How Sparse Probability Maps Shape Mixture-of-Experts Routing

Anonymous authors
Paper under double-blind review

Abstract

Mixture-of-experts (MoE) routers typically apply softmax to the router scores and keep the top-K experts, making every token use exactly K experts. Sparsity-inducing probability maps such as sparsemax, α-entmax and normmax can adaptively assign *exact zeros* to selected experts, and therefore appear to offer token-dependent expert participation, even when using the same top-K machinery. In this work, we study whether and how this sparsity survives training. We train matched 300M and 1B top-2 MoE language models with softmax, 1.5-entmax, sparsemax and 2-normmax, and find that the maps behave very differently once trained: at 1B, entmax discards 30% less probability mass than softmax while almost never dropping a selected expert, sparsemax retains the most mass, and normmax routes 21% of tokens to a single expert. These outcomes are not properties of the maps alone. Each map drops a selected expert only when the gap between the two largest scores reaches a fixed threshold, and the trained routers differ in the score distribution they learn: the entmax router learns scores with roughly half the spread of softmax's, which keeps its top-2 gaps below its threshold, while sparsemax and normmax, which share the same threshold, learn different gap distributions and hence different participation. Routers thus co-adapt their scores to the map, and a map's capacity to produce zeros does not by itself determine expert participation. While none of the sparse maps improves validation loss over softmax, they make the trained models far less sensitive to selecting more experts at inference: sparsemax trained with $K{=}2$ loses 0.02 nats when run with $K{=}8$, where softmax loses 0.58. Our results indicate that adaptive MoE routing has to be designed around the joint behavior of the probability map and the learned scores, rather than around the map alone.[1]

1 Introduction

Mixture-of-experts (MoE) layers scale model capacity by activating only a few expert networks per token (Shazeer et al., 2017; Fedus et al., 2022; Jiang et al., 2024). A router scores the experts, a probability map turns the scores into weights, and a top-K operator keeps the K highest-weighted experts, whose outputs are combined with the renormalized weights. With softmax, every token uses exactly K experts, regardless of how many it needs, and the selected experts often account for only a fraction of the router's probability mass—the rest is discarded by truncation.

Sparsity-inducing probability maps promise a way out without changing the top-K machinery. Sparsemax (Martins & Astudillo, 2016), α-entmax (Peters et al., 2019) and normmax (Santos et al., 2024; 2025) can assign *exact zeros*, so a selected expert may receive zero weight and drop out of the mixture. The number of experts that actually contribute to a token is then *data-dependent*, even when the expert budget K stays fixed. Several recent routers also build on this idea (Oldfield et al., 2024; Thaman, 2025; Wang et al., 2025). However, how sparsity behaves in these models is not obvious. Since the router scores are learned, changing the map also changes the scores that the map receives: a trained sparsemax router and a trained softmax router differ both in how they transform scores and in which scores they learned to produce. As a result, a map that can produce zeros may rarely do so once its router has been trained

[1] Our code will be released upon acceptance.

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.

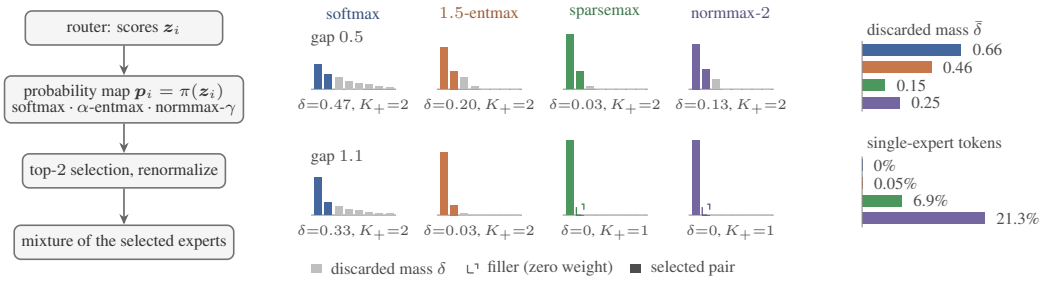

Figure 1: Left: in a top-2 MoE router, the probability map π is the only component we change. Middle: on the same router scores, the maps differ in how much probability mass falls outside the selected pair (discarded mass δ, gray) and in whether a selected expert receives zero weight (a *filler*, dashed), which lowers the number of contributing experts K_+; shown for two illustrative 8-expert score vectors whose top-2 gaps are 0.5 and 1.1. Right: statistics of our trained 1B MoE models.

In this paper, we study whether the sparsity of the probability map survives training, and what determines expert participation when it does not. We train matched 300M and 1B top-2 MoE language models with softmax, 1.5-entmax, sparsemax and 2-normmax under an identical architecture, data and balancing objective, and we measure the probability mass retained by the selected experts (*retention*), the number of selected experts with non-zero weight (*participation*), and predictive quality. To relate the map to the scores it receives, we record the router scores of every trained model and measure how their spread and top-2 gaps sit relative to each map's sparsification threshold. Figure 1 summarizes the setting and the outcomes. We find that a map's ability to produce zeros does not, by itself, determine how many experts a token uses. Our main contributions are:

- **A controlled study of sparse maps as MoE routers.** We train top-2 MoE language models at two scales that differ only in the probability map, and measure retention, participation, load balance and predictive quality, together with the router scores of every trained model (§3).
- **Retention and participation are distinct outcomes.** Under top-2 selection, mass is discarded only when more than two experts have positive probability, while participation drops only when a single expert does (Equation 7). The trained routers land in different regimes: at 1B, 1.5-entmax discards 30% less mass than softmax yet keeps both experts for 99.95% of tokens, whereas normmax-2 routes 21% of tokens to a single expert (§4.2).
- **The map's threshold and the learned scores jointly set participation.** Each map drops a selected expert exactly when the top-2 gap reaches a fixed threshold. The sparse routers learn smaller score spreads than softmax, keeping 1.5-entmax's gaps below its threshold of 2; sparsemax and normmax-2 share the threshold 1, yet route 6.9% and 21.3% of tokens to one expert, as their routers learn different gap distributions (§4.3).
- **Sparse maps reduce sensitivity to the inference-time budget.** None improves validation loss at the trained budget, but running the 1B models trained with K=2 at K=8 costs sparsemax 0.02 nats against 0.58 for softmax, as sparsemax gives zero weight to most of the extra slots (§4.4).

2 Background: Probability Maps and Expert Selection

A bias-free router maps a token representation $\boldsymbol{x}_i \in \mathbb{R}^d$ to scores $\boldsymbol{z}_i = \boldsymbol{W}_r \boldsymbol{x}_i \in \mathbb{R}^n$ and to probabilities $\boldsymbol{p}_i = \pi(\boldsymbol{z}_i) \in \triangle^n$, where $\triangle^n = \{\boldsymbol{p} \in \mathbb{R}^n : \boldsymbol{p} \geq \boldsymbol{0},\ \boldsymbol{1}^\top \boldsymbol{p} = 1\}$ is the $(n-1)$-dimensional probability simplex. We write $\mathbb{1}[\cdot]$ for the indicator function, and $[a]_+ = \max(a, 0)$.

Selection and combination. For $1 \leq K \leq n$, let $\mathcal{S}_i = \mathrm{TopK}(\boldsymbol{p}_i, K)$ be the K selected experts (ties are broken by score, then by index). The *contributing* experts are those selected with positive weight, $\mathcal{T}_i = \{j \in \mathcal{S}_i : p_{ij} > 0\}$. The retained mass, the normalized mixture weights and the layer output are

$$m_i = \sum_{j \in \mathcal{S}_i} p_{ij}, \qquad q_{ij} = \frac{p_{ij}\, \mathbb{1}[j \in \mathcal{S}_i]}{m_i}, \qquad \mathrm{MoE}(\boldsymbol{x}_i) = \sum_{j \in \mathcal{S}_i} q_{ij}\, E_j(\boldsymbol{x}_i), \tag{1}$$

where E_j is the jth expert network. Under softmax, $\mathcal{T}_i = \mathcal{S}_i$ always. Under a sparse map, a selected expert can have $p_{ij} = 0$; we call such a selection a *filler*. The task loss depends on the router only through $\boldsymbol{q}_i$, while the full distribution $\boldsymbol{p}_i$ also enters the balancing objective below.

2.1 Sparsity-inducing probability maps

We suppress the token index and write $\boldsymbol{p} = \pi(\boldsymbol{z})$. All maps we consider are regularized argmax transformations (Blondel et al., 2020),

$$\pi_H(\boldsymbol{z}) = \operatorname*{arg\,max}_{\boldsymbol{p} \in \triangle^n} \boldsymbol{p}^\top \boldsymbol{z} + H(\boldsymbol{p}), \tag{2}$$

and differ only in the entropy H. Shannon entropy, $H^{\mathrm{S}}(\boldsymbol{p}) = -\sum_j p_j \log p_j$, gives softmax, $p_j = e^{z_j} / \sum_k e^{z_k}$, which is dense: every expert receives positive probability at finite scores. The Tsallis entropy $H^{\mathrm{Ts}}_\alpha(\boldsymbol{p}) = \big(1 - \sum_j p_j^\alpha\big)/(\alpha(\alpha-1))$ with $\alpha > 1$ gives α-entmax (Peters et al., 2019),

$$p_j = \big[(\alpha-1)z_j - \tau\big]_+^{1/(\alpha-1)}, \tag{3}$$

where the threshold τ is the unique value for which the probabilities sum to one. Experts whose score falls below $\tau/(\alpha-1)$ receive exactly zero probability; $\alpha \to 1$ recovers softmax and $\alpha = 2$ gives sparsemax, the Euclidean projection onto the simplex (Martins & Astudillo, 2016). The norm entropy $H^{\mathrm{N}}_\gamma(\boldsymbol{p}) = 1 - \|\boldsymbol{p}\|_\gamma$ with $\gamma > 1$ gives γ-normmax (Santos et al., 2024; 2025),

$$p_j = \frac{[z_j - \tau]_+^{1/(\gamma-1)}}{\sum_k [z_k - \tau]_+^{1/(\gamma-1)}}, \qquad \sum_j [z_j - \tau]_+^{\gamma/(\gamma-1)} = 1, \tag{4}$$

which is also sparse but, unlike entmax, favors distributions that are closer to uniform on their support. We use $\alpha = 1.5$, for which the threshold has an exact sorting-based solution and which has been the default in sparse attention work (Peters et al., 2019; Correia et al., 2019), and $\gamma = 2$, the Euclidean case studied by Santos et al. (2025). All four maps are differentiable almost everywhere and are trained by ordinary backpropagation.

