# BRANCH-MOE: BALANCE-AWARE TREE ROUTING FOR LARGE EMBEDDING MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Mixture-of-experts (MoE) layers increase model capacity without a proportional increase in per-example computation. However, conventional flat routers can yield imbalanced expert utilization and treat experts as an unstructured collection, whose indices carry no topological meaning. We introduce **BRANCH-MoE**, a routing architecture that places $E$ experts at the leaves of a binary decision tree of depth $\log_2 E$. At each internal node the branching probability is centered on the arrival-weighted mean score of the traffic reaching that node. This mean is estimated using an exponential moving average, which promotes utilization of both child subtrees without an auxiliary load-balancing loss. We show that this moving-average estimate admits an explicit noise–lag trade-off. We prove that for linear node maps and log-concave arrival distributions, this mechanism prevents routing-mass collapse. We further establish that, under a frozen router, an expert's execution frequency controls its stochastic-gradient convergence rate, and that confident decisions near the root bound cross-device communication when experts are assigned to devices by tree prefix. We evaluate BRANCH-MoE against Switch softmax, DeepSeek-V3 dynamic-bias, Skywork logit-normalized, and deterministic hash routing on Criteo click-through-rate prediction, Forest Covertype, HIGGS, and YearPredictionMSD, using $E = 16$, top-4 routing, and five random seeds. BRANCH-MoE matches the predictive performance of the strongest baselines, attaining the lowest mean held-out loss on Forest Covertype and showing no statistically significant difference from the best-performing methods on Criteo, HIGGS, or YearPredictionMSD across random seeds. It also maintains balanced expert load without an auxiliary penalty. Unlike flat routers, BRANCH-MoE induces input-conditional co-routing locality: for each input token, the experts selected together tend to lie closer to one another in the routing tree. On all three UCI benchmarks, linear node maps yield a normalized tree distance of $0.710$–$0.718$ between co-selected experts, compared with $0.817$ for random expert pairs—a reduction of 12–13%. Every flat-routing baseline remains within $3.5\%$ of the random-pair distance. Together, these results show that hierarchical routing can preserve task quality and balanced utilization while inducing a topology that supports localized expert co-activation and reduced communication.

## 1 INTRODUCTION

Mixture-of-experts (MoE; Jacobs et al., 1991) layers increase parameter capacity while activating only a small subset of expert networks for each token (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022). A standard token-choice layer uses a flat $E$-way router followed by top-$K$ selection. This design has enabled very large sparse models, but it leaves several properties to losses, constraints, or systems heuristics rather than building them into the routing geometry.

**Limitations of flat routing.** First, a flat softmax or top-$K$ gate provides no distribution-free positive lower bound on the mass assigned to every expert: small initial routing advantages can produce highly unequal update frequencies, and an underused expert, receiving little data, can improve slowly (Eigen et al., 2013; Shazeer et al., 2017). Auxiliary load-balancing losses, capacity constraints, balanced assignment, and expert-choice routing mitigate this problem (Shazeer et al., 2017; Fedus et al., 2022; Lewis et al., 2021; Zhou et al., 2022), but the protection does not follow from the geometry of a flat gate. Second, a flat expert label is an unstructured categorical index that does not encode which



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.

experts share a device, node, rack, or other hardware cut, so the router offers no prefix locality for a distributed implementation to exploit. Third, preventing early routing collapse requires an auxiliary load-balancing penalty that competes with the task objective or an external bias controller (Wang et al., 2024; Liu et al., 2024). We do not claim that every conventional MoE collapses or incurs the same communication cost, only that these guarantees do not arise from flat routing alone.

Our proposal, BRANCH-MoE, changes the routing architecture itself. Its novelty is the combination of three coupled mechanisms within one routing tree: (i) a depth-$D$ binary routing tree for $E = 2^D$ experts; (ii) EMA score anchors and induced offsets that center every routing decision on the tokens arriving at its node; and (iii) a routing-mask cascade whose prefixes can serve as hierarchical expert or device addresses. The cascade determines the incoming mask used to update each node's anchor, the anchor determines the routing offset, and the final mask selects the experts each token is dispatched to, with the tree path giving each leaf a structured binary address rather than an arbitrary flat label.

**Our contributions.** We summarize our contributions as follows:

1. **Architecture.** BRANCH-MoE couples a soft mask cascade with one scalar EMA score anchor per node, updated with mask weights and a single shared decay $\alpha$ (Section 3). The anchor induces the inference-time bias $b_j = -\nu_j$, so no auxiliary balancing loss, separately learned bias, or $d$-dimensional centroid is needed; the full soft router evaluates $E - 1$ internal nodes in $O(\log_2 E)$ sequential depth, and inference uses the stored anchors as fixed biases without EMA updates.
2. **Theory** (Section 4). (1) The mask-weighted constant-decay EMA tracks each node's incoming mean score with an explicit noise–lag trade-off. (2) For linear routing under log-concave arrival distributions, anchoring prevents vanishing expected branch mass and yields a positive soft leaf-mass bound that degrades quantitatively with anchor error. (3) Under a frozen router and strongly convex conditional expert risks, the realized expert-execution frequency controls the convergence rate of expert SGD (the soft-mass bound does not itself guarantee top-$K$ execution). (4) We formalize the communication locality induced by tree routing: a token's experts are determined by its path, so co-routed tokens share path prefixes and confident upper-level decisions concentrate a token's routing mass within a shared-prefix subtree; under tree-aligned expert placement, the depth at which two paths diverge bounds whether their dispatches cross a hardware boundary, giving explicit bounds on cross-domain communication (Section 4.4).
3. **Experiments.** Against flat softmax (Switch), auxiliary-loss-free (DeepSeek-V3), logit-normalized (Skywork; Wei et al., 2024), and deterministic hash routing on four tabular benchmarks (Criteo click-through-rate prediction, Forest Covertype, HIGGS, and YearPredictionMSD; $E = 16$, top-4, five seeds), BRANCH-MoE matches the strongest baselines, with the lowest mean held-out loss on Forest Covertype and ties within seed noise on Criteo, HIGGS, and MSD. On all three UCI benchmarks, it is the *only* router whose co-routing statistics fall materially below the analytical random-pair baseline, reducing normalized tree distance to $0.710$–$0.718$ against $0.817$ ($12$–$13\%$ lower) and the leaf index gap by $18$–$23\%$, while every flat router stays within $3.5\%$ of random (Table 1).

## 2 Related Work

Prior work modifies the flat token-choice router (formalized in Section B) to improve sparsity, load balance, or routing stability (Zoph et al., 2022; Dai et al., 2022). Switch Transformer combines top-1 routing with an auxiliary load-balancing loss and fixed expert capacities (Fedus et al., 2022); DeepSeek-V3 adjusts expert selection through a load-driven per-expert bias while computing the output gate from the unbiased affinity (Wang et al., 2024; Liu et al., 2024). BRANCH-MoE also uses routing biases, but its bias $b_j = -\nu_j$, the negated EMA of the arrival-weighted mean score at node $j$, centers the node's decision boundary on its own incoming score distribution and is not a load-error feedback variable. Others impose balance through batch-coupled assignment (BASE and Sinkhorn-BASE; Lewis et al., 2021; Clark et al., 2022), expert-side capacity control (Expert Choice; Zhou et al., 2022), or a fixed routing rule (Hash Layers; Roller et al., 2021). BRANCH-MoE instead retains learned, representation-dependent token routing without solving a globally coupled assignment problem; it does not enforce exact per-batch load equality, but under the conditions of our analysis its anchored hierarchy provides a population-level nonstarvation guarantee. Like hierarchical mixtures of experts (Jordan & Jacobs, 1994; Shazeer et al., 2017), soft decision trees (Irsoy et al., 2012; Frosst & Hinton, 2017), tree-based sparse gates (Ibrahim et al., 2023), and fast feedforward

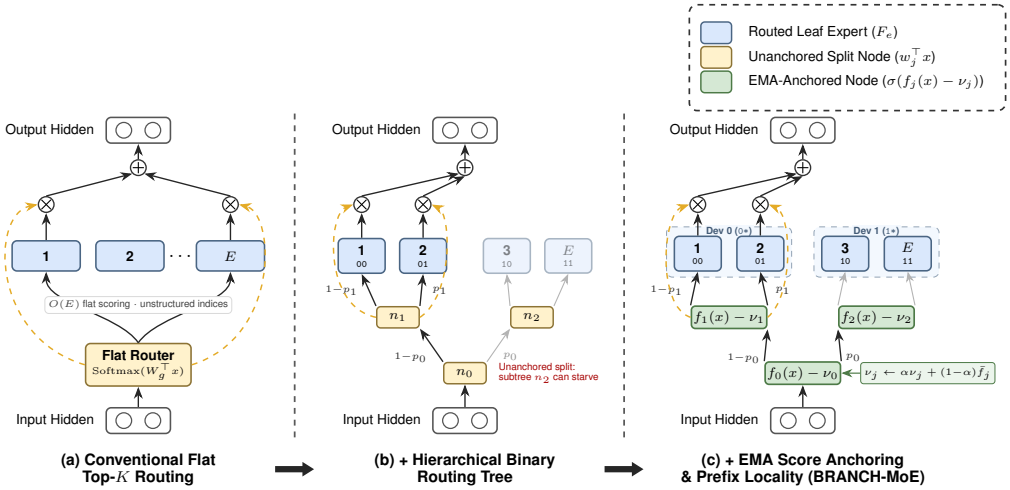



Figure 1: Architectural progression from conventional flat routing to BRANCH-MoE. **(a)** A flat Top-$K$ router scores all $E$ experts independently in $O(E)$ work and treats expert indices as unstructured labels. **(b)** A depth-$D$ binary routing tree with the experts as leaves gives $O(\log_2 E)$ sequential routing depth and a structured root-to-leaf binary address per leaf (`00`, `01`, ...), but unanchored internal splits can starve entire subtrees; the full soft distribution over the leaves still requires evaluating all $E - 1$ internal nodes. **(c)** BRANCH-MoE centers each internal split $j$ on its arrival-weighted EMA score anchor ($p_j = \sigma(f_j(x) - \nu_j)$), supporting non-starvation under the conditions of Section 4.2 without an auxiliary load-balancing loss and inducing prefix-aligned co-routing locality within hardware devices.

networks (Belcak & Wattenhofer, 2023), BRANCH-MoE routes through a tree, but its soft, fully differentiable branch decisions route tokens to trainable neural experts, its cascaded routing masks also update each node's score anchor, and its hierarchical expert addresses complement systems work on hierarchical communication and topology-aware routing (Rajbhandari et al., 2022; Hwang et al., 2023; He et al., 2022; Chen et al., 2022). Section B gives an extended discussion.

## 3 Architecture: BRANCH-MoE

Figure 1 illustrates the architecture, a self-balancing routing tree whose leaves are the experts; Table 2 (Section A) summarizes the notation. Let $\mathcal{T}$ be a complete binary tree of depth $D$ with internal-node set $\mathcal{J}$ and leaf set $\mathcal{E}$; an internal node $(d, j)$ has depth $d \in \{0, \ldots, D-1\}$ and index $j \in \{0, \ldots, 2^d - 1\}$ within that depth, and for any nonroot node or leaf $v$, $\mathrm{path}(v)$ denotes the internal nodes strictly preceding $v$ on its root-to-$v$ path. To avoid overloading the depth index, $d_{\text{in}}$ denotes the hidden dimension ($d$ in Table 2).

