepSeek and MiniMax samples, respectively.

For Figures 4 and 6, expert labels are ranked separately within each panel by decreasing plotted token count, with original expert identifier breaking ties. The same fixed categorical palette assigns a color to each rank in every panel. Ranking changes only the displayed label and color, not the grouping, coordinates, or DB values; it describes the single-label visualization rather than full native Top-K traffic.

Let G be the number of nonempty visualization groups, $\overline{\mathbf{h}}_i^{\text{vis}}$ the sample mean of group i, and s_i the mean Euclidean distance of its vectors to that mean. The Davies–Bouldin (DB) index is

$$\mathrm{DB} = \frac{1}{G}\sum_{i=1}^{G}\max_{j\neq i}\frac{s_i + s_j}{\|\overline{\mathbf{h}}_i^{\text{vis}} - \overline{\mathbf{h}}_j^{\text{vis}}\|_2}. \tag{43}$$

DB uses the original FP32 vectors with Euclidean distance, without additional centering, normalization, or dimensionality reduction. Equation 43 gives the definition; distances are Euclidean, not squared distances. We use scikit-learn's Davies–Bouldin implementation[1] on the full 8,192-vector sample; lower values indicate a smaller dispersion-to-separation ratio. This final-layer, single-label statistic is separate from layer-averaged and overlapping-assignment measurements.

The visualization applies randomized principal component analysis (PCA) to 50 components without whitening, followed by t-SNE with Euclidean distance, perplexity 100, PCA initialization, au-

[1] `https://scikit-learn.org/1.8/modules/clustering.html#davies-bouldin-index`

tomatic learning rate, and 1,000 iterations. The seed is 20260814. Projection is fitted independently for each model without expert labels; labels color the resulting points only. All saved points are retained. The displayed panels reuse these fitted coordinates, and the annotated DB values are computed before projection. Public-model measurements describe local organization under their own samples, dimensions, and routing conventions.

C.5 WIKITEXT-2 EVALUATION

We use the complete test split of `Salesforce/wikitext` (Merity et al., 2016), configuration `wikitext-2-raw-v1`.[2] All 4,358 rows, including empty rows, are joined in dataset order with two newline characters and tokenized as one text. We use the same explicit constructor as training, `LlamaTokenizerFast.from_pretrained`, on the DeepSeek-V3-0324 files. Under the evaluation environment's Transformers 5.12.1, the resulting tokenizer is `LlamaTokenizer` with Metaspace pre-tokenization. Loading the same files through `AutoTokenizer` instead produces ByteLevel pre-tokenization and different token identifiers. For WikiText-2, `add_special_tokens=True` inserts no beginning-of-sequence token, and the text produces 318,524 tokens. Reported perplexities use this training-aligned tokenization throughout.

The evaluation processes 155 windows with maximum length 4,096 and stride 2,048. Previously scored targets are masked in overlapping windows. The first window scores 4,095 targets, and the last scores 1,084 targets from its 3,132 input tokens. Every token after the first is scored exactly once, for 318,523 next-token targets in total. We sum the original fused language-model negative log-likelihood over these targets, divide by their total count, and exponentiate once. Auxiliary terms are excluded, and the final short window is retained. All five runs restore their complete model state from step-63,579 checkpoints, retaining the original BF16 parameter storage, with no parameter updates and unchanged persistent routing buffers throughout evaluation.

[2] `https://huggingface.co/datasets/Salesforce/wikitext`