Singleton thresholds. With the normalizations above, a map assigns the whole mass to a single expert exactly when the gap $g = z_{(1)} - z_{(2)}$ between the two largest scores reaches a fixed threshold, namely $g \geq 2$ for 1.5-entmax and $g \geq 1$ for both sparsemax and normmax-2 (Appendix B); softmax has no finite threshold. In particular, on the same scores, sparsemax and normmax-2 always agree on whether a token has one or two contributors.

Balancing objective. For a microbatch of M tokens, we use the Switch-style objective (Fedus et al., 2022),

$$f_j = \frac{1}{MK} \sum_i \mathbb{1}[j \in \mathcal{S}_i], \qquad \bar{p}_j = \frac{1}{M} \sum_i p_{ij}, \qquad \mathcal{L}_{\mathrm{LB}} = n \sum_j f_j \bar{p}_j, \tag{5}$$

where the selection frequencies $\boldsymbol{f}$ include fillers and are detached from differentiation. Because $\mathcal{L}_{\mathrm{LB}}$ uses the full probabilities $\boldsymbol{p}_i$ rather than the selected mixture $\boldsymbol{q}_i$, its gradient depends on the map even at a shared coefficient; under a sparse map, an expert with $p_{ij} = 0$ receives no balancing gradient from token i.

2.2 Routing measurements

We characterize routing by how much probability mass survives selection, how many selected experts contribute, and how evenly assignments are spread over experts. The *discarded mass* of token i is the mass of the unselected experts,

$$\delta_i = \sum_{j \notin \mathcal{S}_i} p_{ij} = 1 - m_i \in [0,\, 1 - K/n], \tag{6}$$

and we report its average $\bar{\delta}$ over tokens. Since renormalization moves exactly this mass onto the selected experts, δ_i is also the total variation distance between the routing distribution and its selected

mixture, $\mathrm{TV}(\boldsymbol{p}_i, \boldsymbol{q}_i) = \frac{1}{2}\|\boldsymbol{p}_i - \boldsymbol{q}_i\|_1 = \delta_i$. The *number of contributors* is $K_{+,i} = |\mathcal{T}_i| \leq K$. Under top-2 selection, $K_{+,i} = 1$ whenever a selected expert has zero weight. Finally, we compare two assignment histograms over experts: the *nominal* load counts every selection, fillers included, and the *contributor* load counts only positive-weight selections, each assignment adding $1/K$. We report their coefficient of variation (CV) across experts (Shazeer et al., 2017), together with the standard deviation, since removing fillers also changes the mean. Appendix A.3 gives the aggregation details.

3 Experimental setup

Models and training. We train MoE language models at two scales with a Llama 3-style backbone (Grattafiori et al., 2024) in which every second feed-forward block is replaced by an MoE layer: a 300M model (8 layers, width 320, 4 MoE layers with 26 experts each) and a 1B model (9 layers, width 768, 4 MoE layers with 27 experts each). Every token selects $K = 2$ experts, following Mixtral's selection and weighted combination (Jiang et al., 2024), without capacity constraints or token dropping; the 1B model has 302M active parameters. The four maps share everything except π, including the balancing objective of Equation (5) with coefficient 0.01, averaged over MoE layers and evaluated on local microbatches. A dense model of matching depth and width at each scale provides a reference with a larger active-parameter budget. All models are trained on DCLM-baseline (Li et al., 2024) for 10B tokens (19,073 updates of $524{,}288$ tokens) with AdamW, cosine decay and bfloat16 mixed precision. Appendices A.1 and A.2 give the architecture details and the full training and evaluation recipe.

Evaluation. For the original matched comparison, we evaluate the final checkpoint of every run, which is also the one with the lowest DCLM validation loss. Predictive quality is measured by validation loss, mean accuracy over eight OLMES tasks and LAMBADA, and LAMBADA word perplexity (Gu et al., 2025; Paperno et al., 2016). For each OLMES task and model, we use the higher accuracy of the multiple-choice and completion formulations, following the benchmark's recommendation (Gu et al., 2025, §3.4).

Routing diagnostics use 61 documents from the WikiText-103 test subset of Paloma (Magnusson et al., 2024) at context $2{,}048$, giving $285{,}107$ targets per MoE layer. All routing statistics are averaged with equal weight over the four MoE layers. For every trained model we also record the router scores of all four layers on these documents, from which we compute the score spreads and top-2 gap distributions of §4.3.

4 Results

We first check that the MoEs are competent models (§4.1), then ask whether the sparsity of the map survives training (§4.2), whether the differences between trained routers come from the map or from the scores learned under it (§4.3), and what the sparse maps buy at inference time (§4.4). We report both scales and focus the discussion on the 1B models. The 300M models show the same qualitative routing patterns, with their training curves and load statistics in Appendix C.4.

4.1 Model quality

Table 1 reports predictive quality. All four MoEs are within 0.14 nats of validation loss and 3 points of mean accuracy of the dense reference, which has $3.3\times$ their active parameters, so the routers we study are those of reasonable language models rather than of a toy setting. Among the MoEs, while softmax has the lowest validation loss and the highest average numbers, we note that the margins are small across all maps: 0.008–0.012 in validation loss and 0.7–0.8 in average points.

Moreover, individual task leaders vary (normmax-2 leads ARC-Easy and PIQA, sparsemax leads SocialIQA and WinoGrande). These runs show no consistent quality advantage for the sparse maps. The rest of this section asks what they change in routing.

Table 1: Predictive quality of the 300M and 1B models: DCLM validation loss, perplexity and accuracies (%) on LAMBADA (LMB) and the eight OLMES tasks (ARC-Easy, ARC-Challenge, CommonsenseQA, HellaSwag, OpenBookQA, PIQA, SocialIQA, WinoGrande). The dense references have a larger active-parameter budget.

Model	Val.	LMB (ppl)	LMB	ARC-E	ARC-C	CSQA	HS	OBQA	PIQA	SQA	WG	Avg.
300M params.												
Dense	3.411	44.8	31.8	46.7	28.6	37.2	30.0	31.8	60.3	42.3	50.0	39.9
Softmax	**3.534**	104.2	24.3	42.7	27.6	36.5	29.4	29.6	61.9	41.4	50.8	**38.2**
1.5-entmax	3.536	**102.8**	24.2	43.0	27.1	34.6	29.4	27.8	60.0	41.7	51.7	37.7
Sparsemax	3.538	104.0	24.6	42.7	26.5	36.0	27.8	28.4	59.8	40.3	52.5	37.6
Normmax-2	3.536	105.9	23.4	42.8	25.5	33.3	29.6	25.2	60.5	40.9	50.5	36.9
1B params.												
Dense	3.103	18.8	41.6	56.0	29.3	47.3	39.0	33.8	63.9	42.7	51.3	45.0
Softmax	**3.232**	**28.9**	36.3	50.1	28.6	43.7	34.2	33.8	64.1	42.9	51.5	**42.8**
1.5-entmax	3.240	29.5	35.4	50.3	28.3	42.1	34.6	30.4	63.5	41.9	51.3	42.0
Sparsemax	3.242	30.3	35.1	50.1	28.3	41.9	32.6	30.8	63.7	43.2	52.1	42.0
Normmax-2	3.244	32.1	34.8	51.8	27.5	41.2	34.9	30.6	64.8	41.6	51.9	42.1

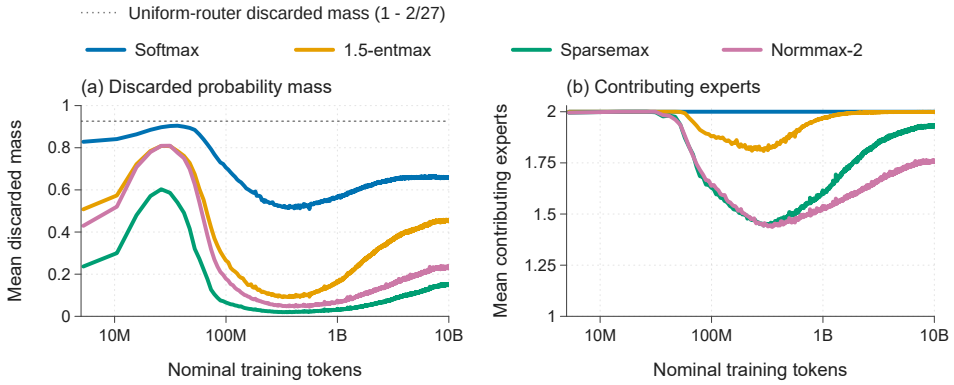

Figure 2: Discarded probability mass (a) and number of contributing experts per token (b) during training of the 1B MoE models, measured on training batches. The horizontal axis is the number of training tokens (log scale). All maps sparsify early in training; entmax then returns to two contributors while sparsemax and normmax do not. The 300M curves are provided in Appendix C.4.

4.2 Does the sparsity of the map survive training?

Figure 2 shows the two routing quantities during training. All three sparse maps drop contributors early on, but entmax recovers to almost exactly two contributors per token, whereas sparsemax and normmax settle at 1.93 and 1.79. At the end of training, reported in Table 2, 1.5-entmax discards 30% less probability mass than softmax ($\bar{\delta} = 0.462$ versus 0.661) with a contributor count that is still 2.000 to three decimals, sparsemax retains the most mass ($\bar{\delta} = 0.153$), and normmax uses the fewest contributors ($\overline{K}_+ = 1.787$). The two quantities are governed by different regimes of the support size s, the number of experts with positive probability. Under top-2 selection,

$$\mathbb{E}[\delta] = \Pr(s \geq 3)\, \mathbb{E}[\delta \mid s \geq 3], \qquad \mathbb{E}[K_+] = 2 - \Pr(s = 1), \tag{7}$$

since only a support larger than 2 can lose mass to truncation, and only a singleton support can reduce participation below 2. Entmax sits almost entirely in the first regime: 99.9% of its states have $s \geq 3$, so it retains more mass than softmax while keeping both experts. Normmax sits in the second: 21.3% of its states are singletons, versus 6.9% for sparsemax, which instead has more states with exactly two positive probabilities (14.1% versus 7.6%), retaining all mass with two contributors. Sparsity in the routing distribution and sparsity in participation are therefore different outcomes, and a map can deliver one without the other. Results for the 300M models, provided in Appendix C.4, share the same findings.

Table 2: Routing statistics of the trained models. Besides discarded mass $\bar{\delta}$ and activated experts $\overline{K}_+$, we also report the fraction of token–layer states with singleton support and with support of exactly two experts, the discarded mass conditional on support larger than two, the coefficient of variation of the expert loads counting every selection or only positive-weight selections (Contrib. CV), and the standard deviation of the contributor loads (Contrib. SD). Lower CV and SD indicate more even expert loads. Bold marks the lowest value at each scale, including ties at the displayed precision. Rows under softmax-based load balancing means that we used softmax probabilities for the auxiliary balancing loss.