### 3.1 Phase 1: Adaptive Geometric Anchoring

Each internal node $(d, j)$ carries a trainable scalar-valued *node map* $f_{j,t}^{(d)} : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}$, evaluated at training step $t$, and a scalar *anchor* $\nu_{j,t}^{(d)} \in \mathbb{R}$. The parameters of $f_{j,t}^{(d)}$ are optimized by ordinary backpropagation through the routing probabilities, whereas $\nu_{j,t}^{(d)}$ is a non-gradient state variable maintained by the EMA Equation (5); there is no independently learned routing bias or additional scalar calibration variable, since the anchor plays that role and is refreshed whenever the node statistics change.

**Node maps: linear and nonlinear projections.** We instantiate $f_{j,t}^{(d)}$ either as a *linear* projection with a routing direction $W_{j,t}^{(d)} \in \mathbb{R}^{d_{\text{in}}}$ and no bias term, or as a *nonlinear* per-node two-layer MLP of

hidden width $h$:

$$f_{j,t}^{(d)}(x) = \big(W_{j,t}^{(d)}\big)^\top x, \tag{1}$$

$$f_{j,t}^{(d)}(x) = \big(v_{j,t}^{(d)}\big)^\top \varphi\Big(U_{j,t}^{(d)} x + c_{j,t}^{(d)}\Big), \qquad U_{j,t}^{(d)} \in \mathbb{R}^{h\times d_{\text{in}}}, \quad v_{j,t}^{(d)}, c_{j,t}^{(d)} \in \mathbb{R}^h, \tag{2}$$

with $\varphi$ applied elementwise (ReLU in our experiments); the matrix $W_t \in \mathbb{R}^{(E-1)\times d_{\text{in}}}$ stacks the $\big(W_{j,t}^{(d)}\big)^\top$ as its rows. Every node owns a private MLP whose output layer carries no bias, since anchoring removes any additive constant; taking $\varphi = \mathrm{id}$ and $h = 1$ recovers Equation (1) up to reparameterization, so linear routing is the degenerate case of nonlinear routing. The added cost is $O(hd_{\text{in}})$ parameters per node with $h$ small ($h \in \{2, 4, 8\}$ here), and the router can still be implemented as a fused batched computation.

**Arrival-weighted anchoring.** The anchor statistics and routing masks are constructed jointly in a root-to-leaf pass. At training step $t$, let $m_{j,t}^{(d)}(x)$ denote the probability that token $x$ reaches node $(d, j)$, and collect the depth-$d$ masks into $m_t^{(d)}(x) := \big(m_{0,t}^{(d)}(x), \ldots, m_{2^d-1,t}^{(d)}(x)\big)^\top \in \mathbb{R}^{2^d}$ (abbreviated $m^{(d)}(x)$ when the training index is clear); every token reaches the root with unit mass, so $m_{0,t}^{(0)}(x) = 1$. Given the depth-$d$ masks and a batch $X_t = (x_1, \ldots, x_B)$, the normalized *arrival weight* of token $x_b$ at node $(d, j)$, the batch *anchor statistic* (the arrival-weighted mean of the node scores), and the persistent single-decay EMA anchor are

$$\omega_{j,t}^{(d)}(x_b) := \frac{m_{j,t}^{(d)}(x_b)}{\sum_{b'=1}^{B} m_{j,t}^{(d)}(x_{b'})}, \tag{3}$$

$$\widehat{\nu}_{j,t}^{(d)} := \sum_{b=1}^{B} \omega_{j,t}^{(d)}(x_b)\, f_{j,t}^{(d)}(x_b), \tag{4}$$

$$\nu_{j,t+1}^{(d)} = \alpha_t \nu_{j,t}^{(d)} + (1-\alpha_t)\widehat{\nu}_{j,t}^{(d)}, \qquad \alpha_t = \min(\alpha, \tfrac{t-1}{t}), \qquad t \geq 1, \tag{5}$$

where $\alpha \in (0, 1)$ is the architectural decay rate and the cap $(t-1)/t$ is a warmup correction that removes initialization bias from the first updates. Every token thus contributes to every node's anchor in proportion to the routing mass reaching that node, so the statistic at $(d, j)$ is dominated by the token population selected by its ancestors; at the root, all arrival weights equal $1/B$ and $\widehat{\nu}_{0,t}^{(0)}$ is the ordinary batch mean of the root scores. Following the batch-normalization convention, the active anchor differs between training and inference; the centered node logit and its soft right-branch probability are

$$a_{j,t}^{(d)} := \begin{cases} \widehat{\nu}_{j,t}^{(d)}, & \text{during training,} \\ \nu_{j,t}^{(d)}, & \text{during inference,} \end{cases} \tag{6}$$

$$z_{j,t}^{(d)}(x) = f_{j,t}^{(d)}(x) - a_{j,t}^{(d)}, \qquad p_{j,t}^{(d)}(x) = \sigma\Big(z_{j,t}^{(d)}(x)\Big), \tag{7}$$

with $\sigma(\cdot)$ the sigmoid function. The live batch statistic makes training-time centering responsive to changes in the upstream representations, while the stored EMA makes inference deterministic and independent of batch composition.

**Hierarchical mask propagation.** The incoming masks for depth $d+1$ follow from the branch probabilities at depth $d$:

$$\begin{aligned} m_{2j,t}^{(d+1)}(x) &= m_{j,t}^{(d)}(x)\big(1 - p_{j,t}^{(d)}(x)\big), \\ m_{2j+1,t}^{(d+1)}(x) &= m_{j,t}^{(d)}(x) p_{j,t}^{(d)}(x), \qquad j = 0, \ldots, 2^d - 1. \end{aligned} \tag{8}$$

Consequently, $m_{j,t}^{(d)}(x)$ is the product of the branch probabilities along the unique root-to-node path ending at $(d, j)$. For finite logits, every sigmoid probability lies strictly between zero and one, so $m_{j,t}^{(d)}(x) \in (0, 1]$, with equality only at the root: in exact arithmetic, soft routing sends every token to every node with strictly positive mass rather than assigning it to a single subtree. The cascade also preserves total mass, $\sum_{j=0}^{2^d-1} m_{j,t}^{(d)}(x) = 1$ for every depth $d$, by induction from $m_{0,t}^{(0)}(x) = 1$, as

the two children of $(d, j)$ receive masses summing to $m_{j,t}^{(d)}(x)$. Section C.1 gives the batch form of Equation (8) and a numerical safeguard against underflow of deep masks in reduced precision.

**What is anchored, and in which space.** Substituting Equation (4) into the training branch of Equation (7) gives

$$\sum_{b=1}^{B} \omega_{j,t}^{(d)}(x_b)\, z_{j,t}^{(d)}(x_b) = 0. \tag{9}$$

This identity holds for any node map $f_{j,t}^{(d)}$, whether linear or nonlinear: each node centers its scores relative to the population that actually reaches it, so any routing imbalance must arise from variation of the scores around this population-specific mean rather than from a shared offset in the incoming representations. For the linear node map Equation (1), $\widehat{\nu}_{j,t}^{(d)} = \big(W_{j,t}^{(d)}\big)^{\top} \bar{x}_{j,t}^{(d)}$ with $\bar{x}_{j,t}^{(d)}$ the arrival-weighted centroid of the incoming tokens, so during training the separator is the hyperplane with normal $W_{j,t}^{(d)}$ passing through the centroid of its incoming population, and during inference the stored anchor induces the scalar bias $b_{j,t}^{(d)} = -\nu_{j,t}^{(d)}$. Scalar anchoring is therefore exactly equivalent to centroid anchoring for the live statistic in the linear case, while requiring only $O(1)$ persistent state per node rather than $O(\dim(x))$. Section C.2 gives the derivation and treats the inference statistic and nonlinear node maps, whose decision boundaries are level sets of the score; Section C.3 shows that anchoring also centers each node's parameter gradients on the arrival-weighted mean gradient of its population.

### 3.2 Phase 2: expert execution

Phase 1 produces the leaf-mask matrix $M_{\text{experts}} = M^{(D)} \in \mathbb{R}^{B \times E}$, with $[M^{(D)}]_{b,e} = m_e^{(D)}(x_b)$. Each leaf probability is the product of the $D$ branch probabilities along its root-to-leaf path, and every internal node splits its incoming mass between its two children with probabilities $p$ and $1 - p$, so the router outputs a normalized probability distribution over the $E$ leaf experts rather than unnormalized expert scores. It returns this distribution and the indices of its $K$ largest entries, leaving top-$K$ sparsification to the expert layer: the same routing distribution thus supports both dense and sparse implementations, and routing statistics and intended expert loads can be measured before capacity constraints alter the realized assignments. Let $\mathcal{S}_K(x) \subseteq \mathcal{E}$ index the $K$ largest entries of $m^{(D)}(x)$, with retained mass $R_K(x) := \sum_{e' \in \mathcal{S}_K(x)} m_{e'}^{(D)}(x)$. The gate applied to expert $e$ and the BRANCH-MoE output, the gated combination of the selected experts, are

$$\widetilde{m}_e(x) = \frac{m_e^{(D)}(x)\, \mathbf{1}\{e \in \mathcal{S}_K(x)\}}{R_K(x)^{\eta}}, \qquad \eta \in \{0, 1\}, \tag{10}$$

$$\widehat{y}(x) = \sum_{e \in \mathcal{S}_K(x)} \widetilde{m}_e(x)\, F_e(x). \tag{11}$$

With $\eta = 1$, each token's retained gates are renormalized, after top-$K$ masking, to sum to one; with $\eta = 0$, they retain their original path probabilities. We use $\eta = \mathbf{1}\{K > 1\}$ by default. The sparse implementation evaluates only the $K$ selected experts, a $\Theta(K/E)$ fraction of the expert computation of dense evaluation when dispatch overhead is ignored; Section C.4 discusses both choices.

## 4 Theoretical Analysis

We analyze four components of BRANCH-MoE: tracking of the arrival-weighted score anchors, prevention of routing-mass collapse, optimization of sparsely selected experts, and communication locality. Formal statements for the first and third are deferred to Section D, and all proofs to Section E.

### 4.1 Arrival-weighted population and anchor tracking

Fix a node $(d, j)$ and temporarily freeze the router and the node score. If the batch statistic equals a deterministic target plus martingale-difference noise with conditional second moment at most $s^2$ (Assumption D.1), then in the constant-decay phase of Equation (5) the anchor's initialization error decays geometrically as $\alpha^t$ and its asymptotic mean-square fluctuation is at most $(1 - \alpha)s^2/(1 + \alpha)$

(Theorem D.2); a target drifting by at most $v$ per update adds a lag term $v(1-\alpha^t)/(1-\alpha)$ to the bias part of the mean-square error bound (Proposition D.3). A larger $\alpha$ thus suppresses sampling noise, whereas a smaller $\alpha$ reduces lag. Under soft routing every node receives positive mass in exact arithmetic, but deep-node arrival weights can be highly concentrated, increasing $s^2$ and the risk of numerical underflow.

### 4.2 Score anchoring and routing-mass anti-collapse

Fix the soft router, including its active anchors, and a node $(d, j)$, and suppress the fixed step index. Its expected incoming mass is $q_j^{(d)} := \mathbb{E}[m_j^{(d)}(X)] > 0$. For every measurable set $\mathcal{A}$ in the input space, define the arrival-weighted law by $\mathcal{D}_j^{(d)}(\mathcal{A}) := \mathbb{E}[m_j^{(d)}(X)\mathbf{1}\{X \in \mathcal{A}\}]/q_j^{(d)}$. Let $\nu_j^{(d),\star} := \mathbb{E}_{\mathcal{D}_j^{(d)}}[f_j^{(d)}(X)]$ be the population score anchor and let $\delta_j^{(d)} := a_j^{(d)} - \nu_j^{(d),\star}$ denote the active-anchor error. The relative masses sent to the right and left children are, respectively, $\pi_{j,+}^{(d)} := \mathbb{E}_{\mathcal{D}_j^{(d)}}[\sigma(f_j^{(d)}(X) - a_j^{(d)})]$ and $\pi_{j,-}^{(d)} := \mathbb{E}_{\mathcal{D}_j^{(d)}}[\sigma(a_j^{(d)} - f_j^{(d)}(X))]$.

