Load Balancing Mixture of Experts with Similarity Preserving Routers

Nabil Omi*
University of Washington

Siddhartha Sen
Microsoft

Ali Farhadi
University of Washington

Abstract

Sparse Mixture of Experts (MoE) models offer a scalable and efficient architecture for training large neural networks by activating only a subset of parameters ("experts") for each input. A learned router computes a distribution over these experts, and assigns input tokens to a small subset. However, without auxiliary balancing mechanisms, routers often converge to using only a few experts, severely limiting model capacity and degrading performance. Most current load balancing mechanisms encourage a uniform routing probability across experts. Early in pretraining, this can result in inconsistent routing behavior, resulting in the model spending its capacity learning redundant knowledge. We address this by introducing a novel load balancing loss that helps preserve relational structure, encouraging consistent expert choices for similar inputs during training. Our experimental results show that applying our loss to the router results in 36% faster convergence and lower redundancy compared to a popular load balancing loss.

1 Introduction

As the demand for larger and more capable neural networks continues to grow [1, 2], the need for architectures that can scale efficiently–without incurring prohibitive computational costs–has become increasingly important. This is especially true in the context of large language models (LLMs), where state-of-the-art performance often requires billions of parameters and massive training datasets. One such approach, the Mixture of Experts (MoE) model [3], introduces sparsely activated sub-networks at certain layers, allowing for increased model capacity while preserving computational efficiency.

While MoE architectures offer improved parameter scalability, they often suffer from poor expert utilization during pretraining. Without mechanisms that encourage balanced routing, the model frequently learns to rely on only a small subset of experts [4, 5]. Typically, routing decisions are made per token using a learned router that outputs a probability distribution over experts–a paradigm known as Token Choice (TC) [6]. To encourage balanced expert usage, various strategies have been proposed, including sequence-level auxiliary losses such as load balancing loss (LBL) [6] or the Expert Choice (EC) routing variant which generates a distribution over a sparse set of activated tokens for each expert [7]. Section 5 covers additional strategies for load balancing.

Load balancing strategies often encourage a uniform distribution over experts to avoid collapse. This approach has proven to be useful to stabilize MoEs during training, and has been used in many recent works [8–11]. However, in this paper, we argue that imposing a uniform distribution over experts causes MoE models to expend their capacity acquiring the same knowledge across multiple experts. Besides the inefficiencies imposed by this approach, exposing similar tokens to several different experts during training results in inconsistent routing behavior and expert assignments. This in turn further exacerbates knowledge redundancy across experts. Previous work [9, 12] suggests that the amount of knowledge shared between experts is correlated to losses in performance. Critically, the

*Corresponding author: nabilomi@cs.uw.edu

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

damage from inconsistent routing occurs early in pretraining, when token embeddings are volatile and routing decisions are unstable. As we show in Section 4.3, the majority of expert redundancy accumulates in the first quarter of training, after which routing patterns stabilize and redundancy growth plateaus. Once experts have learned redundant knowledge, this inefficiency persists throughout training.

To encourage consistent expert assignments, we propose preserving the relational structure among tokens during routing, resulting in similar expert distributions for similar tokens. We introduce **sim**ilarity-preserving routers for MoE load **bal**ancing (SIMBAL), a novel auxiliary loss that achieves this by softly encouraging semi-orthogonality in the router weights. The router projects tokens onto a lower dimensional subspace; semi-orthogonality ensures this projection preserves relational structure, rather than introducing arbitrary distortions that decouple routing decisions from pairwise token relationships.

In summary, our contributions are:

1. We introduce Pairwise Expert Similarity (PES), a metric for monitoring fine-grained expert redundancy during training without expensive ablations.
2. We identify that expert redundancy in MoE models is primarily acquired early in pretraining, when routing is unstable.
3. We propose SIMBAL, a load balancing loss that encourages semi-orthogonality in router weights, preserving token-wise structure. This results in improved routing stability during pretraining, 36% faster convergence, and reduced perplexity at the same compute budget.

2 Background

2.1 Mixtures of Experts

A Mixture of Experts (MoE) model *sparsely activates* certain parameters during inference, in contrast to standard dense networks where all parameters are used. In this work, we focus on Mixture of Experts models for the Transformer architecture [13], a popular choice for training models on sequence-wise data such as those seen in natural language.

Transformers are typically composed of a series of blocks, each consisting of a self-attention module followed by a feed-forward network (FFN). The FFN is usually a two-layer fully connected network with a large hidden dimensionality. For example, given an input vector $x \in \mathbb{R}^{D_M}$, where D_M is the model (input/output) dimensionality, the standard FFN computes:

$$\text{FFN}(x) = W_2 \cdot \sigma(W_1 x + b_1) + b_2, \tag{1}$$

where $W_1 \in \mathbb{R}^{D_F \times D_M}$, $W_2 \in \mathbb{R}^{D_M \times D_F}$, $b_1 \in \mathbb{R}^{D_F}$, and $b_2 \in \mathbb{R}^{D_M}$. The intermediate hidden dimension D_F is typically much larger than D_M. The nonlinearity σ is an activation function; we use SwiGLU [14].

In a Mixture of Experts Transformer, the FFN is replaced by a set of smaller, parallel FFNs called "experts." Let there be E such experts. Each expert has its own parameters $\{W_1^{(e)}, W_2^{(e)}, b_1^{(e)}, b_2^{(e)}\}$, where $W_1^{(e)} \in \mathbb{R}^{D_E \times D_M}$, $W_2^{(e)} \in \mathbb{R}^{D_M \times D_E}$, $b_1^{(e)} \in \mathbb{R}^{D_E}$, and $b_2^{(e)} \in \mathbb{R}^{D_M}$. Here, D_E is the hidden dimension used within each expert.

A routing mechanism assigns each token $x \in \mathbb{R}^{D_M}$ to a small subset of A activated experts (typically $A \ll E$). The router is a linear transformation $R \in \mathbb{R}^{D_M \times E}$ followed by a sparse top-A selection, producing expert indices $i_1, \ldots, i_A$ and associated routing weights $r_1, \ldots, r_A$. The MoE layer then computes:

$$\text{MoE}(x) = \sum_{a=1}^{A} r_a \cdot \left(W_2^{(i_a)} \cdot \sigma(W_1^{(i_a)} x + b_1^{(i_a)}) + b_2^{(i_a)} \right). \tag{2}$$

This definition of the MoE can also be viewed as a weighted sum over expert FFN outputs, skipping the computation for any expert where the weight is zero. This architecture enables scaling model

capacity via E without a proportional increase in computational cost, as only A experts are active per input.