Map	$\bar{\delta}$	$\overline{K}_+$	s=1 %	s=2 %	$\mathbb{E}[\delta \mid s \geq 3]$	Nominal CV ↓	Contrib. CV ↓	Contrib. SD ↓
300M params.								
Softmax	0.566	2.000	0.000	0.000	0.566	0.404	0.404	0.016
1.5-entmax	0.311	1.997	0.295	0.388	0.314	0.400	**0.400**	0.015
Sparsemax	0.099	1.904	9.598	28.174	0.160	0.434	0.450	0.017
Normmax-2	0.146	1.769	23.144	16.228	0.241	**0.367**	0.404	**0.014**
1B params.								
Softmax	0.661	2.000	0.000	0.000	0.661	0.339	0.339	**0.013**
1.5-entmax	0.462	2.000	0.048	0.095	0.462	**0.338**	**0.338**	**0.013**
Sparsemax	0.153	1.931	6.883	14.079	0.194	0.414	0.424	0.015
Normmax-2	0.250	1.787	21.256	7.587	0.351	0.341	0.388	**0.013**
300M params. with softmax-based load balancing								
1.5-entmax	0.292	1.999	0.079	0.245	0.293	0.485	0.485	0.019
Sparsemax	0.078	1.921	7.941	30.019	0.126	0.643	0.668	0.025

4.3 Is it the map, or the scores learned under it?

The trained models differ both in the map and in the scores that the map receives, and the two are easy to separate because each map has a fixed sparsification threshold (see §2.1). Concretely, a selected expert is dropped exactly when the top-2 score gap is at least 2 for 1.5-entmax and at least 1 for sparsemax and normmax-2. The singleton rate of a trained model is therefore nothing but the fraction of its learned gaps that reach its map's threshold. Figure 3 shows these gap distributions for the 1B models, with both thresholds marked in every panel. The numerical spreads and quantiles at both scales are retained in Table 7 in Appendix C.2.

The learned scores turn out to matter as much as the map. All three sparse routers learned markedly smaller score spreads than the softmax router, with an RMS between 0.32 and 0.38 against 0.86. Under a sparse map, this scale determines how often a score gap reaches the singleton threshold. For 1.5-entmax the consequence is decisive. Its 99th-percentile gap is 1.45, well below the threshold of 2, so the trained router produces a singleton in only 0.05% of states, even though 7.8% of the softmax router's gaps reach that threshold. Sparsemax and normmax-2 tell a complementary story. The two maps share the threshold 1, so on any given score vector they always agree on whether a token keeps one or two contributors, and yet the trained models differ by 0.14 contributors per token. The difference can only come from the scores the two routers learned, and it does: the median gap of the normmax-2 router is 0.61 against 0.40 for the sparsemax router, and its gaps reach 1 three times as often.

Taken together, 1.5-entmax keeps both experts because its threshold is the highest of the three maps and because its router learned scores that stay under it, while sparsemax and normmax-2 differ only in what their routers learned. Participation is thus set jointly by the map's threshold and by the learned score distribution, and the router adapts the latter to the former.

Depth. The layer averages hide a strong depth dependence. In Table 6 in the appendix, we note that normmax-2 routes 54% of tokens to a single expert in the first MoE layer but only 7% in the last. Sparsemax shows the same pattern with smaller magnitude (19% to 4%), and 1.5-entmax has essentially no singletons anywhere. Early layers learn the sparsest routers, which suggests that the useful degree of participation is itself depth-dependent and provides a further reason not to set K arbitrarily.

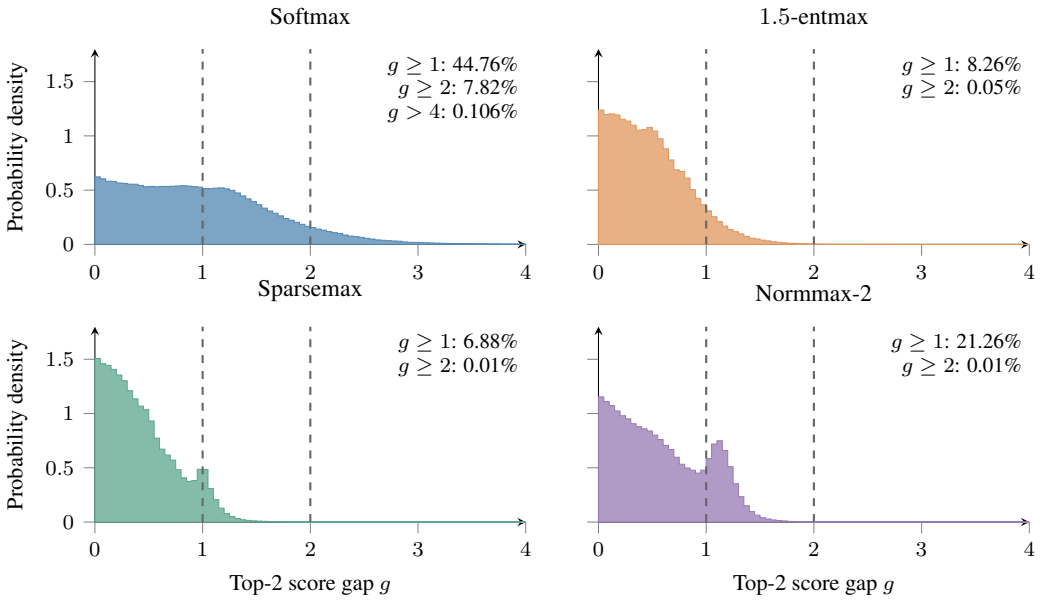

Figure 3: Top-2 score gaps of the trained 1B routers. Each panel pools 285,107 targets from each of the four MoE layers with equal weight. Gray dashed lines mark the singleton thresholds $g = 1$ for sparsemax and normmax-2 and $g = 2$ for 1.5-entmax. Softmax has no finite singleton threshold. Panel annotations give the fractions at or above each threshold, computed on the full sample, including density beyond $g = 4$ for softmax. The peaks near $g = 1$ for sparsemax and normmax-2 are consistent with gradient saturation: for $g > 1$, these maps are locally constant and gradients through the routing probabilities vanish.

Table 3: Robustness to the inference-time budget. NLL (nats/target) on the retained DCLM diagnostic sample when a model trained with K=2 is run, with frozen weights, with $K \in \{1, 2, 4, 8\}$ selected experts; Δ is the increase from K=2 to K=8. Retention and contributor counts for every setting are in Appendix C.3.

	300M					1B				
Map	K=1	K=2	K=4	K=8	Δ	K=1	K=2	K=4	K=8	Δ
Softmax	3.703	3.560	3.623	3.913	+0.354	3.360	3.254	3.328	3.835	+0.582
1.5-entmax	3.711	3.562	3.598	3.693	+0.131	3.363	3.261	3.306	3.512	+0.250
Sparsemax	3.715	3.565	3.566	3.568	**+0.004**	3.365	3.263	3.274	3.284	**+0.021**
Normmax-2	3.703	3.562	3.574	3.586	+0.024	3.368	3.266	3.286	3.329	+0.064

4.4 What the sparse maps buy at inference time

Although they do not improve quality, the sparse maps make the trained model much less sensitive to selecting more experts at inference. We carry that analysis in Table 3. When the models trained with K=2 are run with K=8 and frozen weights, softmax loses 0.58 nats, because it is forced to spread weight over six experts it never learned to combine. Sparsemax loses only 0.02 nats, since it keeps assigning zero weight to most of the extra slots and uses 4.3 experts on average at K=8, and normmax-2 and 1.5-entmax fall in between. The robustness is one-sided: reducing the budget to K=1 costs every map about 0.10 nats. For every map the loss is lowest at the trained budget, so selecting more experts to retain more mass does not help either, which confirms that retention, participation, and quality are three distinct quantities.

Load balance. Counting contributors rather than nominal selections also changes the picture of expert balance, as the last three columns of Table 2 show. The 1.5-entmax router has the lowest load CV under both conventions (0.338), and because almost all of its selections carry positive

weight, excluding the fillers changes nothing for it. Normmax-2 is different: its nominal CV is only slightly above softmax's (0.341 against 0.339), but once the fillers are excluded its contributor CV rises to 0.388, so its load is less balanced than the nominal histogram suggests, and sparsemax is the least balanced under either convention. The balancing objective is one possible source of these differences. Because it uses the full distribution, an expert with zero probability receives no direct balancing gradient under a sparse map, and changing the score scale also changes the probabilities seen by this objective.

To examine the balancing mechanism, we trained 300M 1.5-entmax and sparsemax routers with $\mathrm{softmax}(\boldsymbol{z})$ used only in the auxiliary loss. Routing and expert combination still use their respective sparse maps. For 1.5-entmax, nominal and contributor CV both rise from 0.400 to 0.485, while centered score RMS rises from 0.517 to 0.603 and the singleton fraction falls from 0.295% to 0.079%. Sparsemax shows the same pattern. Nominal CV rises from 0.434 to 0.643 and contributor CV from 0.450 to 0.668 (Table 2), while centered score RMS rises from 0.413 to 0.442 and the singleton fraction falls from 9.598% to 7.941%. Since higher CV means a less even distribution of traffic across experts, the dense auxiliary probability does not improve load balance in either run. Figure 6 shows why fewer tokens become singletons: the entmax tail near $g = 2$ thins, and the sparsemax peak just above $g = 1$ becomes smaller. More tokens therefore keep both selected experts, but their contributions are distributed less evenly across the expert pool. Per-token participation and load balance move in opposite directions in both runs, which does not support the simple explanation that sparsemax's imbalance comes solely from zero-probability experts receiving no direct balancing gradient.

5 Related work

Sparse maps and differentiable gates in MoE routers. Softmax followed by top-K remains the dominant router (Shazeer et al., 2017; Fedus et al., 2022; Jiang et al., 2024), with sigmoid scores renormalized among the selected experts as the main production alternative (DeepSeek-AI, 2024). Sparse probability maps have been used as routers before. For example, Multilinear MoE compares entmax with softmax over factorized experts (Oldfield et al., 2024), DUAL studies sparsemax under routing constraints (Thaman, 2025), and ReMoE replaces the map by rectified gates with a separate sparsity controller (Wang et al., 2025). Differentiable relaxations of the selection step itself include DSelect-k, a smooth gate with explicit cardinality control (Hazimeh et al., 2021), the sparse top-k operators of Sander et al. (2023), Soft MoE, which replaces discrete routing by soft slot assignment (Puigcerver et al., 2024), and, more recently, SoftMoE, a truncated LapSum relaxation with learned per-layer budgets (Zasada et al., 2026), and ProbMoE, which samples cardinality-constrained subsets and differentiates through inclusion marginals (Zhao et al., 2026). These works propose new routers and evaluate them end-to-end, whereas we keep the router fixed to top-2 and ask what a sparse map delivers once the router has been trained with it, and how the score distribution it learns interacts with the map's sparsification threshold (§4.3).