**Lemma 4.1** (Log-concavity under soft linear routing)**.** *Suppose the input distribution has a log-concave density on its affine support and every ancestor of node $(d, j)$ uses an affine score. Then the arrival-weighted law $\mathcal{D}_j^{(d)}$ is log-concave.*

**Corollary 4.2** (Linear routing under a log-concave arrival law)**.** *Suppose $\mathcal{D}_j^{(d)}$ is log-concave on its affine support and*

$$f_j^{(d)}(x) = \left(W_j^{(d)}\right)^\top x \tag{12}$$

*with $W_j^{(d)}$ nonzero on the linear span of that support. At the exact population anchor,*

$$\min\{\pi_{j,+}^{(d)}, \pi_{j,-}^{(d)}\} \geq \frac{1}{2\mathrm{e}}. \tag{13}$$

*If the centered projected score has density bounded by $A_j^{(d)}$, then for an arbitrary active anchor,*

$$\min\{\pi_{j,+}^{(d)}, \pi_{j,-}^{(d)}\} \geq \frac{1}{2}\left(\frac{1}{\mathrm{e}} - A_j^{(d)}|\delta_j^{(d)}|\right)_+. \tag{14}$$

For a fixed soft router, let

$$q_e := \mathbb{E}[m_e^{(D)}(X)] \tag{15}$$

be the expected routing mass of leaf $e$. If the density condition in Corollary 4.2 holds at every node on the path to $e$, define

$$\rho_j^{(d)} := \frac{1}{2}\left(\frac{1}{\mathrm{e}} - A_j^{(d)}|\delta_j^{(d)}|\right)_+, \tag{16}$$

then repeated conditioning along the path to $e$ gives

$$q_e \geq \underline{q}_e := \prod_{(d,j)\in \mathrm{path}(e)} \rho_j^{(d)}. \tag{17}$$

In the linear log-concave case with exact population anchors,

$$\underline{q}_e \geq (2\mathrm{e})^{-D} \qquad \text{for every leaf } e. \tag{18}$$

These bounds do not, by themselves, imply that every leaf is retained by deterministic top-$K$ truncation; this distinction is addressed next.

### 4.3 Expert optimization under sparse top-$K$ routing

Freeze the router, the top-$K$ rule, and any capacity policy, and let $p_e$ be the probability that expert $e$ executes an example (Equation (45)), i.e., the actual execution probability after top-$K$ selection rather than the expected leaf mass $q_e$ in Equation (15). For i.i.d. examples executed with probability $p_e$ independently of the history, a $\mu_e$-strongly convex and $L_e$-smooth conditional risk with minimizer

$\theta_e^\star$, and conditionally unbiased stochastic gradients with variance at most $\sigma_e^2$ (Assumption D.4), SGD on the parameters $\theta_e$ of expert $F_e$ with step size $0 < \gamma_e \le 1/L_e$ satisfies $\mathbb{E}\|\theta_{e,T} - \theta_e^\star\|^2 \le \exp(-p_e\gamma_e\mu_e T)\|\theta_{e,0} - \theta_e^\star\|^2 + \gamma_e\sigma_e^2/\mu_e$ after $T$ global examples (Theorem D.5). The realized execution frequency is thus the optimization bottleneck for sparsely activated experts: if $p_e \ge p_{\min} > 0$ for all experts, a larger $p_{\min}$ yields faster worst-expert convergence in global training steps. Although the soft-mass bound Equation (17) does not formally imply a positive top-$K$ execution probability, it removes an important mechanism of expert starvation and should make realized execution frequencies larger and more stable than under unconstrained learned routing (Section D.2 discusses this implication).

### 4.4 Communication locality from learned tree routing

We finally analyze communication at a hardware boundary whose domains are aligned with depth-$r$ routing subtrees; a token is transmitted once to every contacted domain and may then be replicated locally. The argument depends only on the centered node scores, so it applies to both linear and nonlinear node maps. Fix a token $x$, follow the more probable child for $r$ levels, and let $u_0, u_1, \dots, u_r$ be the resulting path, where $u_i = (i, j_i)$ and $u_0$ is the root. Let $\mathcal{L}(u_r)$ be the leaves below $u_r$, assumed to contain at least $K$ experts, and let $\mathcal{S}_K(x)$ be the top-$K$ leaf set. Under prefix-aligned placement, let $N_r(x)$ denote the number of depth-$r$ communication domains containing at least one selected expert, and suppose a token representation contains $s_{\text{tok}}$ bytes and transmission across this boundary costs $c_r$ per byte. The learned score margin at $u_i$, the leaf mass outside the prefix subtree, the smallest retained probability of $\mathcal{S}_K(x)$, and the communication cost are

$$\gamma_i(x) := \left|z^{(i)}_{j_i,t}(x)\right| = \left|f^{(i)}_{j_i,t}(x) - a^{(i)}_{j_i,t}\right|, \tag{19}$$

$$\varepsilon_r(x) := \sum_{e \notin \mathcal{L}(u_r)} m^{(D)}_{e,t}(x), \tag{20}$$

$$\tau_K(x) := \min_{e \in \mathcal{S}_K(x)} m^{(D)}_{e,t}(x), \tag{21}$$

$$\mathrm{Comm}_r(x) := s_{\text{tok}} c_r N_r(x). \tag{22}$$

At $u_i$, the preferred child receives probability $\sigma(\gamma_i(x))$, while the other child receives $\sigma(-\gamma_i(x)) \le e^{-\gamma_i(x)}$.

**Theorem 4.3** (Learned-margin communication locality)**.** *For every token $x$,*

$$\varepsilon_r(x) = 1 - \prod_{i=0}^{r-1} \sigma(\gamma_i(x)) \le \sum_{i=0}^{r-1} e^{-\gamma_i(x)}. \tag{23}$$

*Moreover,*

$$|\mathcal{S}_K(x) \setminus \mathcal{L}(u_r)| \le \left\lfloor \frac{\varepsilon_r(x)}{\tau_K(x)} \right\rfloor, \tag{24}$$

*and*

$$N_r(x) \le \min\left\{K, 1 + \left\lfloor \frac{\varepsilon_r(x)}{\tau_K(x)} \right\rfloor\right\}. \tag{25}$$

*In particular, if*

$$\sum_{i=0}^{r-1} e^{-\gamma_i(x)} < \tau_K(x), \tag{26}$$

*then all top-$K$ experts lie in $\mathcal{L}(u_r)$ and*

$$N_r(x) = 1, \qquad \mathrm{Comm}_r(x) = s_{\text{tok}} c_r. \tag{27}$$

If the first $r$ score margins satisfy $\gamma_i(x) \ge \gamma$, then

$$\varepsilon_r(x) \le r e^{-\gamma}. \tag{28}$$

Thus $\gamma > \log(r/\tau_K(x))$ is sufficient for one-domain dispatch at the depth-$r$ boundary.

Table 1: Main results ($E$=16, top-4, 5 seeds; mean $\pm$ sd). (a) Best held-out loss, each router at its best of four node maps (tag), selected on the reporting split; BRANCH-MoE also with the linear map covered by the theory. Bold: best arm. Hash ignores the node map (UCI: best of its four identical runs). (b) Normalized tree distance (random pairs 0.817; bold: most local on UCI) / load CV of the same arms. †Not read as locality (Section G.1). Per-map results: Section G.

| Method | Criteo (logloss) | Covertype (CE) | HIGGS (logloss) | MSD (MSE) |
|---|---|---|---|---|
| *(a) Best held-out loss ↓* | | | | |
| BRANCH-MoE (best map) | **0.44596**±0.00015 h4 | **0.2372**±0.0013 h8 | 0.5256±0.0014 h8 | 0.6767±0.0027 h8 |
| BRANCH-MoE (linear) | 0.44615±0.00025 | 0.2545±0.0041 | 0.5259±0.0008 | 0.6843±0.0035 |
| Skywork | 0.44603±0.00030 lin | 0.2406±0.0036 h8 | **0.5247**±0.0005 lin | 0.6757±0.0015 h2 |
| DeepSeek-V3 | 0.44605±0.00005 h8 | 0.2421±0.0033 h8 | 0.5264±0.0013 lin | **0.6746**±0.0043 h8 |
| Switch | 0.44623±0.00019 lin | 0.2406±0.0028 h8 | 0.5276±0.0003 h4 | 0.6766±0.0038 h8 |
| Hash | 0.44707±0.00028 | 0.4465±0.0011 | 0.5574±0.0007 | 0.6899±0.0024 |
| *(b) Tree distance ↓ / load CV ↓ of the same arms* | | | | |
| BRANCH-MoE (best map) | **0.757** / 0.268 | 0.742 / 0.112 | 0.736 / 0.231 | 0.749 / 0.219 |
| BRANCH-MoE (linear) | 0.813 / 0.115 | **0.718** / 0.093 | **0.710** / 0.227 | **0.718** / 0.177 |
| Skywork | 0.818 / 0.004 | 0.817 / 0.137 | 0.812 / 0.183 | 0.813 / 0.536 |
| DeepSeek-V3 | 0.819 / 0.004 | 0.813 / 0.001 | 0.816 / 0.002 | 0.813 / 0.003 |
| Switch | 0.821 / 0.003 | 0.819 / 0.141 | 0.814 / 0.104 | 0.800 / 0.278 |
| Hash | 0.817 / 0.002 | 0.817 / 0.003 | 0.817 / 0.002 | 0.817 / 0.004 |

**Interpretation.** An upper-level decision acts on an entire subtree: a large learned margin suppresses the total probability of every leaf in the nonpreferred branch, so after $r$ confident decisions the routing mass is concentrated in a subtree of $E/2^r$ experts; if that subtree is placed within one hardware domain, the lower levels refine the expert choice locally. This per-token result is compatible with the population-level anti-collapse result of Section 4.2: arrival-weighted anchoring prevents the token population as a whole from placing arbitrarily little routing mass on either child of an internal node, but does not require each token to split its mass evenly. A token may assign most of its probability to one branch while other tokens favor the opposite branch, so BRANCH-MoE can maintain broad expert utilization *across tokens* while producing concentrated, structured assignments *for each token*.

## 5 Experiments

We evaluate BRANCH-MoE against representative MoE routers on four benchmarks, testing (i) whether it is competitive on task quality without an auxiliary balancing loss and (ii) whether it induces the expert locality analyzed in Section 4.4. In short, quality is at parity with the strongest baselines on all four benchmarks, locality appears on the three UCI benchmarks but not on Criteo, and load stays balanced without an auxiliary loss.