2.2 Expert Routing

Despite the small parameter count of MoE routers (in our larger setting, 0.018% of the total parameters), they have an outsized impact on the performance and capacity of the model, as they orchestrate billions of parameters. Thus, it is imperative to pay careful attention to this mechanism when training MoE models. In MoE Transformers, routing is computed from the previous attention output $x \in \mathbb{R}^{D_M}$ via a learned router matrix, producing scores $xR \in \mathbb{R}^E$. Applying a gating function G results in routing weights $r = G(xR)$. We use softmax, which generates a probability distribution over experts, from which the top-A active experts are selected and weighted for each token.

We compare our approach to balancing with the Load Balancing Loss (LBL) presented by Fedus et al. [6]. This setup is highly popular and represents the state-of-the-art, being used in Muennighoff et al. [8], DeepSeek-AI et al. [10], Dai et al. [9], and [11] (we give an overview of alternative methods and their limitations in Section 5.) LBL encourages uniform expert usage by correlating how frequently each expert is selected with how much routing weight it receives. Let f_i be the fraction of tokens routed to expert i, P_i the average routing probability for expert i, and E the number of experts. The LBL is defined as:

$$\mathcal{L}_{\text{LBL}} = \alpha \cdot E \cdot \sum_{i=1}^{E} f_i \cdot P_i \tag{3}$$

Minimizing this loss encourages the router to distribute tokens more evenly across experts. However, it may require tuning of a loss coefficient α to avoid overpowering the main training objective. We include PyTorch implementation details in Appendix G.

3 Methods

We first introduce Pairwise Expert Similarity (PES), a metric for monitoring expert redundancy during training (Section 3.1). We then describe SIMBAL, an auxiliary loss that encourages semi-orthogonality in router weights to preserve token-wise structure (Section 3.2). Finally, we detail our model architecture and training setup (Section 3.3).

3.1 Measuring Expert Similarity

In MoE models, expert redundancy occurs when multiple experts learn overlapping knowledge, wasting capacity without adding unique functionality. Quantifying it is therefore central to understanding specialization and efficiency. Prior work measures redundancy by dropping the top-K experts and recording the performance hit [9], but our claim that excessive routing instability early in training degrades performance requires analyzing the redundancy of *all* experts, not just the top-routed ones. Extending the dropping analysis this way means sweeping drop counts at every layer (for example, Figure 2(a)). Since deeper layers depend on earlier ones, these ablations cannot be amortized: cost scales as $L \cdot N$ full forward passes, or 384 validation runs per checkpoint for MoE-L ($L = 12$, $N = 32$). It is also noisy, as loss climbs toward untrained-model levels once many experts are dropped, swamping the signal between nearby checkpoints.

To enable finer-grained, more frequent monitoring, we introduce *Pairwise Expert Similarity (PES)*, which directly measures how similar experts' outputs are on a shared batch of tokens. Computing PES requires a single extra inference pass in which all N experts per layer are evaluated rather than only the top-K. The added cost is dominated by the MoE FFN and scales as N/K; for our configurations ($N/K = 8$) this is roughly a 3.6 to $4.9\times$ FLOP overhead per token (a reduction of two orders of magnitude), small enough to run periodically during training. PES is defined as:

$$\text{PES}_{\text{layer}}(\mathbf{x}) = \frac{2}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \cos(\mathbf{f}_i(\mathbf{x}),\, \mathbf{f}_j(\mathbf{x})) \tag{4}$$

where $\mathbf{f}_i$ is the function computed by the i-th expert. We report PES averaged over a validation batch B and across all MoE layers; lower values indicate more diverse, less redundant experts.

PES and expert dropping both target expert redundancy, the extent to which experts learn overlapping, substitutable functions. Expert dropping tests this indirectly by removing top-routed experts and measuring the increase in perplexity: useful, but coarse and expensive. PES compares expert outputs on the same hidden states, giving a smooth expert-level signal in one diagnostic pass. We use PES to track how SIMBAL shapes expert specialization, with expert dropping as a check against prior work.

Table 2 reports PES, and Figure 4 reproduces the model-level dropping analysis from Dai et al. [9], which measures the effect of dropping top-routed experts on overall validation perplexity. Both analyses agree that SIMBAL reduces redundancy: it has lower PES (expert outputs overlap less), and its perplexity rises more sharply under expert dropping, meaning those experts are harder to replace.

3.2 Load Balancing via Orthogonal Routers

A natural strategy to ensure expert choices correlate with token-wise relationships is to constrain the router weights to form an orthonormal matrix. However, MoE routers are tall matrices ($D_M \gg E$), and thus cannot be square orthonormal. Instead, we constrain the router weights to be semi-orthogonal: $R^\top R = I_E$. This ensures that routing score similarity faithfully reflects input similarity within the router's learned subspace, rather than being distorted by correlated columns or imbalanced norms which can result in instability during training.

Intuition behind semi-orthogonality encouraging load balance. Semi-orthogonality has a direct consequence for expert utilization under mild assumptions on the router input distribution. Let $z = xR \in \mathbb{R}^E$ denote the (pre-gating) routing scores. If x is approximately isotropic with $\mathrm{Cov}(x) \approx \sigma^2 I_{D_M}$, then

$$\mathrm{Cov}(z) = R^\top \mathrm{Cov}(x)\, R \approx \sigma^2 R^\top R = \sigma^2 I_E, \tag{5}$$

so each pair z_e, z_f is uncorrelated and all coordinates have the same variance; no expert direction is privileged at second order. We assume the stronger property that the coordinates of z are approximately exchangeable—i.e., their joint distribution is approximately invariant under expert relabeling. Equation (5) is consistent with this assumption but does not imply it, since R is a fixed learned projection rather than a random one. Under approximate exchangeability, the permutation-equivariance of softmax and top-k propagates to the selection set $S(x) \subset \{1, \ldots, E\}$, yielding $\Pr[e \in S(x)] \approx k/E$ for every expert e. A semi-orthogonal R thus removes a structural mechanism—correlated or unequal-norm columns—through which imbalance would otherwise be amplified. Appendix A reports IsoScore [15] (a measure for the isotropy of an embedding space) > 0.9 on router inputs in our trained MoE models (and open-weight pre-trained checkpoints, to verify that this is not an artifact of our training); residual deviations are a natural source of residual imbalance, consistent with our empirical observations.