Load balancing and routing geometry. Balancing objectives range from the Switch-style loss we use (Fedus et al., 2022) to balanced assignment (Lewis et al., 2021; Zhou et al., 2022), loss-free bias adjustment (Wang et al., 2024) and ϕ-Balancing, which targets population-level balance through a convex potential and mirror descent (Chen et al., 2026). Yoon et al. (2026) show that the task gradient reaches only the selected experts and that balancing provides no token-level counterfactual signal, and Huu et al. (2026) analyze the switching surfaces of top-K routing and propose smoothing near them. Our observation that the balancing objective interacts with the map through the full distribution (§4.4) is complementary: it concerns which experts the balancing gradient can reach under a sparse map.

Sparse probability transformations. Sparsemax (Martins & Astudillo, 2016) and α-entmax (Peters et al., 2019) are regularized argmax maps in the Fenchel–Young framework of Blondel et al. (2020), which also covers the norm entropies behind normmax (Santos et al., 2024; 2025). Their sparsity has been exploited in attention, where the support size is known to depend on the scale of the scores: Correia et al. (2019) learn α per head, and Vasylenko et al. (2026) learn a temperature so that α-entmax attention interpolates between sparse and dense regimes as the context grows. Our results transfer this observation to routing, where the score scale learned by the router, together with

the map's threshold, sets participation. This suggests that learned scales or shapes are the right knob for adaptive MoE routing.

6 CONCLUSIONS

We studied whether the sparsity of a probability map survives training in a top-2 MoE router. Across matched 300M and 1B models, the answer is that it survives only partly, and never as a property of the map alone. The 1.5-entmax router retains more probability mass than softmax while keeping both selected experts, the sparsemax router retains the most mass, and the normmax-2 router uses the fewest contributors. These behaviors are set jointly by the map's sparsification threshold and by the score distribution the router learns under it. The 1.5-entmax router keeps both experts because its learned scores have half the spread of softmax's, which keeps its top-2 gaps below the threshold, whereas sparsemax and normmax-2, which sparsify at the same gap, differ only because their routers learned different gap distributions. Retention, participation and predictive quality also turn out to be three distinct quantities: none of the sparse maps improves validation loss over softmax, yet all of them make the trained model less sensitive than softmax to increasing the inference-time budget.

Our results also say where a sparse router should be designed, rather than merely swapped in. Because the router co-adapts its score scale to the map, that scale is a modelling choice that we currently leave to the optimizer, and it would be better learned explicitly, as is already done for sparse attention (Correia et al., 2019; Vasylenko et al., 2026). A router whose scale is under control could then be trusted to choose its own budget: the sparse maps already assign zero weight to most of the extra slots when K is raised, so selecting every expert in the support instead of a fixed top-K, capped by an at-most-k operator, is a small step from what the trained models already do. The balancing objective is another design choice to test: its gradients depend on the map, so changing the probabilities used for balancing may alter the score scales and expert loads learned during training. We leave these directions to future work.

Limitations. Our matched comparison uses a single training seed per configuration, so the small quality differences between maps (at most 0.012 nats of validation loss) do not establish a ranking. We test two members of the entmax family ($\alpha = 1.5$ and sparsemax, $\alpha = 2$), one normmax ($\gamma = 2$) and one balancing objective, whose probabilities we vary in a single 300M ablation. Contributor counts measure expert participation, not compute. As such, whether zero-weight experts are skipped, and what that saves, depends on the dispatch implementation, which we do not measure. We did not verify that the diagnostic samples are disjoint from the training data. Finally, the score-scale analysis is descriptive: it relates each router's singleton rate to its learned gap distribution but does not identify the training mechanism that produced the smaller scales.

AI USE STATEMENT

In this work, we used generative AI tools to assist in writing the proofs and fixing notation issues. We also used generative AI tools for summarizing and analyzing existing literature, searching for information, discovering literature beyond our manual search, and editing and polishing the paper for readability. We used generative AI tools to write and edit software code. All uses of AI tools were accompanied by a careful review process. In particular, we manually checked the proofs and made them more accessible, and LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Appendix A gives the architecture shapes and parameter counts (A.1), the full training recipe including optimizer settings, schedule, batch composition and the balancing objective (A.2), and the evaluation protocol, including the quality metrics, the exact definition of every routing diagnostic and the inference-time budget experiment (A.3); Appendix B derives the singleton thresholds and defines the score-scale analysis, and Appendix C collects the per-layer and 300M results. In the

original matched comparison, all models at a given scale share the same data, seed and hyperparameters and differ only in the probability map. We will release the training and evaluation code, the configuration files of every run, the recorded router scores of every trained model used in §4.3, and a numerical supplement with the routing, score-scale and quality summaries at both scales, so that every table and figure in the paper can be regenerated.

REFERENCES

Mathieu Blondel, André F. T. Martins, and Vlad Niculae. Learning with Fenchel-Young losses. *Journal of Machine Learning Research*, 21(35):1–69, 2020. URL https://jmlr.org/papers/v21/19-021.html.

Lizhang Chen, Jonathan Li, Qi Wang, Runlong Liao, Shuozhe Li, Chen Liang, Ni Lao, and Qiang Liu. ϕ-balancing for mixture-of-experts training. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://openreview.net/forum?id=DZbzIOguz4.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. Adaptively sparse transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2174–2184. Association for Computational Linguistics, 2019.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. URL https://arxiv.org/abs/2412.19437.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. OLMES: A standard for language model evaluations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5020–5048, 2025. doi: 10.18653/v1/2025.findings-naacl.282. URL https://aclanthology.org/2025.findings-naacl.282/.

Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder, Lichan Hong, and Ed H. Chi. DSelect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Tho Tran Huu, Huu-Tuan Nguyen, Thien-Hai Nguyen, Nhat-Tri Ho, Viet-Hoang Tran, Tho Quan, and Tan Minh Nguyen. Geometric and stochastic analysis of discontinuities in sparse mixture-of-experts. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://openreview.net/forum?id=AyrNR1ExN5.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024. URL https://arxiv.org/abs/2401.04088.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. BASE layers: Simplifying training of large, sparse models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6265–6274, 2021. URL https://proceedings.mlr.press/v139/lewis21a.html.

Jeffrey Li, Alex Fang, Georgios Smyrnis, et al. DataComp-LM: In search of the next generation of training sets for language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 14200–14282, 2024.

URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/19e4ea30dded58259665db375885e412-Abstract-Datasets_and_Benchmarks_Track.html.

Ian Magnusson, Akshita Bhagia, Valentin Hofmann, Luca Soldaini, Ananya Harsh Jha, Oyvind Tafjord, Dustin Schwenk, Evan Pete Walsh, Yanai Elazar, Kyle Lo, Dirk Groeneveld, Iz Beltagy, Hannaneh Hajishirzi, Noah A. Smith, Kyle Richardson, and Jesse Dodge. Paloma: A benchmark for evaluating language model fit. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://papers.nips.cc/paper_files/paper/2024/hash/760b2d94398aa61468aa3bc11506d9ea-Abstract-Datasets_and_Benchmarks_Track.html.

André F. T. Martins and Ramón F. Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1614–1623, 2016. URL https://proceedings.mlr.press/v48/martins16.html.

James Oldfield, Markos Georgopoulos, Grigorios G. Chrysos, Christos Tzelepis, Yannis Panagakis, Mihalis Nicolaou, Jiankang Deng, and Ioannis Patras. Multilinear mixture of experts: Scalable expert specialization through factorization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 53022–53063, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5eeb693f46d753e5fe24c97212c22bd2-Paper-Conference.pdf.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Ben Peters, Vlad Niculae, and André F. T. Martins. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1504–1519, 2019. doi: 10.18653/v1/P19-1146. URL https://aclanthology.org/P19-1146/.

Joan Puigcerver, Carlos Riquelme, Basil Mustafa, and Neil Houlsby. From sparse to soft mixtures of experts. In *The Twelfth International Conference on Learning Representations*, 2024.

Michael Eli Sander, Joan Puigcerver, Josip Djolonga, Gabriel Peyré, and Mathieu Blondel. Fast, differentiable and sparse top-k: a convex analysis perspective. In *International Conference on Machine Learning*, pp. 29919–29936. PMLR, 2023.

Saul Santos, Vlad Niculae, Daniel C. McNamee, and André F. T. Martins. Sparse and structured hopfield networks. In *Forty-first International Conference on Machine Learning*, 2024.

Saul Santos, Vlad Niculae, Daniel McNamee, and André F. T. Martins. Hopfield-Fenchel-Young networks: A unified framework for associative memory retrieval. *Journal of Machine Learning Research*, 26(265):1–51, 2025. URL https://jmlr.org/papers/v26/24-1961.html.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Kushal Thaman. One must imagine experts happy: Rebalancing neural routers via constrained optimization. In *ICLR 2025 Workshop on Sparsity in LLMs*, 2025. URL https://iclr.cc/virtual/2025/33506.

Pavlo Vasylenko, Hugo Pitorro, André F. T. Martins, and Marcos V. Treviso. Long-context generalization with sparse attention. In *The Fourteenth International Conference on Learning Representations*, 2026.

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. URL https://arxiv.org/abs/2408.15664v1.

Ziteng Wang, Jun Zhu, and Jianfei Chen. ReMoE: Fully differentiable mixture-of-experts with ReLU routing. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/94dc604e115237a7f4a758b3146cd976-Abstract-Conference.html.

Youngsik Yoon, Siwei Wang, Wei Chen, and Jungseul Ok. When Are Experts Misrouted? Counterfactual Routing Analysis in Mixture-of-Experts Language Models, 2026. URL https://arxiv.org/abs/2605.07260v1.

Mikołaj Zasada, Łukasz Struski, Jacek Tabor, and Marcin Kurdziel. SoftMoE: Soft differentiable routing for mixture-of-experts in LLMs. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://openreview.net/forum?id=vGTFOp3jLO.

Heng Zhao, Zilei Shao, Guy Van den Broeck, and Zhe Zeng. ProbMoE: Differentiable Probabilistic Routing for Mixture-of-Experts, 2026. URL https://arxiv.org/abs/2606.01509.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V. Le, and James Laudon. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/2f00ecd787b432c1d36f3de9800728eb-Abstract-Conference.html.