### 5.1 Setup

We use Criteo (≈41.5M examples, 39 fields), Forest Covertype (581K, 54 features), HIGGS (600K, 28), and YearPredictionMSD (515K, 90). All MoE conditions share one block of $E = 16$ SwiGLU experts (top-4, capacity factor 1.25) between a minimal input stage and a linear head, and differ only in the gate. The baselines are softmax top-$k$ with an auxiliary loss (Switch), dynamic-bias routing (DeepSeek-V3), logit-normalized softmax with an auxiliary loss (Skywork), and a hash-based control (Hash). Node maps are linear or two-layer MLPs of width $h \in \{2, 4, 8\}$. Every condition has five seeds; comparisons are paired by seed on UCI and unpaired (Welch) on Criteo, where TPU execution is not bitwise reproducible. We report the normalized tree distance $L/D$ between co-selected experts ($L = \mathrm{bit_length}(i \oplus j)$, $D = \log_2 E$; random pairs score 0.817, lower is more local) and the load CV (std/mean of per-expert load). Details: Section F.

### 5.2 Results

**Quality.** BRANCH-MoE is at parity with the strongest baselines (Table 1a). On Criteo the learned routers span $1.9 \times 10^{-4}$ logloss with the linear map, about half the smallest resolvable difference; at

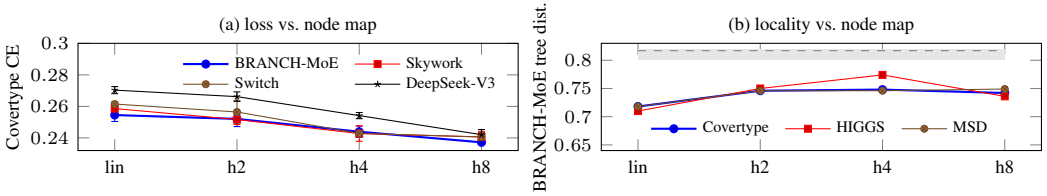



Figure 2: Node-map effect. (a) Covertype loss of the learned routers vs. node map, ±sd (Hash, at 0.447, is off-scale). (b) BRANCH-MoE's tree distance vs. node map on UCI; dashed: random pairs (0.817); shaded: flat routers at their best maps.

its best map ($h = 4$) BRANCH-MoE has the lowest mean, but its leads over Skywork ($p = 0.63$) and DeepSeek-V3 ($p = 0.26$) are within seed noise, and DeepSeek-V3 has the higher AUC. On Covertype BRANCH-MoE ($h = 8$) has the lowest mean cross-entropy (0.2372); its 0.0034 margin over Switch is resolved only by the paired test ($t_4 = -2.86$, critical value 2.776), the equal margin over Skywork by neither test, and the 0.0049 margin over DeepSeek-V3 by the paired test ($t_4 = -3.25$). BRANCH-MoE ranks second on HIGGS (0.0009 behind Skywork, within seed spread) and fourth on MSD, where the four learned routers span 0.0021 against seed sds of 0.0015–0.0043. Hash is the worst MoE arm everywhere.

**Locality.** On the three UCI benchmarks BRANCH-MoE is the only router materially below random (Table 1b). With the linear map its tree distance is 0.710–0.718, 12–13% below random (18–23% on index distance), and 0.736–0.749 at $h = 8$; every flat router stays within 3.5% of random (closest: Switch on MSD, 0.800). On Criteo BRANCH-MoE shows no locality: with the linear map its tree distance (0.813) is indistinguishable from random ($p = 0.48$), although an untrained tree scores 0.725. The lower values with MLP maps ($^\dagger$) also appear for the flat Skywork router (0.75–0.77), so we do not read them as locality. Criteo differs from UCI in input representation, batch size, and training length; we have not isolated which matters (Section G.1).

**Balance.** BRANCH-MoE uses no auxiliary loss or bias controller. On UCI its linear-map load CV (0.09–0.23) is comparable to that of the auxiliary-loss routers at their best maps (Switch 0.10–0.28, Skywork 0.14–0.54); on Covertype the only dead experts are Skywork's at $h = 2$ (6 of 80 slots). On Criteo its CV of 0.115 exceeds the flat routers' ($\leq 0.006$) but costs nothing measurable: it drops 0.08% of dispatches.

**Node maps.** The node map can matter more than the routing rule (Figure 2a). On Covertype, replacing the linear map by an $h = 8$ MLP lowers cross-entropy by 0.017–0.028 for every learned router (paired $t$ from $-7.99$ to $-21.44$), 3.5–5.8× the largest gap between routers at $h = 8$ (0.0049; 0.016 at linear), while Hash, which ignores the node map, moves by only 0.00013. The sign depends on the dataset: MSD agrees; on HIGGS the linear map is best for Skywork and DeepSeek-V3 and ties for BRANCH-MoE; on Criteo no router's loss changes significantly ($\leq 4.1 \times 10^{-4}$, $p > 0.1$) except Switch's, which fails with every MLP map (≈68% of dispatches dropped). The MLP also loosens BRANCH-MoE's structure: on UCI its tree distance rises by 0.024–0.031 (Figure 2b; same sign on all three datasets, not resolved on Covertype alone), and on Criteo its load CV rises from 0.115 to 0.21–0.38 ($p \leq 0.01$ at $h = 4$ and 8).

**Takeaway.** Balance without an auxiliary loss and locality on UCI are supported, and quality is at parity; locality on Criteo and whether MLP maps erode locality remain open (Section G).

**Conclusion and limitations.** BRANCH-MoE combines arrival-weighted score anchoring with hierarchical routing to encourage balanced population-level utilization and structured per-token expert selection without an auxiliary balancing loss. Our analysis characterizes anchor tracking, establishes soft-mass anti-collapse for linear routing under log-concave arrivals, relates realized expert-selection frequency to optimization progress, and bounds communication locality under prefix-aligned placement. The experiments demonstrate competitive task quality and structured locality across several benchmarks. The present study is limited to a single BRANCH-MoE layer and therefore does not capture interactions among routing decisions across multiple layers. Future work will extend both the theory and empirical evaluation to multilayer architectures and develop fused, topology-aware implementations for large-scale distributed settings.

### AI USE STATEMENT

In this work, we used generative AI tools to improve the manuscript's clarity, presentation, and writing, and to reorganize it in accordance with the conference's formatting requirements. We have not used generative AI tools for conducting experiments. All AI-assisted material was carefully reviewed and verified by the authors. We accept full responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

## REFERENCES

Peter Belcak and Roger Wattenhofer. Fast feedforward networks. *arXiv preprint arXiv:2308.14711*, 2023.

Chang Chen, Min Li, Zhihua Wu, Dianhai Yu, and Chao Yang. Ta-moe: Topology-aware large scale mixture-of-expert training. *Advances in Neural Information Processing Systems*, 35:22173–22186, 2022.

Aidan Clark, Diego de Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. Unified scaling laws for routed language models. In *International conference on machine learning*, pp. 4057–4086. PMLR, 2022.

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. Stablemoe: Stable routing strategy for mixture of experts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7085–7095, 2022.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013.

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.

Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

Jiaao He, Jidong Zhai, Tiago Antunes, Haojie Wang, Fuwen Luo, Shangfeng Shi, and Qin Li. Fastermoe: modeling and optimizing training of large-scale dynamic pre-trained models. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pp. 120–134, 2022.

Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, et al. Tutel: Adaptive mixture-of-experts at scale. *Proceedings of Machine Learning and Systems*, 5:269–287, 2023.

Shibal Ibrahim, Wenyu Chen, Hussein Hazimeh, Natalia Ponomareva, Zhe Zhao, and Rahul Mazumder. Comet: Learning cardinality constrained mixture of experts with trees and local search. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 832–844, 2023.

Ozan Irsoy, Olcay Taner Yıldız, and Ethem Alpaydın. Soft decision trees. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pp. 1819–1822. IEEE, 2012.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pp. 6265–6274. PMLR, 2021.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International conference on machine learning*, pp. 18332–18346. PMLR, 2022.

Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. Hash layers for large sparse models. *advances in neural information processing systems*, 34:17555–17566, 2021.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024.

Tianwen Wei, Bo Zhu, Liang Zhao, Cheng Cheng, Biye Li, Weiwei Lü, Peng Cheng, Jianhao Zhang, Xiaoyu Zhang, Liang Zeng, et al. Skywork-moe: A deep dive into training techniques for mixture-of-experts language models. *arXiv preprint arXiv:2406.06563*, 2024.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

# A NOTATION

Table 2: Principal notation.

| Symbol | Definition |
|---|---|
| $B, d$ | Number of resident tokens and hidden dimension |
| $D,\ E = 2^D$ | Routing-tree depth and number of leaf experts |
| $\mathcal{T}, \mathcal{J}, \mathcal{E}$ | Binary tree, internal nodes, and leaves |
| $W \in \mathbb{R}^{(2^D-1)\times d}$ | Matrix of learned node directions |
| $\nu_j,\ b_j = -\nu_j$ | Node score anchor and induced routing bias |
| $\bar{x}_j$ | Arrival-weighted centroid of the tokens reaching node $j$ |
| $P^{(d)}$ | Routing probabilities at tree depth $d$ |
| $M^{(d)} \in \mathbb{R}^{B\times 2^d}$ | Incoming routing masks at depth $d$ |
| $M_{\text{experts}} = M^{(D)}$ | Final expert routing mask |
| $X \in \mathbb{R}^{B\times d}$ | Input token representations |
| $\alpha \in (0, 1)$ | Single EMA decay used for anchor tracking |
| $\mathcal{S}_K$ | Top-$K$ expert set selected for a token |
| $F_e(\cdot;\theta_e), \theta_e$ | Expert $e$ and its trainable parameter vector |

# B BACKGROUND AND EXTENDED RELATED WORK

This appendix expands on Section 2. We first recall the flat token-choice router that serves as the reference template. Let $\{F_e(\cdot;\theta_e)\}_{e=1}^{E}$ be expert networks, where $\theta_e$ collects the trainable parameters of expert $e$, and let $W_g \in \mathbb{R}^{d\times E}$ be a flat gating matrix. A standard token-choice router forms the score vector $s = x^\top W_g$, whose component $s_f = x^\top (W_g)_{:,f}$ is the score of expert $f$, and computes

$$G(x) = \mathrm{softmax}(x^\top W_g). \tag{29}$$

After retaining the $K$ largest components, the routed output is

$$y(x) = \sum_{e=1}^{E} \mathrm{TopK}(G(x))_e F_e(x). \tag{30}$$

Prior work modifies different aspects of this template to improve sparsity, load balance, or routing stability. Switch Transformer specializes it to top-1 routing and combines an auxiliary load-balancing loss with fixed expert capacities (Fedus et al., 2022). DeepSeek-V3 instead adjusts expert selection through a dynamic per-expert bias: the bias increases for underloaded experts and decreases for overloaded experts, while the gate multiplying the expert output is computed from the original, unbiased affinity score (Liu et al., 2024). Its principal balancing mechanism is therefore auxiliary-loss-free, although it retains a small sequence-level balancing loss. BRANCH-MoE also uses routing biases, but for a different purpose. Its bias $b_j = -\nu_j$ is the negated EMA of the arrival-weighted mean score at node $j$, and anchors that node's decision boundary at the center of its own incoming score distribution; it is not a load-error feedback variable.

Other methods alter the assignment rule more fundamentally. BASE constructs a balanced batch assignment through a matching problem (Lewis et al., 2021), while Sinkhorn-BASE approximates such an assignment using entropy-regularized optimal transport and Sinkhorn iterations (Clark et al., 2022). Expert Choice reverses the usual assignment direction: each expert selects a fixed-size bucket of high-scoring tokens, controlling expert workloads while allowing different tokens to be processed by different numbers of experts (Zhou et al., 2022). Hash Layers remove the learned router and assign inputs to experts using a fixed input-dependent hash, thereby avoiding router parameters, balancing losses, and batchwise matching (Roller et al., 2021). These methods respectively impose balance through batch-coupled assignment, expert-side capacity control, or a fixed routing rule. BRANCH-MoE instead retains learned, representation-dependent token routing without solving a globally coupled assignment problem.