We encourage semi-orthogonality with a soft penalty on the entrywise deviation of the Gram matrix $R^\top R$ from the identity:

$$\mathcal{L}_{\text{orth}} \;=\; \left\| R^\top R - I_E \right\|_1 \;=\; \sum_{i,j} \left| [R^\top R]_{ij} - \delta_{ij} \right|. \tag{6}$$

The diagonal terms drive each expert's routing direction to unit norm; the off-diagonal terms penalize pairwise alignment between experts. This loss is dataset-agnostic and computationally cheap, which is important, as Qiu et al. [16] finds that data-dependent losses require large batch sizes to be effective. We additionally initialize the router with a (near) orthogonal initialization [17], which we find accelerates convergence. We call this method SIMBAL, as we are effectively balancing by preserving the pair-wise similarity of tokens. The experiments in our paper scale this loss by 0.1, but we find SIMBAL to be relatively insensitive to this as long as the coefficient is small, as shown in Table 6 in the appendix. PyTorch implementation details are in Appendix G.

3.3 Model Architecture and Training

Model Architecture. Our model architecture closely follows prior work by [18] and [8]. We use a Transformer backbone with RMSNorm [19], SwiGLU activations [14], and Rotary Position Embeddings (RoPE) [20]. We apply Z-loss [21, 22] with a coefficient of 1e-5, as in [18]. Unlike

OLMo 2, we do not modify the placement of normalization layers nor do we apply QK-Norm [23]. We replace all FFN layers with MoE layers. Further architectural details can be found in Table 8. Our implementation builds upon the open-source OLMo codebase [18].

For the LBL baseline, we follow Muennighoff et al. [8] and Wang et al. [24], using a loss coefficient of 0.01. In contrast, our method does not require a coefficient; with appropriate initialization, the load-balancing loss converges quickly.

Model Scales and Training. We pretrain models at two scales: a medium model (MoE-M) to ablate design decisions with 230M active and 627M total parameters, and a large model (MoE-L) with 762M active and 3.14B total parameters (including embeddings). For each scale, we performed a brief hyperparameter sweep across three learning rates. All models are trained using the AdamW optimizer [25], with a weight decay of 0.01, linear warm-up from 10% of the peak learning rate over 2000 steps, followed by cosine decay [26] to 10% of the peak learning rate. Additional model specifications are listed in Table 8. All model parameters are in `bfloat16`.

All models are trained on a subset of tokens from the DCLM-pool-400m-1x dataset [27] (used in other work such as Muennighoff et al. [8]), tokenized using the cl100k_base tokenizer from the tiktoken library [28]. We reserve one file shard (77M tokens) for validation. All MoE-M models are trained on 19.9B tokens, while MoE-L models are trained on 78.6B tokens–sufficient for our analysis because routing instability and redundancy accumulation manifest early in pretraining and plateau well before the end (Section 4.3). No further fine-tuning is performed, as our focus is on pretraining dynamics, where load balancing has the greatest impact.

Resources. All models are trained using Distributed Data Parallelism (DDP) [29]. For MoE-M, we use 8 NVIDIA A100 40GB GPUs per training run; for MoE-L, we use 8 AMD MI300X 192GB accelerators. We provide FLOP estimates in Appendix C.

As for overheads, both auxiliary losses add negligible computation relative to a training step. For an illustrative batch of 32 sequences per accelerator across 8 accelerators, with sequence length 2,048, we estimate approximately 0.30 GFLOPs for SIMBAL and 0.40 GFLOPs for LBL, compared with 1.91 PFLOPs for the MoE-L training step. LBL additionally constructs a routing mask of approximately 2 MB per layer per accelerator, whereas SIMBAL operates on a 32×32 router Gram matrix, independent of batch size and sequence length.

4 Experiments

4.1 Orthogonalization and Balancing

One of our key contributions is load balancing via semi-orthogonal routers. Semi-orthogonality ensures that routing scores faithfully reflect input similarity within the router's learned subspace, rather than being distorted by correlated columns or imbalanced norms. These distortions lead to inconsistent expert assignments for similar tokens. Rather than enforcing semi-orthogonality through explicit parameter constraints (which is computationally expensive, requires frequent reparameterization, and is prone to numerical instability at scale), we use the loss function described in Section 3.2. We provide a detailed comparison with existing parameterizations in Appendix D. We now evaluate SIMBAL across three axes: load balancing, orthogonalization, and routing geometry.

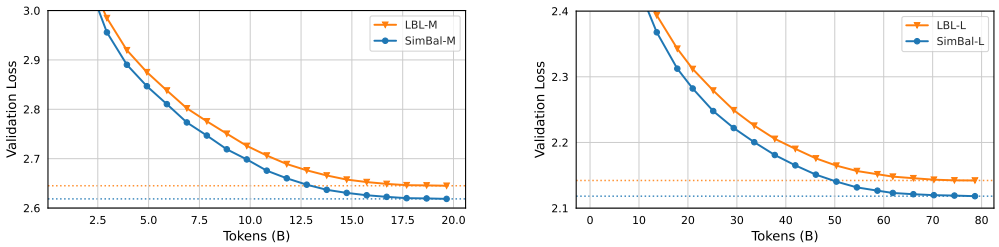
Load balancing. In terms of expert utilization in MoEs, our method avoids collapse comparably to LBL, ensuring that no experts remain unutilized. To verify that sequence-wise balance is not substantially degraded, we compare SIMBAL against LBL by measuring the entropy of the routing distributions and Sequence-wise Expert Utilization (SEU), reported as the mean over the fraction of experts used per sequence. We report our results in Table 1.

Orthogonalization. To assess whether SIMBAL effectively orthogonalizes routing matrices, we measure the mean layer-wise L2 distance of the final router gram matrix from the identity (Table 1). SIMBAL achieves near-numerical-precision orthogonality; layer-wise values across MoE-M and MoE-L are reported in Appendix E.

Ablations. To rule out two confounders—(a) initialization alone, and (b) the use of any static orthogonal matrix, we train two ablations: one with an unconstrained router with an orthogonal

Table 1: Load balancing and orthogonalization of LBL and SIMBAL on MoE-L.

Metric	SEU	Entropy	$(R^T R - I)^2$
LBL	1.000	1.268	0.0311
SIMBAL	0.991	1.168	2.121×10^{-8}



(a) MoE-M (b) MoE-L

Figure 1: Validation loss curves for checkpoints during training. In both MoE-M and MoE-L, we achieve the same loss roughly 36% faster.

initialization (no auxiliary loss), and one where the router is frozen at its orthogonal initialization generated by QR decomposition. The first collapses (Figure 5). The second outperforms LBL (Table 2, Frozen) but still trails SIMBAL, suggesting that tunability lets the model find a better router *within* the space of semi-orthogonal solutions. To validate our findings about the load balancing mechanism, we additionally find that the frozen orthogonal router does not have significantly degraded routing entropy compared to SIMBAL, with an entropy of 1.121.