Table 4: Architectures and parameter counts. FF width is the hidden width of each SwiGLU branch; MoE layers replace every second feed-forward block. Nominal active parameters include the full embedding tensors and count two experts per MoE layer, even when one combine weight is zero.

			Heads	FF	MoE		Parameters		
Model	Layers	Width	Q/KV	width	layers	Experts	Total	Active	Router
Dense 283M	10	768	12/3	3072	–	–	283M	283M	–
MoE 300M	8	320	8/2	2048	4	26	297M	108M	33K
Dense 1B	18	1536	24/6	6144	–	–	1.01B	1.01B	–
MoE 1B	9	768	12/3	3072	4	27	1.01B	302M	83K

A Experimental details

A.1 Architecture and parameter accounting

All models use an untied input embedding and language-model head with vocabulary size $128{,}256$, pre-RMSNorm blocks and a final RMSNorm, grouped-query attention, rotary embeddings with base $500{,}000$, and bias-free SwiGLU feed-forward blocks (Grattafiori et al., 2024). The configured maximum context is $8{,}192$; training and the routing diagnostics use $2{,}048$ tokens. The norm epsilon is 10^{-5}, dropout is zero and the initialization standard deviation is 0.02. Table 4 gives the shapes and parameter counts.

With L layers, L_{M} MoE layers, width d, SwiGLU width h, vocabulary V and $d_{\mathrm{KV}} = d\, n_{\mathrm{KV}}/n_{\mathrm{Q}}$, the parameter count is

$$P_{\mathrm{total}} = 2Vd + L(2d^2 + 2dd_{\mathrm{KV}} + 2d) + d + 3dh[(L - L_{\mathrm{M}}) + L_{\mathrm{M}}n] + L_{\mathrm{M}}dn,$$

where the last term counts the router matrices. Replacing n by $K = 2$ in the expert term gives the nominal active count, which includes the full embedding tensors and counts two experts per MoE layer even when one combine weight is zero. The four maps add no parameters at fixed α and γ. Nominal active parameters do not equate training or inference FLOPs across dense and MoE models.

A.2 Training recipe

Training uses DCLM-baseline (Li et al., 2024) with the Meta-Llama-3 tokenizer and seed 42. AdamW has peak learning rate 3×10^{-4}, minimum rate 3×10^{-6}, betas $(0.9, 0.95)$, weight decay 0.1 and gradient clipping at norm 1.0; the learning rate warms up linearly for 500 updates and then follows cosine decay to the minimum at update $19{,}073$. Mixed precision is bfloat16. Every run uses four GPUs with batch size 4 per process and 16 gradient-accumulation microbatches, i.e. 256 sequences or $524{,}288$ tokens per optimizer update, and $19{,}073$ updates for the nominal 10B-token budget. The MoE runs use the Switch-style balancing loss of Equation (5), evaluated on each local microbatch of $8{,}192$ tokens with coefficient 0.01 and averaged over the four MoE layers; there is no router z-loss, and the dense references have no balancing loss. Validation cross-entropy on a 10M-token cache is logged every 500 updates and at the final update; for all ten models the minimum is at the final update, which is the checkpoint we evaluate.

A.3 Evaluation protocol

Quality metrics. Validation loss and the diagnostic-domain losses exclude the balancing loss; every unmasked $2{,}048$-token chunk contributes $2{,}047$ next-token targets. The eight OLMES tasks are ARC-Challenge, ARC-Easy, CommonsenseQA, HellaSwag, OpenBookQA, PIQA, SocialIQA and WinoGrande (Gu et al., 2025), each evaluated in the multiple-choice and completion formulations with the OLMES scoring rules. We report the higher of the two accuracies per model and task, as recommended in Gu et al. (2025, §3.4). LAMBADA (Paperno et al., 2016) counts a prediction as correct only if every token of the final word is predicted correctly, and its word perplexity uses the full target-word log probability. The mean accuracy weights the eight OLMES tasks and LAMBADA equally. Evaluation runs in FP32 with autocast and TF32 disabled. Table 1 in the main text gives the per-task results at both scales.

Diagnostic samples. Domain losses use 489 packed 2,048-token blocks of retained DCLM (1,000,983 scored targets) and the 61 Paloma WikiText-103 test documents (285,107 targets). The inference-budget evaluation uses the retained DCLM sample. All Paloma diagnostics reported here use the test sample at context 2,048.

Routing diagnostics. For every MoE layer and token batch, the nominal load assigns $1/K$ to each selected index, fillers included, and the contributor load only to selections with strictly positive weight; histograms are divided by the number of routed tokens, and the CV is the population standard deviation divided by the mean, averaged over layers. The router computes scores and probabilities in FP32 under mixed precision and casts the normalized combine weights p_{ij}/m_i to the expert-output dtype. The training curves of Figures 2 and 4 use every logged training batch at a cadence of 10 updates. The checkpoint diagnostic runs every trained model in FP32 on the 61 Paloma test documents with identical inputs, contexts and target order, giving 285,107 targets per layer and 1,140,428 target–layer events per model; the recorded router scores of all four layers are the data from which the spreads and gap quantiles of §4.3 and Appendix B.4 are computed. Targets and layers are dependent observations from one checkpoint per map, not training replications.

Inference-time budget. For the experiment of §4.4 we take each trained model with frozen weights and run it with $K \in \{1, 2, 4, 8\}$ selected experts on the retained DCLM diagnostic sample, keeping the map and renormalizing the selected probabilities as in Equation (1). Besides the loss, we record the retained mass $1 - \delta_i$ and the number of contributors $K_{+,i}$ of every target–layer event and average them over targets and layers; Table 3 in the main text reports the loss and Table 9 in Appendix C the full set.

B Singleton thresholds and the score-scale analysis

This appendix derives the singleton thresholds quoted in §2.1 from the optimality conditions of each map, records the closed forms that follow from them, and then defines the score-scale quantities used in §4.3. Throughout, the token index is suppressed, $\boldsymbol{z} \in \mathbb{R}^n$ is a score vector with sorted entries $z_{(1)} \geq z_{(2)} \geq \cdots \geq z_{(n)}$, $g = z_{(1)} - z_{(2)}$ is its top-2 gap, and $\mathcal{I} = \{j : p_j > 0\}$ is the support of $\boldsymbol{p} = \pi(\boldsymbol{z})$, with size $s = |\mathcal{I}|$.

B.1 Optimality conditions

Every map we use solves the regularized argmax problem of Equation (2),

$$\boldsymbol{p} = \operatorname*{arg\,max}_{\boldsymbol{p} \in \triangle^n} \; \boldsymbol{p}^\top \boldsymbol{z} + H(\boldsymbol{p}), \tag{8}$$

for a concave entropy H. Introducing a multiplier $\theta \in \mathbb{R}$ for the constraint $\mathbf{1}^\top \boldsymbol{p} = 1$ and multipliers $\mu_j \geq 0$ for the constraints $p_j \geq 0$, the Lagrangian is

$$\mathcal{L}(\boldsymbol{p}, \theta, \boldsymbol{\mu}) = \boldsymbol{p}^\top \boldsymbol{z} + H(\boldsymbol{p}) - \theta\,(\mathbf{1}^\top \boldsymbol{p} - 1) + \boldsymbol{\mu}^\top \boldsymbol{p}, \tag{9}$$

and the Karush–Kuhn–Tucker (KKT) conditions read, for every expert j,

$$z_j + \partial_j H(\boldsymbol{p}) - \theta + \mu_j = 0, \qquad \mu_j \geq 0, \qquad \mu_j p_j = 0, \qquad \boldsymbol{p} \in \triangle^n. \tag{10}$$

Because the objective is concave and the simplex is convex, these conditions are necessary and sufficient, and for the three entropies below the maximizer is unique (Blondel et al., 2020; Santos et al., 2025). Splitting Equation (10) by whether p_j is positive gives the two conditions we use repeatedly:

$$z_j + \partial_j H(\boldsymbol{p}) = \theta \quad \text{for } j \in \mathcal{I}, \qquad\qquad z_j + \partial_j H(\boldsymbol{p}) \leq \theta \quad \text{for } j \notin \mathcal{I}. \tag{11}$$

The first says that on the support the score, shifted by the entropy gradient, is constant; the second says that an expert is left out exactly when its shifted score does not reach that constant.

Softmax. With the Shannon entropy $H^{\mathrm{S}}(\boldsymbol{p}) = -\sum_j p_j \log p_j$ we have $\partial_j H^{\mathrm{S}}(\boldsymbol{p}) = -\log p_j - 1$, which diverges to $+\infty$ as $p_j \to 0$. The inactive condition in Equation (11) can therefore never hold, so every expert is on the support, and the active condition gives $\log p_j = z_j - 1 - \theta$, i.e. $p_j = e^{z_j} / \sum_k e^{z_k}$ after choosing θ so that the probabilities sum to one.

α-entmax. The Tsallis entropy $H^{\mathrm{Ts}}_\alpha(\boldsymbol{p}) = \big(1 - \sum_j p_j^\alpha\big)/(\alpha(\alpha-1))$ has gradient

$$\partial_j H^{\mathrm{Ts}}_\alpha(\boldsymbol{p}) = -\frac{\alpha p_j^{\alpha-1}}{\alpha(\alpha-1)} = -\frac{p_j^{\alpha-1}}{\alpha-1}, \tag{12}$$

which is finite at $p_j = 0$ for every $\alpha > 1$, so experts can leave the support. Substituting in Equation (11), the active condition becomes $z_j - p_j^{\alpha-1}/(\alpha-1) = \theta$, that is

$$p_j^{\alpha-1} = (\alpha-1)(z_j - \theta) \quad \Longleftrightarrow \quad p_j = \big[(\alpha-1)z_j - (\alpha-1)\theta\big]^{1/(\alpha-1)} \quad \text{for } j \in \mathcal{I}, \tag{13}$$

and the inactive condition becomes $z_j \le \theta$. Both are captured by the single expression

$$p_j = \big[(\alpha-1)z_j - \tau\big]_+^{1/(\alpha-1)}, \qquad \tau = (\alpha-1)\theta, \tag{14}$$

which is Equation (3): the bracket is positive exactly for the experts with $z_j > \theta$, and those are the support. The threshold τ is fixed by the simplex constraint,

$$\sum_j \big[(\alpha-1)z_j - \tau\big]_+^{1/(\alpha-1)} = 1. \tag{15}$$

The left-hand side is continuous and strictly decreasing in τ wherever it is positive, tends to $+\infty$ as $\tau \to -\infty$ and vanishes for $\tau \ge (\alpha-1)z_{(1)}$, so Equation (15) has exactly one root, which can always be found by bisection. For the two values of α we use, the root also has a sorting-based closed form. If the support is the top k experts, then for sparsemax ($\alpha = 2$) Equation (15) reads $\sum_{j\le k}(z_{(j)} - \tau) = 1$, so

$$\tau_k = \frac{1}{k}\Big(\sum_{j\le k} z_{(j)} - 1\Big), \tag{16}$$

and for 1.5-entmax ($\alpha = 3/2$, so that $(\alpha-1) = 1/2$ and $1/(\alpha-1) = 2$) it reads $\sum_{j\le k}(z_{(j)}/2 - \tau)^2 = 1$. Writing m_k and v_k for the mean and the variance of the k halved scores $z_{(1)}/2, \dots, z_{(k)}/2$, we have $\sum_{j\le k}(z_{(j)}/2 - \tau)^2 = k\big[(m_k - \tau)^2 + v_k\big]$, and the root that keeps the probabilities positive is

$$\tau_k = m_k - \sqrt{\frac{1 - k\,v_k}{k}}. \tag{17}$$

In both cases the support size is the unique k for which the candidate τ_k is consistent with Equation (11), i.e. the k largest bracket arguments are positive and the remaining ones are not; after one sort this is a linear scan over k (Martins & Astudillo, 2016; Peters et al., 2019). In our implementation 1.5-entmax uses this exact algorithm, and sparsemax and normmax-2 solve their threshold equations by bisection.