BRANCH-MoE is also related to hierarchical conditional-computation models. Hierarchical mixtures of experts organize routing decisions as a tree (Jordan & Jacobs, 1994). Frosst and Hinton's soft

decision tree uses learned sigmoid decisions and path probabilities to distill a neural network into a tree whose leaves store class distributions (Frosst & Hinton, 2017). BRANCH-MoE uses the hierarchy differently: its soft, fully differentiable branch decisions route tokens to trainable neural experts, and the same cascaded masks used for routing also determine the tokens used to update each node's score anchor.

Notably, BRANCH-MoE does not enforce exact per-batch load equality, as do assignment- constrained methods. Instead, under the conditions stated in our analysis, its anchored hierarchy provides a population-level nonstarvation guarantee while preserving adaptive sparse routing. The resulting tree paths also give experts hierarchical addresses, complementing systems work on token permutation and distributed communication (Hwang et al., 2023).

# C Additional Details of the BRANCH-MoE Router

This appendix collects implementation details and additional properties of the router of Section 3 that are referenced from the main text.

## C.1 Batch form of the mask cascade and numerical safeguard

For the batch $X_t = (x_1, \dots, x_B)$ of Section 3.1, define

$$[M_t^{(d)}]_{b,j} := m_{j,t}^{(d)}(x_b), \qquad [P_t^{(d)}]_{b,j} := p_{j,t}^{(d)}(x_b), \tag{31}$$

so that

$$M_t^{(d)}, P_t^{(d)} \in \mathbb{R}^{B\times 2^d}, \qquad M_t^{(0)} = \mathbf{1}_B.$$

The dimensionally explicit batch form of Equation (8) is

$$\begin{aligned} [M_t^{(d+1)}]_{:,2j} &= [M_t^{(d)}]_{:,j} \odot \big(1 - [P_t^{(d)}]_{:,j}\big), \\ [M_t^{(d+1)}]_{:,2j+1} &= [M_t^{(d)}]_{:,j} \odot [P_t^{(d)}]_{:,j}, \qquad j = 0, \dots, 2^d - 1. \end{aligned} \tag{32}$$

Equivalently, after applying the fixed column permutation that interleaves the two children of each parent,

$$M_t^{(d+1)} = \Big[ M_t^{(d)} \odot (1 - P_t^{(d)}) \,\Big\|\, M_t^{(d)} \odot P_t^{(d)} \Big]. \tag{33}$$

The denominator in Equation (3) is strictly positive since the probabilities are strictly positive. In reduced precision, however, a depth-$d$ mask is a product of $d$ sigmoid factors and may underflow to zero. The implementation therefore clamps the denominator away from zero and skips the corresponding EMA update if the pre-clamp arrival mass vanishes, leaving $\nu_{j,t}^{(d)}$ unchanged. This is a numerical safeguard rather than a modification of the routing rule.

## C.2 What is anchored: linear and nonlinear node maps

This appendix details the discussion following the score-centering identity Equation (9) in Section 3.1. For the linear node map

$$f_{j,t}^{(d)}(x) = \big(W_{j,t}^{(d)}\big)^\top x,$$

define the arrival-weighted centroid

$$\bar{x}_{j,t}^{(d)} := \sum_{b=1}^{B} \omega_{j,t}^{(d)}(x_b) x_b. \tag{34}$$

Linearity gives

$$\widehat{\nu}_{j,t}^{(d)} = \big(W_{j,t}^{(d)}\big)^\top \bar{x}_{j,t}^{(d)}, \tag{35}$$

and hence the training-time logit becomes

$$z_{j,t}^{(d)}(x) = \big(W_{j,t}^{(d)}\big)^\top \big(x - \bar{x}_{j,t}^{(d)}\big). \tag{36}$$

Thus, during training, the separator is the hyperplane with normal $W_{j,t}^{(d)}$ passing through the centroid of its incoming population. During inference,

$$z_{j,t}^{(d)}(x) = \left(W_{j,t}^{(d)}\right)^\top x - \nu_{j,t}^{(d)},$$

so the stored anchor induces the scalar bias

$$b_{j,t}^{(d)} = -\nu_{j,t}^{(d)}. \tag{37}$$

Scalar anchoring is therefore exactly equivalent to centroid anchoring for the live statistic in the linear case, while requiring only $O(1)$ persistent state per node rather than $O(\dim(x))$.[1]

For a nonlinear node map, the anchored quantity is the scalar node score rather than a point in token space. At training time, the decision boundary is the level set

$$\left\{x : f_{j,t}^{(d)}(x) = \widehat{\nu}_{j,t}^{(d)}\right\}.$$

This boundary may be curved, piecewise linear, or disconnected, depending on the node map, and need not pass through a centroid in the original token space. What remains exact is the score-centering identity Equation (9). Accordingly, geometric arguments that require halfspace sublevel sets apply directly to linear routing, or to routing that is linear after a shared feature map, but not to an arbitrary nonlinear node map.

### C.3 Gradient geometry

During training, the arrival weights in Equation (3) are detached when they are used to form the live anchor. Their forward values determine which tokens contribute to the node statistic, but gradients from the anchor are not propagated through the routing masks. The live anchor remains differentiable through the node scores themselves. Therefore, for the parameters $\theta_{j,t}^{(d)}$ of node $(d, j)$,

$$\nabla_{\theta_{j,t}^{(d)}} z_{j,t}^{(d)}(x_b) = \nabla_{\theta_{j,t}^{(d)}} f_{j,t}^{(d)}(x_b) - \sum_{b'=1}^{B} \omega_{j,t}^{(d)}(x_{b'}) \nabla_{\theta_{j,t}^{(d)}} f_{j,t}^{(d)}(x_{b'}). \tag{38}$$

Thus, each node receives parameter gradients centered by the arrival-weighted mean gradient of the population it serves. For the linear map,

$$\nabla_{W_{j,t}^{(d)}} z_{j,t}^{(d)}(x_b) = x_b - \bar{x}_{j,t}^{(d)}. \tag{39}$$

For a nonlinear map, the same centering occurs in the corresponding Jacobian features. Anchoring therefore removes the shared mean component that could otherwise dominate the router gradient when the incoming representations contain a large common offset.

### C.4 Gate normalization and sparse execution

For the gate Equation (10) and the output Equation (11) of Section 3.2, the following remarks apply. When $\eta = 1$, each token's retained gates are renormalized to sum to one. When $\eta = 0$, the gates retain their original path probabilities. Importantly, normalization is applied only after top-$K$ masking; unselected experts therefore have no effect on the relative weights of the selected experts.

Regarding the choice of $\eta$ in Equation (10), note that renormalization preserves the relative weights within $\mathcal{S}_K(x)$. For $K > 1$, this produces a convex combination of the selected expert outputs and stabilizes the scale of the mixture. For $K = 1$, however, renormalization would force the unique nonzero gate to equal one, discarding the router's only scalar measure of confidence in the selected leaf. We therefore use $\eta = \mathbf{1}\{K > 1\}$ as the default choice.

The sparse implementation evaluates only the $K$ selected experts, grouping tokens by expert and scattering the gated outputs back to their original positions. Ignoring dispatch overhead, this requires a $\Theta(K/E)$ fraction of the expert computation of dense evaluation.

[1] For the inference statistic, the equivalence is affected by the drift of $W_{j,t}^{(d)}$ over the EMA window: $\nu_{j,t}^{(d)}$ averages past values of $\left(W_{j,\tau}^{(d)}\right)^\top \bar{x}_{j,\tau}^{(d)}$ computed with potentially different router parameters.

# D Deferred Theoretical Results

This appendix states the results summarized in Section 4.1 and Section 4.3; their proofs are in Section E.

## D.1 Arrival-weighted population and anchor tracking

Fix a node $(d, j)$ and temporarily freeze the router and the node score. To avoid excessive subscripts, write $\widehat{\nu}_t = \widehat{\nu}_{j,t}^{(d)}$ and $\nu_t = \nu_{j,t}^{(d)}$.

**Assumption D.1** (Stationary anchor estimates)**.** *There is a deterministic scalar $\bar{\nu}$ such that*

$$\widehat{\nu}_t = \bar{\nu} + \xi_t, \qquad \mathbb{E}[\xi_t \mid \mathcal{F}_t] = 0, \qquad \mathbb{E}[\xi_t^2 \mid \mathcal{F}_t] \le s^2. \tag{40}$$

*Here $\mathcal{F}_t$ contains the model state and all batches preceding the $t$-th update, so $\nu_t$ is $\mathcal{F}_t$-measurable and $\widehat{\nu}_t$ is computed from a fresh batch. The initialization $\nu_0$ is deterministic.*

**Theorem D.2** (Stationary behavior of the anchor EMA)**.** *Under Assumption D.1, consider the constant-decay phase of Equation (5), after the warmup has reached $\alpha_t = \alpha$. Reindexing the first update in this phase as $t = 0$, the anchor satisfies*

$$\mathbb{E}[\nu_t - \bar{\nu}] = \alpha^t(\nu_0 - \bar{\nu}), \tag{41}$$

$$\mathbb{E}[(\nu_t - \bar{\nu})^2] \le \alpha^{2t}(\nu_0 - \bar{\nu})^2 + \frac{1-\alpha}{1+\alpha}(1 - \alpha^{2t})s^2. \tag{42}$$

*Thus, the initialization error decays geometrically, while the asymptotic mean-square fluctuation is at most $(1-\alpha)s^2/(1+\alpha)$.*

The theorem makes the role of $\alpha$ transparent. A larger $\alpha$ reduces the sampling-noise floor but makes the EMA slower to respond. Under soft routing every node receives positive mass in exact arithmetic, but deep-node arrival weights can be highly concentrated, increasing the variance parameter $s^2$ and the risk of numerical underflow.

**Proposition D.3** (Tracking a drifting anchor)**.** *Under the same constant-decay update, suppose*

$$\mathbb{E}[\widehat{\nu}_t \mid \mathcal{F}_t] = \bar{\nu}_t, \qquad \mathbb{E}[(\widehat{\nu}_t - \bar{\nu}_t)^2 \mid \mathcal{F}_t] \le s^2, \qquad |\bar{\nu}_{t+1} - \bar{\nu}_t| \le v, \tag{43}$$

*where $(\bar{\nu}_t)$ and $\nu_0$ are deterministic. Then*

$$\mathbb{E}[(\nu_t - \bar{\nu}_t)^2] \le \left(\alpha^t|\nu_0 - \bar{\nu}_0| + \frac{v(1-\alpha^t)}{1-\alpha}\right)^2 + \frac{1-\alpha}{1+\alpha}(1-\alpha^{2t})s^2. \tag{44}$$

The two nontransient terms exhibit the tracking trade-off of the single EMA: large $\alpha$ suppresses sampling noise, whereas small $\alpha$ reduces lag.

## D.2 Expert optimization under sparse top-$K$ routing

Freeze the router, the top-$K$ rule, and any capacity policy. Let

$$I_{e,t} := \mathbf{1}\{\text{expert } e \text{ executes example } t\}, \qquad p_e := \mathbb{P}(I_{e,t} = 1). \tag{45}$$

Thus $p_e$ is the actual execution probability after top-$K$ selection, not the expected leaf mass $q_e$ in Equation (15). Let $\theta_e$ collect the trainable parameters of expert $F_e$, and let $\mathcal{L}_e$ denote its conditional risk on the examples it executes.