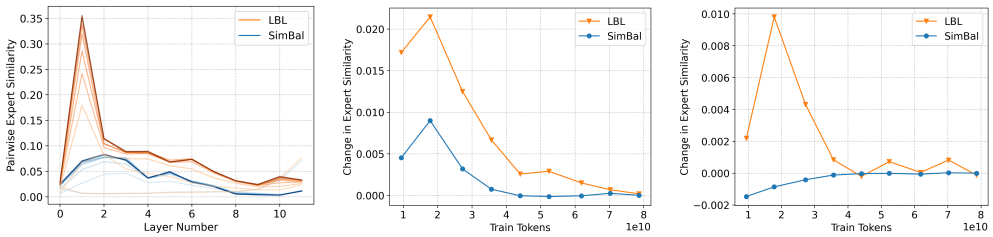
4.2 Language Modeling

We compare our method to LBL and ReMoE [35], a fully differentiable MoE baseline with ReLU routing, by training language models according to the setup described in Section 3.3 and evaluating the perplexity of the final checkpoint. We evaluate ReMoE at the MoE-M scale. The resulting models are reported in Table 2. We additionally report the SEU of the models.

Convergence speed. Across both MoE-M and MoE-L scales, SIMBAL converges approximately 36% faster than LBL. We show validation loss during training in Figures 1(a) and 1(b). For MoE-L, SIMBAL approaches the target loss after processing roughly 50B tokens, compared to 78.6B for LBL, a 36% improvement. Similarly, in the MoE-M setting, SIMBAL reaches comparable loss levels at around 12.7B tokens, versus 19.9B for LBL.

Downstream performance. We additionally evaluate MoE-L on standard downstream benchmarks at both 52B and 78.6B training tokens (Table 3). Downstream performance at this scale is inherently noisy: strong and consistent benchmark results typically require significantly more tokens [8]. Nevertheless, at 52B tokens, SIMBAL outperforms LBL on five of six benchmarks, improving mean performance from 43.75% to 44.73%. At 78.6B tokens, the mean gap increases to 1.91 percentage points, from 43.28% to 45.19%. These results suggest that improved routing stability translates to downstream gains even at this relatively modest scale, with benefits already apparent at the earlier checkpoint. As Figure 2(b) shows, redundancy accumulated under LBL does not diminish over time, while SIMBAL maintains consistently low redundancy throughout.

Statistical significance. Due to computational limitations, we additionally train four seeds each of SIMBAL and LBL on MoE-M to assess the statistical significance of our results. LBL achieves a mean perplexity of 14.051 (std 0.026) across the five seeds, while SIMBAL achieves 13.691 (std 0.039). The improvement of 0.36 in perplexity corresponds to a Cohen's d of approximately 10.9—an effect size more than an order of magnitude larger than the seed-to-seed variation of either method—confirming that the gap is highly significant rather than an artifact of seed selection.



(a) Redundancy Per Layer (↓) (b) Δ Redundant Expert Knowledge (c) Min PES Rate of Change (MoE-L)

Figure 2: Expert redundancy in MoE-L. **(a)** Per-layer PES; SIMBAL (blue) stays below LBL (orange), darker = later in training. **(b)** Rate of change of layer-averaged PES; LBL remains above zero for most of training, indicating ongoing accumulation. **(c)** Rate of change in minimum PES across layers.

Table 2: Model setup and performance. Frozen represents a frozen orthogonal router.

Model	Dense-M	MoE-M	MoE-M	MoE-M	MoE-M	Dense-L	MoE-L	MoE-L
Balancing	–	LBL	Frozen	ReMoE	SIMBAL	–	LBL	SIMBAL
Perplexity ↓	19.468	14.086	13.778	13.759	13.685	10.047	8.517	8.304
Min PES ↓	–	0.0255	0.0063	–	0.0044	–	0.0241	0.0028

Table 3: Comparison of LBL-L and SIMBAL-L performance at 52B and 78.6B training tokens. Final-checkpoint results include standard errors.

	52B tokens		**78.6B tokens**	
Benchmark	**LBL-L**	**SIMBAL-L**	**LBL-L ± stderr**	**SIMBAL-L ± stderr**
ARC Challenge [30]	22.61%	23.21%	22.44% ± 1.22%	23.21% ± 1.23%
ARC Easy [30]	40.91%	40.95%	40.49% ± 1.01%	41.16% ± 1.01%
HellaSwag [31]	35.32%	35.59%	35.45% ± 0.48%	35.74% ± 0.48%
PIQA [32]	66.00%	66.38%	66.49% ± 1.10%	66.81% ± 1.10%
WinoGrande [33]	51.22%	50.99%	49.72% ± 1.41%	52.49% ± 1.40%
GLUE [34]	46.45%	51.26%	45.10% ± 1.98%	51.73% ± 1.97%
mean	43.75%	44.73%	43.28%	45.19%

Combining with loss-free balancing. Inspired by the use of multiple balancing mechanisms in DeepSeek-v3 [10], we compare the use of a loss-free balancing mechanism [24] combined with load balancing loss in Table 4. We find that the loss-free balancing results in a model with good perplexity but poor sequence-wise balancing, which can be inefficient during inference. With SIMBAL, we improve sequence-wise balancing to a lesser extent than LBL, but without the degradation in performance. We provide further details and test with an additional gating mechanism in Appendix F.

Sensitivity to loss coefficient. Finally, we examine sensitivity to the auxiliary loss coefficient (0.01, 0.1, 1.0), with results in Table 6. Based on our 5-seed runs on MoE-M, the effect is negligible, and we do not recommend tuning this hyperparameter.

4.3 Redundancy and Specialization in Experts

Lower perplexity alone does not explain *why* SIMBAL outperforms LBL. We argue the gap stems from routing behavior: by penalizing expert similarity rather than enforcing uniform load, SIMBAL produces routers that stabilize earlier and develop less redundant experts. We support this with three measurements: router saturation over training, pairwise expert redundancy, and behavior under inference-time pruning.

Routing stability. We hypothesize that SIMBAL produces more stable routing than LBL. LBL enforces uniform distributions, causing routing instability in early pre-training: changing embeddings combined with near-uniform assignment mean small input perturbations can reassign tokens across experts. We measure this via *router saturation*, the agreement in expert assignments between a given checkpoint and the final model, which directly captures how token-level routing evolves during training. Figure 3 shows that SIMBAL saturates faster than LBL and maintains higher saturation

Table 4: Evaluating the usage of SIMBAL in conjunction with the loss-free approach from [24].