γ-normmax. The norm entropy $H^{\mathrm{N}}_\gamma(\boldsymbol{p}) = 1 - \|\boldsymbol{p}\|_\gamma$ has gradient

$$\partial_j H^{\mathrm{N}}_\gamma(\boldsymbol{p}) = -\frac{p_j^{\gamma-1}}{\|\boldsymbol{p}\|_\gamma^{\gamma-1}} = -\Big(\frac{p_j}{R}\Big)^{\gamma-1}, \qquad R = \|\boldsymbol{p}\|_\gamma, \tag{18}$$

which again vanishes at $p_j = 0$ for $\gamma > 1$. Equation (11) therefore gives $(p_j/R)^{\gamma-1} = z_j - \theta$ on the support and $z_j \le \theta$ off it, so

$$p_j = R\,[z_j - \theta]_+^{1/(\gamma-1)}. \tag{19}$$

Summing Equation (19) over j and using $\mathbf{1}^\top \boldsymbol{p} = 1$ identifies $R = 1/\sum_k [z_k - \theta]_+^{1/(\gamma-1)}$, which turns Equation (19) into the ratio form of Equation (4) with $\tau = \theta$. Substituting Equation (19) back into the definition $R^\gamma = \sum_j p_j^\gamma$ and dividing by R^γ gives the normalization $\sum_j [z_j - \theta]_+^{\gamma/(\gamma-1)} = 1$ stated in Equation (4). As for entmax, the left-hand side of this normalization is strictly decreasing in θ, so the threshold is unique and can be found by bisection; for $\gamma = 2$ it reads $\sum_j (z_j - \theta)_+^2 = 1$ and admits the same sorted closed form as Equation (17), with the raw scores in place of the halved ones.

Order preservation. In each of the three cases, the active condition has the form $z_j - \theta = \phi(p_j)$ for a strictly increasing function ϕ with $\phi(0) = 0$ (namely $\phi(p) = p^{\alpha-1}/(\alpha-1)$ for entmax and $\phi(p) = (p/R)^{\gamma-1}$ for normmax), and the inactive condition is $z_j - \theta \le 0$. Hence p_j is a nondecreasing function of z_j: larger scores receive larger probabilities, equal scores receive equal probabilities, and every expert with $p_j > 0$ has a larger score than every expert with $p_j = 0$. Consequently the top-K experts by probability are the top-K by score (ties broken by the same rule), which is why the selection $\mathcal{S}$ of Equation (1) and all the statements below can be phrased in terms of scores alone.

B.2 Singleton thresholds

Proposition 1 (Singleton thresholds)**.** *Let $g = z_{(1)} - z_{(2)}$. The support of α-entmax is the singleton $\{(1)\}$ if and only if $g \ge 1/(\alpha-1)$; the support of γ-normmax is $\{(1)\}$ if and only if $g \ge 1$, for every $\gamma > 1$; and the support of softmax is never a singleton. In particular the thresholds are 2 for 1.5-entmax and 1 for both sparsemax and normmax-2.*

Proof. For softmax the claim was shown above: all probabilities are positive at finite scores. For the other two maps we test the candidate $\boldsymbol{p} = \boldsymbol{e}_{(1)}$ against the KKT conditions (11); since the conditions are sufficient, the candidate is the solution whenever it satisfies them, and since the solution is unique, it is not the solution whenever it violates them.

Entmax. At $\boldsymbol{p} = \boldsymbol{e}_{(1)}$ the active condition for expert (1) reads $1^{\alpha-1} = (\alpha-1)(z_{(1)} - \theta)$, which fixes the multiplier,

$$\theta = z_{(1)} - \frac{1}{\alpha - 1}. \tag{20}$$

The inactive condition for every other expert j is $z_j \le \theta$, that is $z_{(1)} - z_j \ge 1/(\alpha-1)$. Because $z_{(2)}$ is the largest of the remaining scores, all these inequalities hold if and only if the one for $j = (2)$ holds, i.e. if and only if $g \ge 1/(\alpha-1)$. At $\alpha = 3/2$ this is $g \ge 2$ and at $\alpha = 2$ it is $g \ge 1$.

Normmax. At $\boldsymbol{p} = \boldsymbol{e}_{(1)}$ the norm is $R = \|\boldsymbol{e}_{(1)}\|_\gamma = 1$, so the active condition for expert (1) reads $(1/1)^{\gamma-1} = z_{(1)} - \theta$, i.e. $\theta = z_{(1)} - 1$ regardless of γ. The inactive conditions $z_j \le \theta$ for $j \ne (1)$ again reduce to the one for (2), which is $g \ge 1$. □

Two consequences of Proposition 1 are used in the main text. Firstly, since the maps preserve score order, top-2 selection always picks $\{(1), (2)\}$, and the second selection is a filler exactly when g reaches the threshold: for a trained router, the singleton rate reported in Tables 2 and 7 is therefore the fraction of its top-2 gaps at or above its map's threshold, $\Pr(K_+{=}1) = \Pr(g \ge t_\pi)$ with $t_\pi \in \{2, 1, 1\}$ for 1.5-entmax, sparsemax and normmax-2. Secondly, sparsemax and normmax-2 share the threshold 1, so on the same score vector they always agree on whether the token has one or two contributors, even though their full supports and their mixture weights differ. Any difference between their contributor counts in Table 2 must therefore come from the scores their routers learned, which is the observation that §4.3 develops.

How the second expert fades. The thresholds describe where the second selected expert reaches zero weight; the closed forms on a two-expert support show how it gets there. Suppose only experts (1) and (2) are active, so that $p_{(1)} + p_{(2)} = 1$, and subtract the two active conditions of Equation (11). For sparsemax, $p_{(1)} - p_{(2)} = g$, hence

$$p_{(2)} = \frac{1 - g}{2}, \tag{21}$$

a weight that decreases linearly in the gap and vanishes at $g = 1$. For 1.5-entmax, $\sqrt{p_{(1)}} - \sqrt{p_{(2)}} = g/2$; writing $a = \sqrt{p_{(1)}}$ and $b = \sqrt{p_{(2)}}$ with $a^2 + b^2 = 1$, squaring gives $1 - 2ab = g^2/4$, so $ab = (4 - g^2)/8$ and $a + b = \sqrt{1 + 2ab} = \sqrt{8 - g^2}/2$; then $b = \frac{1}{2}[(a+b) - (a-b)]$ yields

$$p_{(2)} = \frac{\left(\sqrt{8 - g^2} - g\right)^2}{16}, \tag{22}$$

which equals $1/2$ at $g = 0$ and vanishes at $g = 2$. For normmax-2, $(p_{(1)} - p_{(2)})/R = g$ with $R = \sqrt{p_{(1)}^2 + p_{(2)}^2}$; parameterizing $p_{(1)} = (1 + x)/2$ and $p_{(2)} = (1 - x)/2$ gives $R^2 = (1 + x^2)/2$ and hence $x^2 = g^2(1 + x^2)/2$, so

$$p_{(2)} = \frac{1}{2}\Big(1 - \frac{g}{\sqrt{2 - g^2}}\Big), \tag{23}$$

which vanishes at $g = 1$. At a gap of $g = 0.5$, for instance, the second expert keeps a weight of 0.250 under sparsemax, 0.311 under normmax-2 and 0.326 under 1.5-entmax; at the same scores, the maps with the larger threshold, or the more uniform tendency, hold on to the second expert longer. These formulas apply whenever the third score is below the map's threshold, which by Equation (11) is the condition for the support to have exactly two experts.

B.3 Retained mass of sparsemax and normmax-2 on the same scores

Although sparsemax and normmax-2 agree on the contributor count, they do not agree on the discarded mass. The following argument shows that, on the same score vector, sparsemax always keeps at least as much top-2 mass as normmax-2.

Let θ be the normmax-2 threshold and define the truncated scores $u_j(t) = (z_j - t)_+$ for any $t \in \mathbb{R}$. By Equation (4) with $\gamma = 2$, the normmax-2 probabilities are $p_j^{\mathrm{N}} = u_j(\theta)/\|\boldsymbol{u}(\theta)\|_1$ with $\|\boldsymbol{u}(\theta)\|_2 = 1$, and by Equation (14) with $\alpha = 2$ the sparsemax probabilities are $p_j^{\mathrm{S}} = u_j(\tau)$ with $\|\boldsymbol{u}(\tau)\|_1 = 1$. The argument has three steps.