**Assumption D.4** (Frozen-router expert SGD)**.** *The examples are i.i.d. and $\mathbb{P}(I_{e,t} = 1 \mid \mathcal{H}_t) = p_e$, where $\mathcal{H}_t$ is the history before example $t$, and $\theta_{e,0}$ is deterministic. The conditional risk $\mathcal{L}_e$ is $\mu_e$-strongly convex and $L_e$-smooth with minimizer $\theta_e^\star$. Conditional on $I_{e,t} = 1$, the stochastic gradient satisfies*

$$\mathbb{E}[g_{e,t} \mid \mathcal{H}_t, I_{e,t} = 1] = \nabla\mathcal{L}_e(\theta_{e,t}),$$
$$\mathbb{E}\left[\|g_{e,t} - \nabla\mathcal{L}_e(\theta_{e,t})\|^2 \mid \mathcal{H}_t, I_{e,t} = 1\right] \le \sigma_e^2. \tag{46}$$

The global-example update is

$$\theta_{e,t+1} = \theta_{e,t} - \gamma_e I_{e,t} g_{e,t}. \tag{47}$$

**Theorem D.5** (Expert progress under intermittent execution)**.** *Under Assumption D.4, if $p_e > 0$ and $0 < \gamma_e \le 1/L_e$, then*

$$\begin{aligned}\mathbb{E}\|\theta_{e,T} - \theta_e^\star\|^2 &\le (1 - p_e\gamma_e\mu_e)^T \|\theta_{e,0} - \theta_e^\star\|^2 + \frac{\gamma_e\sigma_e^2}{\mu_e} \\ &\le \exp(-p_e\gamma_e\mu_e T)\|\theta_{e,0} - \theta_e^\star\|^2 + \frac{\gamma_e\sigma_e^2}{\mu_e}.\end{aligned} \tag{48}$$

*Consequently, if $p_e \ge p_{\min} > 0$ for all experts, then, for $0 < \varepsilon < \|\theta_{e,0} - \theta_e^\star\|^2$, the transient is at most $\varepsilon$ once*

$$T \ge \frac{1}{p_{\min}\gamma_e\mu_e} \log\left(\frac{\|\theta_{e,0} - \theta_e^\star\|^2}{\varepsilon}\right). \tag{49}$$

**Implication for BRANCH-MoE.** Theorem D.5 identifies the realized execution frequency as the optimization bottleneck for sparsely activated experts: a larger minimum frequency $p_{\min}$ yields faster worst-expert convergence in global training steps. BRANCH-MoE is designed to improve this quantity by anchoring each internal decision to the population reaching that node, thereby discouraging persistent high-level routing imbalances that would simultaneously suppress every expert in an entire subtree. Although the soft-mass bound in Equation (17) does not formally imply a positive top-$K$ execution probability, it removes an important mechanism of expert starvation and should make realized execution frequencies larger and more stable than under unconstrained learned routing. Thus, the theorem explains how the improved utilization encouraged by BRANCH-MoE can translate into faster optimization of the least frequently selected experts. Uniform random routing remains the natural benchmark for marginal balance; BRANCH-MoE seeks comparable utilization while retaining task-dependent specialization and tree-aligned communication locality.

# E PROOFS

*Proof of Theorem D.2.* Let $e_t := \nu_t - \bar{\nu}$. From the constant-decay update and Equation (40),

$$e_{t+1} = \alpha e_t + (1 - \alpha)\xi_t.$$

Taking conditional expectation given $\mathcal{F}_t$ gives $\mathbb{E}[e_{t+1} \mid \mathcal{F}_t] = \alpha e_t$. Iterating proves Equation (41).

For the second moment,

$$\begin{aligned}\mathbb{E}[e_{t+1}^2 \mid \mathcal{F}_t] &= \alpha^2 e_t^2 + 2\alpha(1 - \alpha)e_t\mathbb{E}[\xi_t \mid \mathcal{F}_t] + (1 - \alpha)^2\mathbb{E}[\xi_t^2 \mid \mathcal{F}_t] \\ &\le \alpha^2 e_t^2 + (1 - \alpha)^2 s^2.\end{aligned}$$

Taking expectations and unrolling the recursion yields

$$\mathbb{E}[e_t^2] \le \alpha^{2t} e_0^2 + (1 - \alpha)^2 s^2 \sum_{k=0}^{t-1} \alpha^{2k}.$$

Using $\sum_{k=0}^{t-1} \alpha^{2k} = (1 - \alpha^{2t})/(1 - \alpha^2)$ gives Equation (42). □

*Proof of Proposition D.3.* Write $\xi_t := \widehat{\nu}_t - \bar{\nu}_t$ and $e_t := \nu_t - \bar{\nu}_t$. Then

$$e_{t+1} = \alpha e_t + (\bar{\nu}_t - \bar{\nu}_{t+1}) + (1 - \alpha)\xi_t.$$

Unrolling gives

$$\begin{aligned}e_t = \alpha^t e_0 &+ \sum_{k=0}^{t-1} \alpha^{t-1-k}(\bar{\nu}_k - \bar{\nu}_{k+1}) \\ &+ (1 - \alpha)\sum_{k=0}^{t-1} \alpha^{t-1-k}\xi_k.\end{aligned}$$

The absolute value of the deterministic first line is at most

$$\alpha^t|e_0| + v\sum_{k=0}^{t-1}\alpha^{t-1-k} = \alpha^t|e_0| + \frac{v(1-\alpha^t)}{1-\alpha}.$$

The martingale-difference property makes distinct noise terms orthogonal and eliminates their cross term with the deterministic part. Hence the noise variance is at most

$$(1-\alpha)^2 s^2 \sum_{k=0}^{t-1}\alpha^{2k} = \frac{1-\alpha}{1+\alpha}(1-\alpha^{2t})s^2.$$

Combining the two bounds proves Equation (44). □

*Proof of Lemma 4.1.* Along the path to node $(d, j)$, its mask can be written as

$$m_j^{(d)}(x) = \prod_{u\in\mathrm{path}((d,j))} \sigma(s_u z_u(x)),$$

where $s_u \in \{-1, +1\}$ selects the appropriate branch and $z_u(x)$ is an affine centered score. The function $y \mapsto \log\sigma(y) = -\log(1+\exp(-y))$ is concave. Therefore each factor $x \mapsto \sigma(s_u z_u(x))$ is log-concave, and so is their product. The unnormalized arrival density is the product of this mask and the log-concave input density, hence is log-concave. Normalization preserves log-concavity. □

*Proof of Corollary 4.2.* Let $\mu_j^{(d)} := \mathbb{E}_{\mathcal{D}_j^{(d)}}[X]$. Linearity gives

$$\nu_j^{(d),\star} = \left(W_j^{(d)}\right)^\top \mu_j^{(d)},$$

so the population-centered score is

$$Z = \left(W_j^{(d)}\right)^\top (X - \mu_j^{(d)}).$$

Grünbaum's inequality for log-concave measures implies

$$\mathbb{P}(Z \ge 0) \ge \frac{1}{\mathrm{e}}, \qquad \mathbb{P}(Z \le 0) \ge \frac{1}{\mathrm{e}}.$$

At the exact population anchor, $\pi_{j,+}^{(d)} = \mathbb{E}[\sigma(Z)]$ and $\pi_{j,-}^{(d)} = \mathbb{E}[\sigma(-Z)]$, where both expectations are under $\mathcal{D}_j^{(d)}$. Since $\sigma(y) \ge 1/2$ for $y \ge 0$, each expectation is at least $1/(2\mathrm{e})$, proving Equation (13).

For an arbitrary active anchor, the two expectations become $\mathbb{E}[\sigma(Z - \delta_j^{(d)})]$ and $\mathbb{E}[\sigma(\delta_j^{(d)} - Z)]$. They are bounded below by one half of $\mathbb{P}(Z \ge \delta_j^{(d)})$ and $\mathbb{P}(Z \le \delta_j^{(d)})$, respectively. If the density of $Z$ is bounded by $A_j^{(d)}$, moving the threshold from zero to $\delta_j^{(d)}$ changes either tail probability by at most $A_j^{(d)}|\delta_j^{(d)}|$. Combining this fact with Grünbaum's bounds and truncating the resulting lower bound at zero proves Equation (14). □

*Proof of Theorem D.5.* Write $u_t = \theta_{e,t} - \theta_e^\star$. Conditional on $\mathcal{H}_t$, unbiasedness and Equation (47) imply

$$\begin{aligned}\mathbb{E}[\|u_{t+1}\|^2 \mid \mathcal{H}_t] = \|u_t\|^2 &- 2p_e\gamma_e\langle u_t, \nabla\mathcal{L}_e(\theta_{e,t})\rangle \\ &+ p_e\gamma_e^2\mathbb{E}[\|g_{e,t}\|^2 \mid \mathcal{H}_t, I_{e,t} = 1].\end{aligned}$$

Strong convexity gives $\langle u_t, \nabla\mathcal{L}_e(\theta_{e,t})\rangle \ge \mu_e\|u_t\|^2$. Smoothness, optimality of $\theta_e^\star$, and Equation (46) give

$$\mathbb{E}[\|g_{e,t}\|^2 \mid \mathcal{H}_t, I_{e,t} = 1] \le \|\nabla\mathcal{L}_e(\theta_{e,t})\|^2 + \sigma_e^2.$$

Because a differentiable convex $L_e$-smooth function has an $L_e^{-1}$-cocoercive gradient, $\|\nabla\mathcal{L}_e(\theta_{e,t})\|^2 \le L_e\langle u_t, \nabla\mathcal{L}_e(\theta_{e,t})\rangle$. Therefore, using $\gamma_e \le 1/L_e$,

$$\mathbb{E}[\|u_{t+1}\|^2 \mid \mathcal{H}_t] \le (1 - p_e\gamma_e\mu_e)\|u_t\|^2 + p_e\gamma_e^2\sigma_e^2.$$

Iteration of this recursion and summation of the geometric series yield

$$\mathbb{E}\|u_T\|^2 \le (1 - p_e\gamma_e\mu_e)^T\|u_0\|^2 + \frac{\gamma_e\sigma_e^2}{\mu_e}.$$

Finally, $1 - x \le \exp(-x)$ gives the second line of Equation (48). If $p_e \ge p_{\min}$, replacing $p_e$ by $p_{\min}$ in the exponential transient and solving for $T$ gives Equation (49). □

*Proof of Theorem 4.3.* The leaf masks below $u_r = (r, j_r)$ sum to its incoming mask:

$$\sum_{e\in\mathcal{L}(u_r)} m_{e,t}^{(D)}(x) = m_{j_r,t}^{(r)}(x). \tag{50}$$

This follows by induction from the cascade, because the two children of a node with incoming mask $m$ receive masses $m(1-p)$ and $mp$, whose sum is $m$.

Along the preferred path, the probability assigned at level $i$ is $\sigma(\gamma_i(x))$. Therefore,

$$\sum_{e\in\mathcal{L}(u_r)} m_{e,t}^{(D)}(x) = m_{j_r,t}^{(r)}(x) = \prod_{i=0}^{r-1}\sigma(\gamma_i(x)).$$

Since the full leaf mask sums to one,

$$\varepsilon_r(x) = 1 - \prod_{i=0}^{r-1}\sigma(\gamma_i(x)).$$

For numbers $a_i \in [0,1]$, $1 - \prod_i(1-a_i) \le \sum_i a_i$. Taking $a_i = \sigma(-\gamma_i(x))$ gives

$$\varepsilon_r(x) \le \sum_{i=0}^{r-1}\sigma(-\gamma_i(x)) \le \sum_{i=0}^{r-1} e^{-\gamma_i(x)},$$

which proves Equation (23).