Model	MoE-M	MoE-M	MoE-M
Gating	Softmax	Softmax	Softmax
Balancing	LF	LF+LBL	LF+SIMBAL
Perplexity ↓	13.708	14.154	13.695
SEU ↑	0.505	0.997	0.755

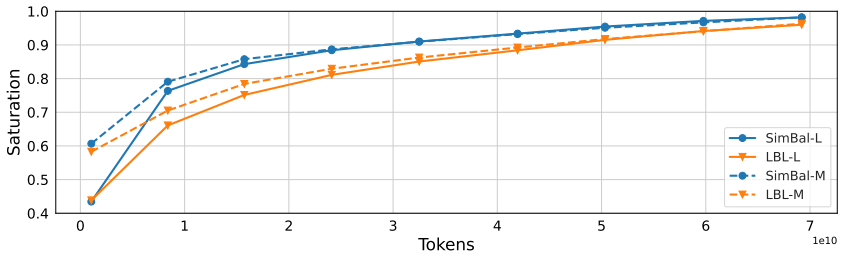


Figure 3: Router saturation for MoE-M and MoE-L: agreement between each checkpoint and the final model. SIMBAL saturates faster and maintains higher agreement, indicating more stable routing.

across checkpoints at both scales, indicating that similarity-preserving routers latch onto stable routing patterns early while LBL continues to shift assignments throughout training.

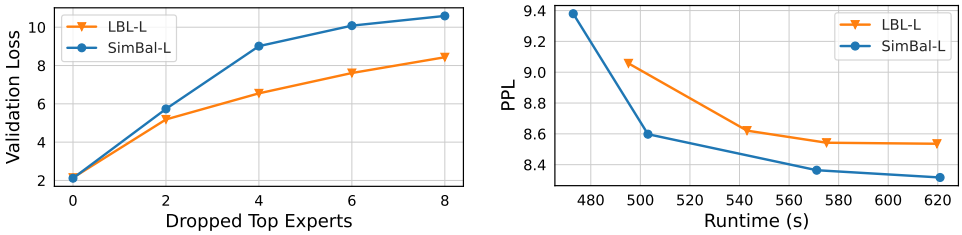
Expert redundancy. This routing instability should result in greater expert redundancy: when similar tokens are routed inconsistently, multiple experts learn overlapping knowledge. As shown in Figure 2(b), most redundancy in LBL (orange) arises early, coinciding with embedding volatility and unstable routing. Redundancy remains above zero through much of training, reinforcing that LBL amplifies it. In contrast, SIMBAL (blue) stabilizes quickly: while expert distributions adapt, they converge to consistently lower PES (Figure 2(a)). Moreover, the rate of change remains near zero for most of training, showing that our method avoids the issues of LBL. The semi-orthogonality constraint induces genuinely different routing behavior. Final PES values are summarized in Table 2, where we report the minimum PES across all layers to reduce sensitivity to outliers; we choose minimum since we do not observe substantial dips in PES by layer, primarily jumps, and we wanted this metric to be as simple and intuitive as possible. SIMBAL consistently produces models with substantially lower minimum PES than LBL. Figure 2(c) shows the rate of change in minimum PES over time.

Inference-time pruning. This specialization also makes routing weights more informative: a low probability under SIMBAL signals a low-weight expert, while LBL's uniformity inflates such assignments. This makes SIMBAL well-suited to inference-time *expert pruning* [36], where experts below a probability threshold are dropped at runtime (Table 7, Figure 4(b)). SIMBAL prunes experts with smaller weights, yielding large speedups at minimal perplexity cost; LBL's flatter distributions mean fewer experts clear the threshold, muting the speedup (the smaller perplexity hit likely reflects redundant experts compensating for one another, Figure 4). At threshold 0.15, where perplexities are comparable, SIMBAL achieves a 7.4% speedup (543s vs. 503s).

5 Related Work

There has been significant interest in MoE models for scaling LLMs, as shown in Lepikhin et al. [37], Zoph et al. [38], Fedus et al. [6], Xue et al. [11], DeepSeek-AI et al. [10], Databricks [39], Llama [40], Muennighoff et al. [8], and more. We explore related design choices below.

Routing and Load Balancing Mechanisms. Efficient routing in MoE architectures involves selecting appropriate experts for each token (Token Choice) [6] while ensuring balanced expert utilization. Some previous work suggests allowing experts to choose the tokens they process (Expert Choice) [7], but this tends to have issues regarding performance in autoregressive generation [8], and leak information about future tokens [24].



(a) Dropped Top Experts vs. Validation Loss

(b) Dynamic-K Expert Selection

Figure 4: **(a)** Validation loss vs. number of dropped top experts [9]. SIMBAL degrades more sharply, indicating lower redundancy. **(b)** Dynamic-K expert pruning [36] on MoE-L. SIMBAL preserves more performance when low-weight experts are dropped. Lower = better; raw values in Table 7.

Traditional approaches employ an auxiliary load balancing loss [6] to encourage a uniform distribution over experts, which can interfere with the main training objective and potentially degrade performance. To address this, auxiliary-loss-free (LF) strategies have been introduced [24], notably used in DeepSeek-V3 [10], but in conjunction with an auxiliary balancing loss. LF dynamically adjusts per-expert bias terms added to the routing scores, guiding top-K expert selection without introducing additional gradients. LF struggles to balance MoE usage sequence-wise, often degrading throughput.

Existing data-dependent approaches (LF and LBL) are highly sensitive to batch size: Qiu et al. [16] report a substantial perplexity drop when training with batch size 512 vs. 4 (per-device, no sync). This effect is far milder for LBL, and entirely absent for SIMBAL, which is invariant to the data. Finally, while Qiu et al. [16] argue that LBL requires distributed synchronization to maximize batch size and improve specialization, SIMBAL eliminates this need altogether.

Orthogonality in MoE. Prior studies have applied orthogonality to diversify expert representations in MoE models. OMoE [12] introduces an optimizer that updates each expert in a direction orthogonal to the subspace spanned by other experts, enhancing representation diversity. MOORE [41] employs the Gram-Schmidt process to enforce orthogonality among expert representations in multi-task reinforcement learning. In contrast, our approach applies orthogonality at the *router* level, not the experts themselves. This strategy offers computational efficiency by avoiding expensive operations during training and allows seamless integration into existing architectures. Moreover, by not constraining expert weights, we avoid potential performance degradation due to restrictive parameter constraints. ERNIE 4.5 [42] claim to use an orthogonality loss within their router, but their loss does not result in an orthogonal router, as they minimize cosine similarity of router features. This de-correlates the routes but does not encourage (semi-)orthogonality. As ERNIE 4.5 is a technical report focused on the LLM as a whole, it does not analyze why this loss helps or characterize its effect on routing behavior.