1. *The sparsemax threshold is at least the normmax-2 threshold.* For every vector, $\|\boldsymbol{u}\|_1 \geq \|\boldsymbol{u}\|_2$, so $\|\boldsymbol{u}(\theta)\|_1 \geq \|\boldsymbol{u}(\theta)\|_2 = 1 = \|\boldsymbol{u}(\tau)\|_1$. Since $t \mapsto \|\boldsymbol{u}(t)\|_1$ is nonincreasing, this forces $\tau \geq \theta$. In particular the sparsemax support is contained in the normmax-2 support.
2. *The top-2 share of truncated mass is nondecreasing in the threshold.* Let $m(t) = \big(u_{(1)}(t) + u_{(2)}(t)\big)/\|\boldsymbol{u}(t)\|_1$ be the fraction of truncated mass on the two largest coordinates, and let $s(t)$ be the number of positive coordinates of $\boldsymbol{u}(t)$. On any interval of t where $s(t) \geq 2$ is constant, the numerator has derivative -2 and the denominator has derivative $-s(t)$, so
$$\frac{\mathrm{d}}{\mathrm{d}t}\, m(t) = \frac{-2\,\|\boldsymbol{u}(t)\|_1 + s(t)\big(u_{(1)}(t) + u_{(2)}(t)\big)}{\|\boldsymbol{u}(t)\|_1^2} \geq 0, \tag{24}$$
because the mean of the two largest positive coordinates is at least the mean of all $s(t)$ positive coordinates, i.e. $\big(u_{(1)} + u_{(2)}\big)/2 \geq \|\boldsymbol{u}\|_1/s$. The function m is continuous where the support changes and equals 1 once $s(t) \leq 2$, so it is nondecreasing on all of $\mathbb{R}$.
3. *Conclusion.* The retained top-2 mass of normmax-2 is $m(\theta)$ and that of sparsemax is $u_{(1)}(\tau) + u_{(2)}(\tau) = m(\tau)$, because $\|\boldsymbol{u}(\tau)\|_1 = 1$. By the two previous steps, $m(\tau) \geq m(\theta)$, that is $\delta^{\mathrm{S}} \leq \delta^{\mathrm{N}}$.

This is a same-score statement. The trained sparsemax and normmax-2 routers see different scores, so the ordering of their discarded masses in Table 2 (0.153 against 0.250 at 1B) is consistent with the argument but not implied by it.

B.4 Score scale and top-2 gaps

Proposition 1 turns the question of how often a router drops its second expert into a question about the distribution of its top-2 gaps. The gap distribution has a scale and a shape, and the two play different roles.

Definitions. For a trained router and a fixed MoE layer, let $z_{i,e}$ be the recorded score of expert e for target i, with $M = 285{,}107$ targets per layer (Appendix A.3) and n experts. We center each target's scores across experts, $c_{i,e} = z_{i,e} - \bar{z}_i$ with $\bar{z}_i = \frac{1}{n}\sum_e c_{i,e}$, which removes the additive freedom that no map is sensitive to, and define the layer's score scale as the root mean square of the centered scores,

$$\mathrm{RMS} = \Big(\frac{1}{nM}\sum_{i=1}^{M}\sum_{e=1}^{n} c_{i,e}^2\Big)^{1/2}, \tag{25}$$

Table 5: Fraction of top-2 score gaps at least 1 (the sparsemax and normmax-2 threshold) and at least 2 (the 1.5-entmax threshold) for every trained router, averaged over layers, next to the router's own singleton rate. For each sparse router the own rate equals the fraction at its own threshold; softmax has no finite threshold.

	300M			1B		
Map	$g \geq 1$ (%)	$g \geq 2$ (%)	Own (%)	$g \geq 1$ (%)	$g \geq 2$ (%)	Own (%)
Softmax	45.702	13.785	0.000	44.757	7.824	0.000
1.5-entmax	15.347	0.295	0.295	8.258	0.048	0.048
Sparsemax	9.598	0.001	9.598	6.883	0.006	6.883
Normmax-2	23.144	0.133	23.144	21.256	0.014	21.256

The top-2 gap of target i is $g_i = z_{i,(1)} - z_{i,(2)}$, and g_q denotes the qth percentile of $\{g_i\}$ within the layer. Table 7 in Appendix C.2 averages the scale and the gap percentiles over the four MoE layers; Table 8 in Appendix C lists them per layer. All these quantities are computed in float64 from the recorded FP32 scores, without re-running any model or map.

Why the scale matters. Multiplying the scores by $\rho > 0$ multiplies every gap by ρ and leaves the shape of the gap distribution unchanged, so by Proposition 1 the singleton rate of a map with threshold t on the scaled scores is $\Pr(\rho\, g \geq t)$. Equivalently, in terms of the standardized gap $G_i = g_i/\mathrm{RMS}$, the singleton rate is $\Pr(G \geq t/\mathrm{RMS})$: a router can change how often it drops an expert either by reshaping its standardized gaps or by moving its scale relative to the threshold. Because the router weights are trained by ordinary backpropagation under the map, nothing pins the scale, which is why §4.3 treats it as a learned quantity.

Gap fractions under each threshold. Since the thresholds are conditions on the scores alone, we can ask, for every trained router, what fraction of its gaps lies at or above 1 (the sparsemax and normmax-2 threshold) and at or above 2 (the 1.5-entmax threshold); Table 5 reports both, next to the router's own singleton rate. For each sparse router the own rate coincides with the fraction at its own threshold, as Proposition 1 requires, which is also a check of the recorded scores against the recorded routing. The table makes the scale effect visible: at 1B, 7.8% of the softmax router's gaps exceed 2 and 44.8% exceed 1, whereas the 1.5-entmax router, with an RMS less than half as large, has only 0.05% of its gaps above 2 and 8.3% above 1. Sparsemax and normmax-2 sit at the same threshold, so the difference between their singleton rates (6.9% against 21.3%) is entirely a difference between the gap distributions their routers learned; Table 7 shows that it comes from the whole distribution being shifted up rather than from a heavier tail, since normmax-2's median gap is 0.61 against 0.40 for sparsemax and its 90th percentile 1.06 against 0.81. These fractions describe the recorded score distributions in threshold units; they do not apply one router's map to another router's scores.

C Additional results

C.1 Routing statistics by layer

Table 6 breaks the layer averages of Table 2 down by MoE layer. The singleton rate is strongly depth-dependent for sparsemax and normmax-2, which route between a fifth (sparsemax) and over half (normmax-2) of the tokens to a single expert in the first MoE layer and under a tenth in the last, while 1.5-entmax has essentially no singletons at any depth. The two load conventions also separate by layer: writing $f_j^{\mathrm{nom}} = f_j^{+} + f_j^{0}$ for the contributor and filler parts of the nominal load, $\mathrm{Var}_j(f_j^{\mathrm{nom}}) = \mathrm{Var}_j(f_j^{+}) + \mathrm{Var}_j(f_j^{0}) + 2\,\mathrm{Cov}_j(f_j^{+}, f_j^{0})$, so a negative covariance between contributor and filler loads can mask contributor imbalance in the nominal histogram, which is what happens to normmax-2 in the first layers.

Table 6: Routing statistics of every MoE layer (network depths 2, 4, 6 and 8) on the Paloma WikiText-103 test documents. Each block reports one statistic of Table 2 per layer instead of its layer average.

	300M, MoE layer				1B, MoE layer			
Map	2	4	6	8	2	4	6	8
Singleton rate $\Pr(s{=}1)$ *(%)*								
Softmax	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1.5-entmax	0.059	0.048	0.072	1.000	0.003	0.001	0.031	0.159
Sparsemax	27.934	5.451	2.002	3.003	19.267	0.887	3.573	3.806
Normmax-2	58.281	18.621	9.147	6.526	54.455	11.980	11.483	7.104
Discarded mass $\bar{\delta}$								
Softmax	0.609	0.510	0.556	0.588	0.682	0.671	0.626	0.664
1.5-entmax	0.332	0.267	0.279	0.369	0.448	0.480	0.423	0.496
Sparsemax	0.077	0.068	0.095	0.159	0.114	0.210	0.133	0.156
Normmax-2	0.117	0.102	0.144	0.220	0.133	0.307	0.227	0.333
Contributors $\overline{K}_+$								
Softmax	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
1.5-entmax	1.999	2.000	1.999	1.990	2.000	2.000	2.000	1.998
Sparsemax	1.721	1.946	1.980	1.970	1.807	1.991	1.964	1.962
Normmax-2	1.417	1.814	1.909	1.935	1.455	1.880	1.885	1.929
Nominal load CV								
Softmax	0.255	0.322	0.508	0.530	0.260	0.401	0.305	0.391
1.5-entmax	0.261	0.332	0.501	0.505	0.236	0.341	0.326	0.448
Sparsemax	0.279	0.389	0.591	0.479	0.367	0.444	0.419	0.426
Normmax-2	0.256	0.343	0.473	0.396	0.244	0.388	0.372	0.361
Contributor load CV								
Softmax	0.255	0.322	0.508	0.530	0.260	0.401	0.305	0.391
1.5-entmax	0.261	0.332	0.501	0.506	0.236	0.341	0.326	0.449
Sparsemax	0.330	0.389	0.591	0.491	0.396	0.444	0.417	0.437
Normmax-2	0.355	0.356	0.498	0.409	0.373	0.414	0.386	0.377

Table 7: Learned score spread and top-2 gap distribution of the trained routers on the Paloma WikiText-103 test documents. RMS is the root mean square of the centered scores and g_{50}, g_{90}, g_{99} are percentiles of the top-2 score gap $z_{(1)} - z_{(2)}$, all averaged over the four MoE layers. A selected expert is dropped exactly when the gap reaches the map's threshold, so the singleton rate is the fraction of gaps at or above it.

		300M					1B				
Map	Threshold	RMS	g_{50}	g_{90}	g_{99}	Singletons (%)	RMS	g_{50}	g_{90}	g_{99}	Singletons (%)
Softmax	–	0.98	0.91	2.14	3.15	0.00	0.86	0.90	1.88	2.92	0.00
1.5-entmax	2	0.52	0.45	1.12	1.71	0.30	0.36	0.44	0.95	1.45	0.05
Sparsemax	1	0.41	0.36	0.84	1.27	9.60	0.32	0.40	0.81	1.20	6.88
Normmax-2	1	0.53	0.64	1.12	1.60	23.14	0.38	0.61	1.06	1.42	21.26

Table 8: Score scale and top-2 gap percentiles of every trained router in every MoE layer (network depths 2, 4, 6 and 8) on the Paloma WikiText-103 test documents; Table 7 averages these values over layers. RMS is the root mean square of the centered scores and g_q the qth percentile of the top-2 gap.