Let

$$h_r(x) := |\mathcal{S}_K(x) \setminus \mathcal{L}(u_r)|.$$

Every selected leaf has probability at least $\tau_K(x)$, and hence

$$\varepsilon_r(x) \ge \sum_{e\in\mathcal{S}_K(x)\setminus\mathcal{L}(u_r)} m_{e,t}^{(D)}(x) \ge h_r(x)\tau_K(x).$$

Because $h_r(x)$ is integer, this proves Equation (24).

All selected experts inside $\mathcal{L}(u_r)$ occupy one domain, and every selected expert outside it can add at most one further domain. Thus $N_r(x) \le 1 + h_r(x)$, while trivially $N_r(x) \le K$. This proves Equation (25).

Finally, Equation (26) implies $\varepsilon_r(x) < \tau_K(x)$. If a selected expert lay outside $\mathcal{L}(u_r)$, its probability would be at least $\tau_K(x)$, which would contradict the definition of $\varepsilon_r(x)$. Hence every selected expert lies in the same depth-$r$ domain. □

# F Experimental Details

## F.1 Datasets

Table 3 summarizes the four benchmarks. Criteo is the primary large-scale task; the three UCI datasets are tabular benchmarks that admit many-seed replication at modest cost.

[2]The held-out split is exactly 4.42M examples (one full evaluation pass of 2,158 steps at batch 2048). The training split is not counted directly: we infer ≈37M from the ratio of on-disk sizes (5.93 GB train against 0.71 GB held-out), which assumes a common mean record size across the two splits. All reported results depend on the training *step* count, not on this figure, so the corpus size should be read as approximate.

Table 3: Benchmark suite. All MoE conditions use $E = 16$ experts with top-4 routing. "Examples" is the full corpus size (train + held-out); "Seeds" is the number of independent replications per condition.

| Dataset | Task | Examples | Features | Seeds | Metric |
|---|---|---|---|---|---|
| Criteo (small) | CTR (binary) | ≈41.5M[2] | 39 fields | 5 | Logloss / AUC |
| Forest Covertype | 7-class | 581K | 54 | 5 | Cross-entropy / Acc. |
| HIGGS | Binary | 600K | 28 | 5 | Logloss / AUC |
| YearPredictionMSD | Regression | 515K | 90 | 5 | MSE / RMSE (yr) |

**Criteo.** Each example comprises 13 integer (count) features and 26 categorical features, for 39 fields in total, with a binary click label. Categorical fields are mapped to integer IDs by a string lookup built from a vocabulary with a minimum-frequency cutoff of 100; post-cutoff vocabulary sizes $V$ range from 4 to 20,136. Integer fields are discretized into 41 bins by the boundaries $b_i = 1.5^{\,i-1} - 0.49$ for $i = 0, \ldots, 39$, converting magnitude into an ordered categorical ID. Embeddings are not shared across fields: integer fields use width 10, categorical fields use $d_f = \min(\lfloor 6V_f^{1/4} \rfloor, V_f) \in [4, 71]$, and the concatenation yields a 1137-dimensional input.

**UCI benchmarks.** Forest Covertype (54 cartographic features, 7 cover classes, $80/20$ split) and HIGGS (28 particle-physics features, the first 600K of the 11M-event corpus, 500K/100K split) are standardized. YearPredictionMSD (90 timbre statistics) uses the canonical $463{,}715/51{,}630$ split, which prevents artist leakage; its features are likewise standardized, the target is the standardized release year, and we report unstandardized RMSE in years alongside the MSE objective. HIGGS and MSD inputs are additionally clipped at $\pm 10\sigma$; Covertype inputs are not.

On Forest Covertype one preprocessing detail proves critical: 40 of the 54 features are binary soil-type indicators, the rarest firing on only 2 of the 464,809 training rows. Standard $z$-scoring with an $\epsilon = 10^{-7}$ denominator guard maps a column that is constant in the sample used to fit the normalizer to spikes of magnitude $\sim 10^7$ on the rows where it does fire, which corrupts running statistics and induces bimodal expert collapse in tree routers. Pinning any column with adapted variance below $10^{-4}$ to unit variance eliminates the collapse and reduces the seed-to-seed accuracy spread of BRANCH-MoE from 6.2 to 0.46 percentage points.

### F.2 Baselines

We benchmark against four representative MoE routing paradigms, covering auxiliary-loss, auxiliary-loss-free, normalization-based, and deterministic mechanisms.

1. **Top-$k$ Softmax Routing (Switch).** The canonical learned paradigm (Fedus et al., 2022): a learned gating matrix, a softmax, and top-$k$ dispatch, with multiplicative uniform jitter $[1 - \epsilon, 1 + \epsilon]$, $\epsilon = 10^{-2}$, during training. Load balance is enforced by the auxiliary loss $\mathcal{L}_{\text{balance}} = \alpha E \sum_e f_e P_e$ with $\alpha = 0.01$.
2. **Auxiliary-Loss-Free Routing (DeepSeek-V3).** Affinities are computed with an independent sigmoid per expert, so experts do not compete for probability mass (Liu et al., 2024). A non-trainable additive bias $b_e$ enters top-$k$ selection only and is updated after each step by a fixed increment $\gamma = 10^{-3}$ in the direction that relieves congestion; the combining gates remain the unbiased affinities.
3. **Gating Logit Normalization (Skywork).** Logits are standardized across the expert dimension before the softmax, which counteracts representation collapse and sharpens the gate distribution (Wei et al., 2024). A fixed auxiliary balancing coefficient is retained. We include this baseline because it proves to be the strongest competitor on two of the four benchmarks and ties Switch as the runner-up on a third.
4. **Deterministic Balanced Routing (Hash).** A fixed assignment mapping each example to experts by a 32-bit avalanche hash of its feature vector, following Hash Layers (Roller et al., 2021). The assignment is uniform by construction but carries no dependence on content beyond the hash, isolating the value of *learned* gating against a perfectly balanced but semantically uninformed one, at zero router parameter cost.

5. **BRANCH-MoE (ours).** The $E$ experts are the leaves of a balanced binary tree of depth $\log_2 E$. At internal node $n$ the probability of branching right is $\sigma(f_n(\mathbf{x}) - \mu_n)$. During training $\mu_n$ is the arrival-weighted mean score of the current batch at $n$; at inference it is $\nu_n$, a non-trainable exponential moving average (decay $0.99$) of that batch mean, updated at every training step. Centering each split on the mean score of its own traffic places the decision boundary through the middle of the distribution actually arriving there, keeping both children populated without a balancing penalty. Leaf scores are products of branch probabilities along each root-to-leaf path; all $E - 1$ node scores are evaluated, and the products are formed in $\log_2 E$ vectorized steps.

### F.3 Node Maps, Architecture, and Protocol

**Node maps.** The score $f_n$ at each node, and correspondingly the logit map of each flat router, is one of two forms, swept as a factor in its own right:

- **Linear:** $f_n(\mathbf{x}) = \mathbf{w}_n^\top \mathbf{x}$.
- **MLP:** a two-layer network of hidden width $h \in \{2, 4, 8\}$ per node, $f_n(\mathbf{x}) = \mathbf{v}_n^\top \mathrm{ReLU}(\mathbf{U}_n \mathbf{x} + \mathbf{c}_n)$.

This factor is swept across all four learned routers and the Hash control on the three UCI benchmarks, and across all four learned routers on Criteo.

**Architecture and protocol.** To isolate routing we adopt a minimal backbone: an input stage, one MoE block, and a linear head. On Criteo the input stage projects the 39 embeddings to $256$ dimensions, followed by SiLU and layer normalization; on the UCI datasets it is a feature-standardization layer alone, with no projection. The MoE block has no residual connection. Experts are SwiGLU networks, $\mathrm{down}(\mathrm{SiLU}(\mathrm{gate}(\mathbf{x})) \odot \mathrm{up}(\mathbf{x}))$; a post-MoE layer normalization precedes the head on Criteo, HIGGS, and MSD (the Covertype model has none). All conditions use $E = 16$ and top-$4$ routing with capacity factor $1.25$, and differ only in the gate $g(\cdot)$, so comparisons are at matched capacity. Criteo trains for $37{,}500$ steps at batch $2048$ with Adam (lr $= 5 \times 10^{-4}$), organized as $15$ evaluation intervals of $2{,}500$ steps; since an interval covers $5.12$M examples against a $\approx$37M training split, this is $\approx 2.1$ passes over the data rather than $15$. The UCI datasets train for $20$ true epochs with Adam (lr $= 10^{-3}$) at batch $512$ (Covertype) or $256$ (HIGGS, MSD). Every condition is replicated over $5$ seeds. On the UCI benchmarks the seed governs initialization but not the data order, which is shuffled without a seed; all arms within a dataset share the identical seed set, so comparisons are *paired* by seed. On Criteo the seed likewise governs initialization only, and TPU execution is not bitwise reproducible, so Criteo comparisons are unpaired (two-sided Welch $t$-tests). Re-running an identical Criteo configuration with the same seed changes held-out logloss about as much as changing the seed does (standard deviation $1.5$–$2.8 \times 10^{-4}$ within a seed against $1.9$–$3.0 \times 10^{-4}$ across seeds).

### F.4 Locality Metrics

For each token routed to experts $(i, j)$ we measure two index-based distances, averaged over all $\binom{k}{2}$ co-selected pairs.

1. **Tree distance $L$.** Reading an expert index as a $D$-bit root-to-leaf path ($D = \log_2 E$), the number of levels to the lowest common ancestor is $L(i, j) = \mathrm{bit_length}(i \oplus j) \in \{1, \ldots, D\}$. Under block placement across $P = 2^m$ devices, two experts share a device if and only if $L(i, j) \le D - m$, making $L$ an exact topological proxy for cross-device fanout.
2. **Index gap $|i - j|$.** The linear gap between expert indices.

Both require a reference: for uniformly random *distinct* pairs, $\mathbb{E}[L] = (D - 1 + 2^{-D})/(1 - 2^{-D})$ and $\mathbb{E}[|i - j|] = (E + 1)/3$, giving $3.2667$ and $5.6667$ at $E = 16$ ($D = 4$). Normalized by $D$ and $E - 1$ these are $0.8167$ and $0.3778$ — random routing already scores $0.82$ on the tree metric, so locality is only evidenced *below* these values, and an unreferenced number is uninterpretable. We also report the load coefficient of variation $\mathrm{CV} = \mathrm{std}(f)/\mathrm{mean}(f)$ of the per-expert dispatch distribution.

Tree distance is also the more sensitive of the two index-based measures: it separates power-of-two boundary pairs such as $(7, 8)$, which have $|i - j| = 1$ but lie at maximal tree separation, and which the index gap therefore scores as maximally local when they are maximally remote.