6 Limitations

Our experiments use moderate token counts (19.9B–78.6B) compared to production-scale training (trillions of tokens). However, as discussed in Section 4.3 and shown in Figure 2(b), the phenomena we study–routing instability, redundancy accumulation, and expert collapse–manifest early in pretraining and plateau mid-run. Our token budget captures this critical phase; extending to trillions of tokens would primarily observe the stable regime where differences between methods are already locked in. We supplement our analysis with statistical significance comparisons across 5 seeds (Section 4.2), confirming that our results are robust. Furthermore, SIMBAL's data-agnostic loss formulation introduces no additional scaling concerns compared to LBL; we leave verification at production scale to future work with greater computational resources. Finally, although our architectural choices align with recent MoE literature, we do not investigate how token dropping might affect our balancing mechanism, instead focusing on higher-quality dropless models [43].

7 Conclusion

We introduced SIMBAL, a load balancing loss for MoE models that encourages semi-orthogonality in router weights. This produces stable routing patterns and reduces expert redundancy during the critical early phase of pretraining. Across two model scales, SIMBAL achieves 36% faster convergence and lower perplexity than the widely-used Load Balancing Loss. We also proposed Pairwise Expert Similarity (PES), a scalable metric for quantifying redundancy, and showed that lower-redundancy models exhibit better training efficiency and synergy with inference-time pruning. Our results suggest that routing stability (rather than just load balance) is crucial for efficient MoE training.

Acknowledgments and Disclosure of Funding

We gratefully acknowledge the University of Washington for its support of this research. The authors declare no competing interests.

References

[1] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[3] 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, 2017. URL https://arxiv.org/abs/1701.06538.

[4] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts, 2014. URL https://arxiv.org/abs/1312.4314.

[5] Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models, 2016. URL https://arxiv.org/abs/1511.06297.

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

[7] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. Mixture-of-experts with expert choice routing, 2022. URL https://arxiv.org/abs/2202.09368.

[8] Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. Olmoe: Open mixture-of-experts language models, 2025. URL https://arxiv.org/abs/2409.02060.

[9] Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2401.06066.

[10] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo,

Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

[11] Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You. Openmoe: An early effort on open mixture-of-experts language models. *arXiv preprint arXiv:2402.01739*, 2024.

[12] Boan Liu, Liang Ding, Li Shen, Keqin Peng, Yu Cao, Dazhao Cheng, and Dacheng Tao. Diversifying the mixture-of-experts representation for language models with orthogonal optimizer, 2024. URL https://arxiv.org/abs/2310.09762.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

[14] Noam Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

[15] William Rudman, Nate Gillman, Taylor Rayne, and Carsten Eickhoff. Isoscore: Measuring the uniformity of embedding space utilization. In *Findings of the Association for Computational Linguistics: ACL 2022*, page 3325–3339. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-acl.262. URL http://dx.doi.org/10.18653/v1/2022.findings-acl.262.

[16] Zihan Qiu, Zeyu Huang, Bo Zheng, Kaiyue Wen, Zekun Wang, Rui Men, Ivan Titov, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Demons in the detail: On implementing load balancing loss for training specialized mixture-of-expert models, 2025. URL https://arxiv.org/abs/2501.11873.

[17] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, 2014. URL https://arxiv.org/abs/1312.6120.

[18] Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam,

Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 olmo 2 furious, 2025. URL https://arxiv.org/abs/2501.00656.

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

[20] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

[21] Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models, 2024. URL https://arxiv.org/abs/2405.09818.

[22] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022. URL https://arxiv.org/abs/2204.02311.

[23] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd van Steenkiste, Gamaleldin F. Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Patrick Collier, Alexey Gritsenko, Vighnesh Birodkar, Cristina Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Pavetić, Dustin Tran, Thomas Kipf, Mario Lučić, Xiaohua Zhai, Daniel Keysers, Jeremiah Harmsen, and Neil Houlsby. Scaling vision transformers to 22 billion parameters, 2023. URL https://arxiv.org/abs/2302.05442.

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

[25] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

[26] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017. URL https://arxiv.org/abs/1608.03983.

[27] Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. Datacomp-lm: In search of the next generation of training sets for language models, 2025. URL https://arxiv.org/abs/2406.11794.

[28] OpenAI. tiktoken: A fast bpe tokenizer for use with openai's models, 2024. URL https://github.com/openai/tiktoken. GitHub repository.

[29] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. Pytorch distributed: Experiences on accelerating data parallel training. *Proceedings of the VLDB Endowment*, 13(12):3005–3018, 2020. doi: 10.14778/3415478.3415530. URL https://doi.org/10.14778/3415478.3415530.

[30] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

[31] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

[32] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

[33] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

[34] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi, editors, *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446/.

[35] Ziteng Wang, Jun Zhu, and Jianfei Chen. Remoe: Fully differentiable mixture-of-experts with relu routing, 2025. URL https://arxiv.org/abs/2412.14711.

[36] Filip Szatkowski, Bartosz Wójcik, Mikołaj Piórczyński, and Simone Scardapane. Exploiting activation sparsity with dense to dynamic-k mixture-of-experts conversion, 2024. URL https://arxiv.org/abs/2310.04361.

[37] 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, 2020. URL https://arxiv.org/abs/2006.16668.

[38] 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, 2022. URL https://arxiv.org/abs/2202.08906.

[39] Databricks. Introducing dbrx: A new state-of-the-art open llm, March 2024. URL https://www.databricks.com/blog/introducing-dbrx-new-state-art-open-llm. Accessed: 2025-05-15.

[40] Llama. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/, April 2025. Accessed: 2025-05-14.

[41] Ahmed Hendawy, Jan Peters, and Carlo D'Eramo. Multi-task reinforcement learning with mixture of orthogonal experts, 2024. URL https://arxiv.org/abs/2311.11385.

[42] Baidu-ERNIE-Team. Ernie 4.5 technical report, 2025. URL https://ernie.baidu.com/blog/publication/ERNIE_Technical_Report.pdf.

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

[44] Kawin Ethayarajh. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1006. URL https://aclanthology.org/D19-1006/.

[45] William Timkey and Marten van Schijndel. All bark and no bite: Rogue dimensions in transformer language models obscure representational quality. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4527–4546, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.372. URL https://aclanthology.org/2021.emnlp-main.372/.

[46] OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin Fives, Vlad Fomenko, Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam Goucher, Lukas Gross, Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu, Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel, Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott Lessans, Mario Lezcano-Casado, Yuanzhi Li, Zhuohan Li, Ji Lin, Jordan Liss, Lily, Liu, Jiancheng Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCallum, Josh McGrath, Scott McKinney, Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander Neitz, Alex Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo, Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass, Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo Raso, Hongyu Ren, Kimmy Richardson, David Robinson, Bob Rotsted, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey, Zhiqing Sun, Philippe Tillet, Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang, Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang, Brian Zhang, Eddie Zhang, and Shengjia Zhao. gpt-oss-120b gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

[47] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

[48] Nathan Godey, Éric Clergerie, and Benoît Sagot. Anisotropy is inherent to self-attention in transformers. In Yvette Graham and Matthew Purver, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 35–48, St. Julian's, Malta, March 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.eacl-long.3. URL https://aclanthology.org/2024.eacl-long.3/.