	300M, MoE layer				1B, MoE layer			
Map	2	4	6	8	2	4	6	8
RMS								
Softmax	0.875	1.082	1.037	0.935	0.818	0.847	0.861	0.902
1.5-entmax	0.467	0.531	0.552	0.518	0.355	0.354	0.379	0.352
Sparsemax	0.384	0.398	0.455	0.416	0.322	0.283	0.336	0.339
Normmax-2	0.500	0.539	0.554	0.532	0.406	0.348	0.374	0.373
g_{50}								
Softmax	0.849	1.131	0.877	0.773	0.937	0.998	0.993	0.678
1.5-entmax	0.398	0.566	0.498	0.346	0.530	0.435	0.425	0.354
Sparsemax	0.429	0.440	0.315	0.237	0.663	0.314	0.370	0.262
Normmax-2	1.137	0.578	0.468	0.358	1.044	0.490	0.518	0.369
g_{90}								
Softmax	2.430	2.294	1.865	1.968	1.924	1.813	1.967	1.799
1.5-entmax	1.345	1.132	1.094	0.907	1.042	0.897	0.930	0.916
Sparsemax	1.148	0.873	0.711	0.614	1.063	0.689	0.773	0.730
Normmax-2	1.415	1.181	0.981	0.902	1.282	1.033	1.030	0.892
g_{99}								
Softmax	3.067	3.087	2.795	3.649	2.715	2.414	3.049	3.501
1.5-entmax	1.711	1.567	1.557	2.000	1.463	1.244	1.455	1.649
Sparsemax	1.319	1.254	1.120	1.383	1.250	0.989	1.213	1.361
Normmax-2	1.639	1.525	1.369	1.845	1.551	1.266	1.428	1.419

C.2 Score distributions and scale by layer

Table 7 gives the score spreads and gap quantiles behind Figure 3, together with the corresponding 300M values. Its quantiles are averaged across layers, whereas the histograms pool the equally sized layer samples. Table 8 gives the per-layer values behind Table 7. The ordering of the scales is the same in every layer: softmax is the largest and sparsemax the smallest, with 1.5-entmax and normmax-2 close together in between. The 1.5-entmax router's 99th-percentile gap stays below its threshold of 2 in every layer, with the single exception of the last MoE layer at 300M, where it reaches 2.000 and where that router also has its highest singleton rate (1.0%, Table 6). For sparsemax and normmax-2, whose threshold is 1, the same percentiles explain the depth dependence of the singleton rate: the first MoE layer is the only one where normmax-2's median gap exceeds 1, and it is the layer where both routers drop the second expert most often.

Table 9: Frozen-weight inference at budgets $K \in \{1, 2, 4, 8\}$ on the retained DCLM diagnostic sample (489 blocks, 1,000,983 scored targets) for models trained at $K = 2$: loss (nats per target), retained mass $1 - \bar{\delta}$ and number of contributors $\overline{K}_+$.

	300M, budget K				1B, budget K			
Map	1	2	4	8	1	2	4	8
Loss (nats)								
Softmax	3.703	3.560	3.623	3.913	3.360	3.253	3.328	3.835
1.5-entmax	3.711	3.562	3.598	3.693	3.363	3.261	3.306	3.512
Sparsemax	3.715	3.565	3.566	3.568	3.365	3.263	3.273	3.284
Normmax-2	3.703	3.562	3.574	3.586	3.368	3.266	3.286	3.329
Retained mass $1 - \bar{\delta}$								
Softmax	0.323	0.437	0.544	0.688	0.255	0.340	0.434	0.582
1.5-entmax	0.502	0.694	0.833	0.951	0.406	0.543	0.664	0.821
Sparsemax	0.669	0.914	0.990	1.000	0.637	0.848	0.966	0.999
Normmax-2	0.662	0.867	0.968	0.997	0.617	0.766	0.880	0.973
Contributors $\overline{K}_+$								
Softmax	1.000	2.000	4.000	8.000	1.000	2.000	4.000	8.000
1.5-entmax	1.000	1.998	3.987	7.854	1.000	1.999	3.996	7.978
Sparsemax	1.000	1.896	2.796	3.064	1.000	1.933	3.369	4.296
Normmax-2	1.000	1.771	2.781	3.451	1.000	1.760	3.091	5.077

C.3 Inference-time budget

Table 9 completes Table 3 with the retained mass and the number of contributors at every budget. Every map has its lowest loss at the trained $K = 2$, so retaining more mass by selecting more experts does not help, and the sparse maps convert a larger budget into far fewer contributors than softmax: at $K = 8$ the 1B sparsemax router uses 4.3 experts on average and normmax-2 5.1, against 8.0 for softmax and 7.98 for 1.5-entmax.

C.4 Full results at 300M

The 300M models use an eight-layer backbone with 26 experts in each of the four MoE layers (Table 4). Their quality, routing and budget results are reported next to the 1B ones in Tables 1, 2 and 3 of the main text. Table 7 gives their score-gap summaries, and Tables 6 and 8 give their per-layer statistics. Figures 4 and 5 show their training curves and gap distributions.

The picture matches 1B. Softmax has the lowest validation loss and the highest mean accuracy among the MoEs in these runs. The 1.5-entmax router retains more mass than softmax with almost two contributors (99.3% of its states have support larger than two, with conditional discarded mass 0.314 against 0.566 for softmax), sparsemax retains the most mass and normmax-2 uses the fewest contributors. The sparse routers again learn smaller score spreads than softmax (RMS 0.41–0.53 against 0.98); the 1.5-entmax router's 99th-percentile gap is 1.71, below its threshold of 2, while 13.8% of the softmax router's gaps exceed 2, and normmax-2's gaps exceed 1 far more often than sparsemax's (23.1% against 9.6%). The one difference from 1B is the load balance: at 300M, normmax-2 has a lower nominal CV than softmax (0.367 against 0.404) but the same contributor CV (0.404), and 1.5-entmax has the lowest contributor CV (0.400).

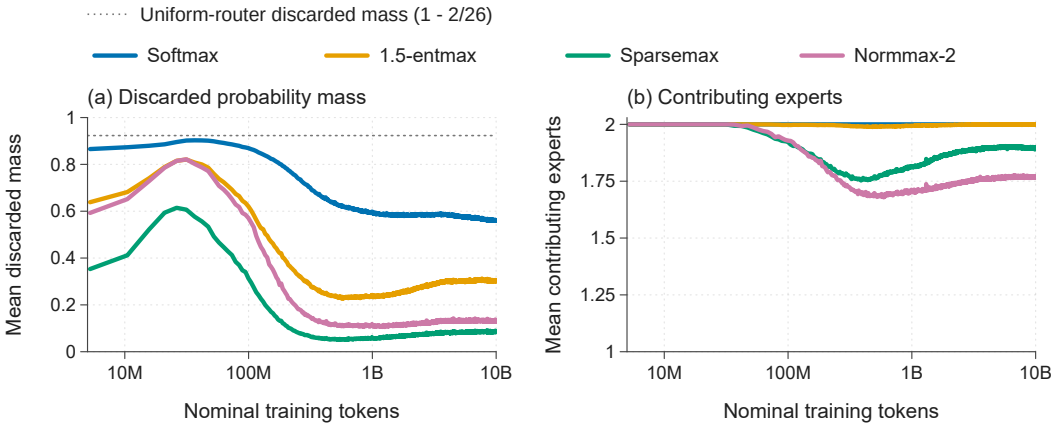

Figure 4: Discarded probability mass (a) and number of contributing experts (b) during training of the 300M MoE models, measured on training batches; the horizontal axis is the number of training tokens (log scale).

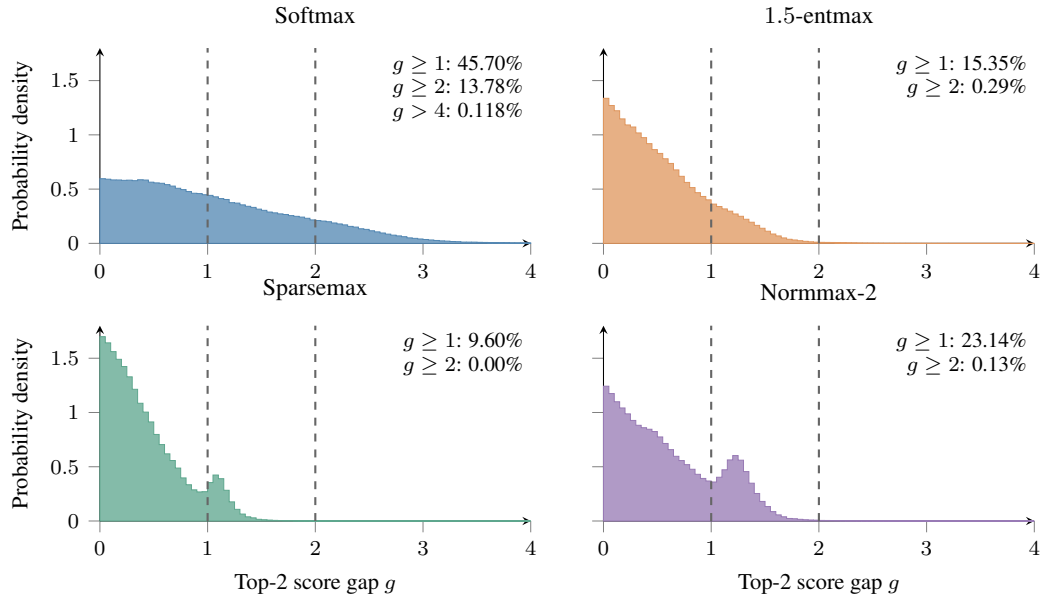

Figure 5: Top-2 score gaps of the original 300M seed-42 routers on the same 61 Paloma WikiText-103 test documents. Samples, normalization and threshold lines follow Figure 3. The peaks near $g = 1$ for sparsemax and normmax-2 are likewise consistent with gradient saturation beyond the singleton threshold.

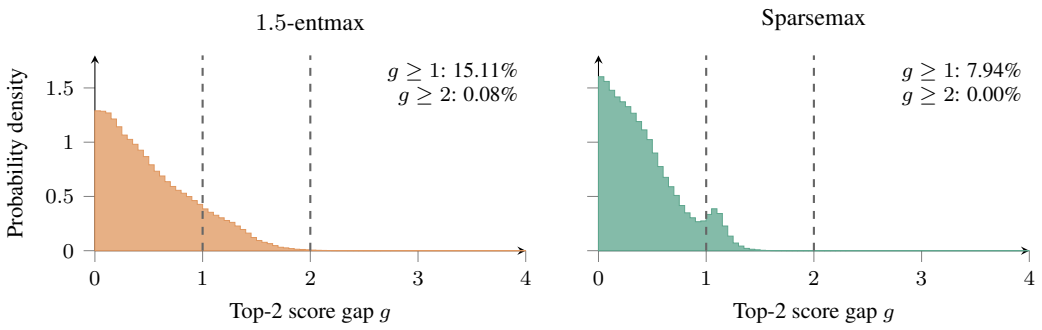

Figure 6: Top-2 score gaps of the 300M sparsemax and 1.5-entmax routers trained with the softmax-based balancing loss. Each panel pools 285,107 target positions from each of the four MoE layers with equal weight, using the same 61 Paloma WikiText-103 test documents as Figure 5. Dashed lines mark the singleton threshold $g = 1$ for sparsemax and $g = 2$ for 1.5-entmax, with $g = 1$ also shown in the entmax panel for reference.