Table 4: Criteo, $E = 16$, top-4, capacity factor 1.25, 5 seeds (mean $\pm$ std). Logloss is binary cross-entropy; logloss and AUC are the best held-out values over the 15 evaluations. Load CV and the normalized tree and index distances are training-time routing statistics; random baselines are 0.8167 and 0.3778. Upper block: every router with the linear node map. Lower block: the two routers whose lowest-loss node map is an MLP, at that map (for Skywork and Switch it is the linear map). Bold marks the best logloss and AUC. At 5 vs. 5 seeds, differences below $\approx 3.7 \times 10^{-4}$ in logloss or $\approx 2.7 \times 10^{-4}$ in AUC are not resolvable (Welch $t$-test, $p < 0.05$).

| Method | Best val. logloss ↓ | Best val. AUC ↑ | Load CV ↓ | Tree ↓ | Index ↓ |
|---|---|---|---|---|---|
| *Linear node map* | | | | | |
| Skywork | **0.446034** ±0.000298 | **0.805827** ±0.000184 | 0.0040 | 0.8184 | **0.3778** |
| DeepSeek-V3 | 0.446082 ±0.000270 | 0.805799 ±0.000174 | 0.0057 | 0.8188 | 0.3849 |
| BRANCH-MoE (ours) | 0.446149 ±0.000250 | 0.805707 ±0.000175 | 0.1152 | **0.8126** | 0.3843 |
| Switch softmax | 0.446226 ±0.000185 | 0.805595 ±0.000204 | 0.0025 | 0.8212 | 0.3812 |
| Hash | 0.447065 ±0.000277 | 0.804716 ±0.000116 | **0.0023** | 0.8167 | **0.3778** |
| *Best MLP node map* | | | | | |
| BRANCH-MoE ($h$=4) | **0.445959** ±0.000153 | **0.805865** ±0.000022 | 0.2676 | 0.7572 | 0.3255 |
| DeepSeek-V3 ($h$=8) | 0.446049 ±0.000047 | 0.805970 ±0.000100 | 0.0038 | 0.8190 | 0.3824 |

# G Additional Results

## G.1 Criteo: Click-Through-Rate Prediction

Table 4 reports the benchmark. With the common linear node map, the four learned routers are statistically tied: their mean held-out logloss spans $1.9 \times 10^{-4}$, about half of the smallest resolvable difference, and BRANCH-MoE ranks third, $1.1 \times 10^{-4}$ behind Skywork (Welch $p = 0.53$). When each router uses its best node map, BRANCH-MoE with $h = 4$ has the lowest mean logloss, but its margins over Skywork ($7.5 \times 10^{-5}$, $p = 0.63$) and DeepSeek-V3 at $h = 8$ ($9.0 \times 10^{-5}$, $p = 0.26$) are well inside seed noise; choosing the best of four node maps after the fact also favours it, and DeepSeek-V3 at $h = 8$ has the higher AUC ($+1.05 \times 10^{-4}$, $p = 0.08$). We therefore claim parity, not superiority. The only unambiguous gap is to Hash, which trails every other row of the table by 0.0008–0.0011 in logloss and 0.0009–0.0013 in AUC (Welch $p < 0.002$ in every comparison). All Criteo tests are unpaired, because TPU execution is not bitwise reproducible (Section F.3).

**Balance.** BRANCH-MoE keeps every expert in use without an auxiliary loss or bias controller, but balances load less tightly than the flat routers: with the linear node map its load CV is 0.115 (largest and smallest expert shares 1.17× and 0.72× the mean), against at most 0.006 for every flat router. At capacity factor 1.25 this has no measurable cost: every linear-map condition drops at most 0.11% of dispatches at capacity, and BRANCH-MoE drops 0.08%.

**Locality does not appear on Criteo.** With the linear node map, BRANCH-MoE's normalized tree distance is $0.813 \pm 0.012$, indistinguishable from the 0.8167 random baseline (one-sample $t$-test, $p = 0.48$), and its index distance (0.384) is marginally *above* random ($p = 0.04$). The tree does not start at random: an untrained BRANCH-MoE scores 0.725, and training moves it to the random value. The flat routers stay within 2% of both random baselines with linear node maps, as expected for arbitrary expert labels. Lower tree distances do appear with MLP node maps (0.757 for BRANCH-MoE at $h = 4$), but they vary strongly across seeds, are lowest in the most imbalanced runs, and appear also for the flat Skywork router (0.75–0.77, $p \leq 0.05$ at every width), whose expert indices carry no meaning. We have not diagnosed the Skywork effect; it points to an index-order artifact in expert selection, such as tie-breaking among exactly equal logits, and we read neither result as locality. Criteo is therefore the one benchmark on which the locality of Section G.4 does

Table 5: Forest Covertype, $E = 16$, top-4, 20 epochs, batch 512, 5 paired seeds (mean $\pm$ std). $h$ is the MLP node-map hidden width (Section F.3). Hash ignores the node map by construction; its four arms are the same configuration run four times and serve as an exact negative control.

| | Best val. cross-entropy ↓ | | | |
|---|---|---|---|---|
| Method | linear | $h$=2 | $h$=4 | $h$=8 |
| BRANCH-MoE (ours) | **0.25451** ±0.00408 | 0.25206 ±0.00487 | 0.24394 ±0.00356 | **0.23720** ±0.00125 |
| Skywork | 0.25862 ±0.00365 | **0.25169** ±0.00324 | **0.24277** ±0.00494 | 0.24062 ±0.00358 |
| Switch softmax | 0.26150 ±0.00136 | 0.25646 ±0.00713 | 0.24286 ±0.00180 | 0.24062 ±0.00281 |
| DeepSeek-V3 | 0.27032 ±0.00225 | 0.26620 ±0.00301 | 0.25417 ±0.00196 | 0.24207 ±0.00326 |
| Hash (control) | 0.44665 ±0.00087 | 0.44771 ±0.00084 | 0.44707 ±0.00162 | 0.44652 ±0.00108 |

not replicate. It differs from the UCI benchmarks in input representation (learned embeddings of categorical fields), batch size (2048 against 256–512), and training length ($\approx$2.1 passes over the data rather than 20 epochs); we have not isolated which of these is responsible.

### G.2 Forest Covertype: Multi-Class Classification

Table 5 reports the node-map sweep. BRANCH-MoE with $h = 8$ is the best arm overall (0.23720 cross-entropy, 90.21% accuracy, best on both), but Skywork and Switch tie for second at the same $h$ (0.24062 each), and the margin of $-0.00342$ over them is not clean. Against Switch a paired $t$-test resolves it only marginally ($t_4 = -2.86$ against a critical value of 2.776) and the unpaired Welch test does not ($t = -2.49$); against Skywork neither test resolves it (paired $t_4 = -1.64$, Welch $t = -2.01$). The margin over DeepSeek-V3 at $h = 8$ is cleaner ($-0.00487$, paired $t_4 = -3.25$). With linear node maps BRANCH-MoE again has the lowest mean: its gap to Switch is larger and resolved ($-0.00699$, paired $t_4 = -3.25$), but its gap to Skywork, the runner-up, is not ($-0.00411$, paired $t_4 = -1.26$). At $h = 2$ and $h = 4$ Skywork has the lowest mean, by margins of 0.0004 and 0.0001 that are far inside seed noise.

The four Hash arms are four runs of one identical configuration, so the spread of their seed means isolates pure run-to-run nondeterminism: $\mathrm{sd} = 0.00053$. The BRANCH-MoE-vs-Switch gap at $h = 8$ is well above this floor yet not resolved by the unpaired test, which is precisely why pairing is required to see it at all; the equal gap to Skywork is not resolved even with pairing.

On balance BRANCH-MoE attains $\mathrm{CV} = 0.093$ with a $1.174\times$ peak load at linear (0.112 / $1.240\times$ at $h = 8$), against 0.121 / $1.210\times$ for Switch and 0.248 / $1.366\times$ for Skywork, the least balanced learned router at linear (0.137 / $1.247\times$ at $h = 8$); DeepSeek-V3's controller is tighter still (0.001 / $1.002\times$), as expected from a mechanism that optimizes balance directly. The only dead experts (below 1% of uniform load) are Skywork's at $h = 2$: 6 of its 80 expert slots across the five seeds.

### G.3 HIGGS and YearPredictionMSD

Table 6 reports the two remaining benchmarks, and the outcome is different. On HIGGS BRANCH-MoE ranks second behind Skywork ($+0.0009$, within the combined seed spread); on MSD it ranks fourth, behind DeepSeek-V3, Skywork, and Switch, though the four are separated by only 0.0021 MSE — 0.014 years of RMSE — against seed standard deviations of 0.0015–0.0043. The honest summary is that on these two datasets all learned routers are statistically tied, and BRANCH-MoE is not the best point estimate.

Table 6: HIGGS (logloss) and YearPredictionMSD (MSE), $E = 16$, top-4, 5 paired seeds (mean $\pm$ std). Each router is shown at its best node map, given in parentheses; for BRANCH-MoE we show both $h = 8$ (best loss) and linear (best locality), since the two differ by 0.0003 on HIGGS. Tree and index distances are normalized against the 0.8167 / 0.3778 random baselines. Hash ignores the node map, so its four arms are one configuration run four times; the best is shown.

| | HIGGS | | | YearPredictionMSD | | |
|---|---|---|---|---|---|---|
| Method | Logloss ↓ | Tree ↓ | Index ↓ | MSE ↓ | Tree ↓ | Index ↓ |
| Skywork | **0.5247** (lin) ±0.0005 | 0.812 | 0.370 | 0.6757 ($h$=2) ±0.0015 | 0.813 | 0.365 |
| BRANCH-MoE ($h$=8) | 0.5256 ±0.0014 | 0.736 | 0.322 | 0.6767 ±0.0027 | 0.749 | 0.324 |
| BRANCH-MoE (linear) | 0.5259 ±0.0008 | **0.710** | **0.292** | 0.6843 ±0.0035 | **0.718** | **0.301** |
| DeepSeek-V3 | 0.5264 (lin) ±0.0013 | 0.816 | 0.380 | **0.6746** ($h$=8) ±0.0043 | 0.813 | 0.371 |
| Switch softmax | 0.5276 ($h$=4) ±0.0003 | 0.814 | 0.374 | 0.6766 ($h$=8) ±0.0038 | 0.800 | 0.365 |
| Hash | 0.5574 ($h$=4) ±0.0007 | 0.817 | 0.378 | 0.6899 ($h$=4) ±0.0024 | 0.817 | 0.378 |

Table 7: Normalized co-routing distances ($E = 16$, top-4; mean over 5 seeds). Flat routers: range over Switch, DeepSeek-V3, Skywork, and Hash, each at the arm shown in Table 1. Bold: most local.

| | BRANCH-MoE (linear) | | BRANCH-MoE ($h$=8) | | Flat routers | |
|---|---|---|---|---|---|---|
| Dataset | Tree | Index | Tree | Index | Tree | Index |
| Forest Covertype | **0.718** | **0.312** | 0.742 | 0.334 | 0.813–0.819 | 0.371–0.385 |
| HIGGS | **0.710** | **0.292** | 0.736 | 0.322 | 0.812–0.817 | 0.370–0.380 |
| YearPredictionMSD | **0.718** | **0.301** | 0.749 | 0.324 | 0.800–0.817 | 0.365–0.378 |
| Criteo | 0.813 | 0.384 | 0.804 | 0.370 | 0.817–0.821 | 0.378–0.382 |
| Random baseline | 0.817 | 0.378 | 0.817 | 0.378 | 0.817 | 0.378 |

### G.4 Routing Locality Across the Suite

The locality prediction of Section 4.4 is the claim that survives most cleanly. On all three UCI benchmarks BRANCH-MoE is the *only* router whose co-routing statistics depart materially from the random baseline (Table 7).

On the three UCI datasets the reduction is 12–13% below baseline on tree distance for linear BRANCH-MoE (8–10% at $h = 8$) and 18–23% on index distance (12–15% at $h = 8$), while every flat router sits within 3.5% of baseline on both — the negative control behaving exactly as expected, since for a flat router the expert index is an arbitrary label. Criteo is the exception: there BRANCH-MoE stays at the random baseline, and with MLP node maps even the flat Skywork router falls below it, an artifact discussed in Section G.1.