[49] Mario Lezcano-Casado. Trivializations for gradient-based optimization on manifolds, 2019. URL https://arxiv.org/abs/1909.09501.

A Isotropy in Router Inputs

Table 5: Router-input IsoScore, averaged across MoE layers. IsoScore is bounded in $[0, 1]$ with 1 corresponding to perfect isotropy. Both our models and open-source MoE LMs produce router inputs with IsoScore ≈ 0.92–0.97 across most layers, indicating that the isotropy assumption underlying SIMBAL's load-balancing argument holds tightly across architectures and scales.

Model	Method	IsoScore (mean $\pm$ std)
Our models		
MOE-M (D_M=768, 8 layers)	Balance loss	0.925 ± 0.019
	SIMBAL	0.919 ± 0.013
MOE-L (D_M=1536, 12 layers)	Balance loss	0.951 ± 0.014
	SIMBAL	0.947 ± 0.015
Open-source models		
OLMoE (D_M=2048, 16 layers)	—	0.966 ± 0.006
GPT-OSS-20B (D_M=2880, 24 layers)	—	0.929 ± 0.036
Qwen3-MoE (D_M=2048, 48 layers)	—	0.950 ± 0.027

We tabulate the IsoScores for MoE-M and MoE-L models in Table 5.

Setup. For each MoE layer, we collect the router input $x \in \mathbb{R}^{D_M}$ over $4{,}096{,}000$ tokens drawn from our validation set and compute IsoScore [15] on the resulting empirical distribution. IsoScore is defined on the centered covariance and is bounded in $[0, 1]$, with 1 corresponding to a covariance proportional to the identity.

Findings. Four observations follow from Table 5. First, router inputs are highly isotropic across all configurations, with mean IsoScore between 0.92 and 0.97. This is substantially higher than values previously reported for BERT hidden states [44, 45], and tight enough that the second-moment argument in Section 3.2 operates in a regime where the assumption $\mathrm{Cov}(x) \approx \sigma^2 I_{D_M}$ is well-supported. This suggests that MoE models may have a very different latent space structure before the router inputs than prior work has suggested. Second, SIMBAL and the auxiliary balance loss produce comparable IsoScore distributions: means differ by at most 0.006 and per-layer standard deviations overlap. This indicates that SIMBAL achieves balance through its structural constraint on R rather than by reshaping the upstream representation; the router input is already approximately isotropic regardless of the balancing mechanism used. Third, the larger model is somewhat more isotropic than the smaller one—MoE-L sits roughly 0.03 higher in IsoScore than MoE-M—suggesting the isotropy assumption underlying our argument should remain valid as D_M grows. Fourth, isotropy is not a peculiarity of our training setup: we measure IsoScore on three open-source MoE LMs—OLMoE [8], GPT-OSS-20B [46], and Qwen3-MoE [47]—and find mean IsoScore between 0.93 and 0.97, comparable to or higher than our trained models. This suggests approximate isotropy of router inputs is a general property of MoE language models at scale.

Per-layer variation. While layer-averaged IsoScores are high, individual layers vary. In MoE-L, the first and last MoE layers exhibit lower IsoScore (0.92–0.94), consistent with prior reports that representation anisotropy concentrates near the input and output of transformer stacks [48]. The same pattern appears in the open-source models: GPT-OSS-20B drops to 0.82 at layer 22, and Qwen3-MoE has two outlier layers (0.82 at layer 2, 0.88 at layer 47), with both models otherwise sitting near 0.95. The argument in Section 3.2 should therefore be understood per-layer: each MoE router operates on its own input distribution, and even the worst-case layers in our models satisfy the isotropy assumption tightly enough for the structural balancing argument to bind.

B Mitigating Expert Collapse

A central failure mode in MoE training is *expert collapse*, where the router concentrates assignments onto a small subset of experts while the remainder receive few or no tokens. Once underutilized, these experts cease to receive meaningful gradient signal, and the model effectively reverts to a dense network with a fraction of its nominal capacity. Beyond the lost capacity, collapse compounds with

Table 6: Performance across three scaling coefficients to SIMBAL. We find that the differences are not significant enough to warrant hyperparameter tuning.

Model	MoE-M	MoE-M	MoE-M
Coefficient	1.0	0.1	0.01
Perplexity ↓	13.716	13.685	13.687
Min PES ↓	0.0045	0.0044	0.0050

Table 7: Dynamic-K expert stage 3 selection [36] synergy with SIMBAL vs. LBL (perplexity and runtime on a full validation run, MoE-L). SIMBAL is able to preserve more performance when lower weighted experts are dropped for faster runtime.

Dropped $P(E)$ <	**SIMBAL (PPL)**	**SIMBAL (s)**	**LBL (PPL)**	**LBL (s)**
0	8.317	620.927	8.536	619.657
0.1	8.364	571.147	8.542	575.121
0.15	8.598	503.065	8.621	543.027
0.2	9.380	472.915	9.057	495.200

itself: the experts that do receive tokens are reinforced, while idle experts drift further from the active routing distribution, making recovery increasingly unlikely as training progresses. Any viable balancing method must therefore prevent collapse from setting in early.

Figure 5 tracks the number of unique experts activated on our 77M-token validation set throughout training. Without any balancing loss, routing collapses rapidly at both scales: the model converges to a small subset of experts within the first few thousand steps, and the run was truncated once it became clear that training would not recover. LBL and SIMBAL, in contrast, both maintain full expert utilization throughout training. This confirms that SIMBAL's semi-orthogonality constraint, despite acting on a fundamentally different quantity than LBL, is sufficient to prevent collapse. The benefits of SIMBAL documented in earlier sections — lower redundancy, more stable routing, better pruning behavior — thus come without sacrificing the basic guarantee that load balancing is meant to provide.

C Compute Estimates

To estimate total training FLOPs, we follow the approximation from Brown et al. [2], using $6 \times N \times T$ per forward pass, where N is the number of non-embedding active parameters and T is the number of training tokens.

For MoE-M and Dense-M, with 230M active parameters and 77M in embeddings, trained on 2×10^{10} tokens, this results in:

$$6 \times ((230 - 77) \times 10^6) \times 2 \times 10^{10} = \mathbf{1.836 \times 10^{19}} \text{ FLOPs}$$

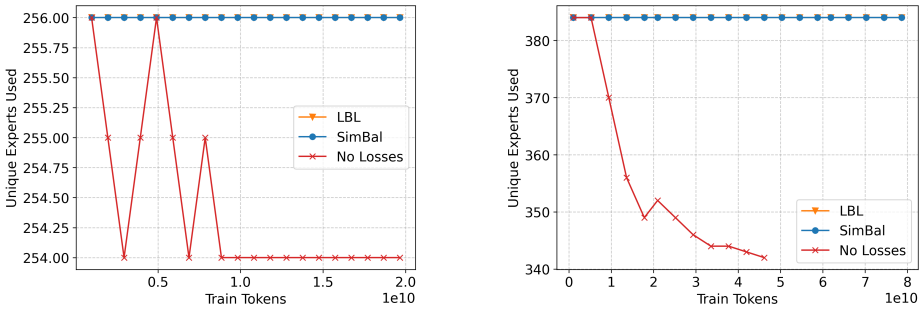
For MoE-L and Dense-L, with 761M active parameters and 154M in embeddings, trained on 7.8×10^{10} tokens, this results in:

$$6 \times ((761 - 154) \times 10^6) \times 7.8 \times 10^{10} = \mathbf{2.840 \times 10^{20}} \text{ FLOPs}$$

Full architectural details are available in Table 8.

Table 8: Parameters used for the model architecture and training. Parameter (active, total) counts include token embeddings. All MoE models have 32 experts, with the top 4 activated. P = parameters

	Dense-M	**MoE-M**	**Dense-L**	**MoE-L**
D_M	768	768	1536	1536
Depth	8	8	12	12
Heads	8	8	12	12
D_F	3072	768	6144	1536
RoPE θ	1e4	1e4	1e5	1e5
Peak LR	5e-4	5e-4	3e-4	3e-4
Embed P	77M	77M	154M	154M
Active P	230M	230M	761M	761M
Total P	230M	627M	761M	3.14B



(a) MoE-M (b) MoE-L

Figure 5: Expert utilization throughout training for MoE-M (left) and MoE-L (right), comparing LBL, our method (SIMBAL), and a baseline with no load balancing. We measure the number of unique experts activated on our full 77M-token validation set over time. Without any balancing (even with an orthogonal initialization), the expert routing collapses to a smaller set of experts. Both LBL and SIMBAL maintain full expert utilization and avoid expert collapse. The no-loss baseline was truncated early.

Table 9: Comparison of orthogonality preservation methods, average and standard deviation over 100 trials. We report the maximum deviation from orthonormality (**Max Dev**) and the mean L1 distance to the identity matrix (**L1 Dist**) after casting to our training precision. **Trained** refers to our loss-based method after 100 optimization steps. **Param** uses the orthogonal parameterization from Lezcano-Casado [49]. **OrthoInit** follows the initialization from Saxe et al. [17]. All matrices have shape 1536×32, matching our router dimensions. Best results in each column are bolded.

Method	Max Dev	L1 Dist
Trained	$\mathbf{1.03 \times 10^{-5} \pm 2.76 \times 10^{-6}}$	$\mathbf{8.52 \times 10^{-7} \pm 5.84 \times 10^{-8}}$
Param	$2.00 \times 10^{-4} \pm 2.31 \times 10^{-5}$	$4.80 \times 10^{-5} \pm 1.60 \times 10^{-6}$
OrthoInit	$1.93 \times 10^{-4} \pm 1.88 \times 10^{-5}$	$4.62 \times 10^{-5} \pm 1.79 \times 10^{-6}$

D Synthetic Orthogonalization Experiment

As PyTorch currently lacks support for orthogonal parameterizations in lower-precision formats commonly used to train language models (that we use), we perform orthogonalization in `float32`, and then cast the resulting matrix to `bfloat16`, our training precision. Our loss-based method trains the matrix directly in `bfloat16`. We report both the maximum and mean deviation from orthonormality, as well as the final loss values, in Table 9. We find that our loss consistently produces matrices that more closely approximate orthonormality than direct orthogonal parameterizations in our scenario. In fact, our approach matches or exceeds the throughput of efficient orthogonal parameterizations, while avoiding the need for expensive reorthogonalization steps. For this synthetic experiment, we train with AdamW (with no weight decay), and a learning rate cosine decayed from 1×10^{-4} to 1×10^{-5} over 100 consecutive steps. In our MoEs, we simply add our loss as an auxiliary loss term and update once per language model training step. We examine the coefficient sensitivity of SIMBAL to determine if tuning is necessary.

E Layer-Wise Orthogonalization

We provide tables for layer-wise orthogonalization performance for SIMBAL, and compare the results to LBL on MoE-M (Table 10) and MoE-L (Table 11). LBL alone does not orthogonalize the router whatsoever, while SIMBAL is able to achieve mean squared error similar to commonly used ϵ for numerical stability.

Router	SIMBAL	LBL
Layer 0 Router	1.94017e-10	0.00146701
Layer 1 Router	1.70156e-10	0.01486
Layer 2 Router	1.91267e-10	0.0155954
Layer 3 Router	1.89254e-10	0.0102319
Layer 4 Router	1.50925e-08	0.0100937
Layer 5 Router	2.99727e-08	0.0143029
Layer 6 Router	1.82301e-10	0.020765
Layer 7 Router	1.73648e-10	0.0258847

Table 10: Router orthogonality of MoE-M, as measured by $(R^T R - I)^2$

Router	SIMBAL	LBL
Layer 0 Router	1.49951e-08	0.0125956
Layer 1 Router	1.00854e-10	0.027788
Layer 2 Router	1.03228e-10	0.0183506
Layer 3 Router	4.47955e-08	0.0128958
Layer 4 Router	1.5001e-08	0.00668315
Layer 5 Router	9.38376e-11	0.00399825
Layer 6 Router	1.16159e-10	0.00375414
Layer 7 Router	2.99078e-08	0.00736187
Layer 8 Router	4.47949e-08	0.0200508
Layer 9 Router	2.99088e-08	0.0377724
Layer 10 Router	5.97087e-08	0.083971
Layer 11 Router	1.49907e-08	0.138501

Table 11: Router orthogonality of MoE-L, as measured by $(R^T R - I)^2$

F Loss-Free Load Balancing Combination

Loss-Free (LF) balancing [24] applies a direct bias to routing scores ($s = xR$, rather than routing weights $r = G(xR)$) without adding an auxiliary loss. Let f_i be the expert frequency in the current batch and $\bar{f} = 1/E$ the uniform target. Each expert's score is adjusted by a fixed scalar γ:

$$b_i' = b_i + \gamma \cdot \text{sign}(\bar{f} - f_i) \tag{7}$$

The scores are then used for computing the top-A experts with the new scores s_i:

$$s_i = xR + b_i \tag{8}$$

This encourages uniform expert assignment, but is not used in the weighting of the experts (r). It thus allows non-uniform expert weighting but still allocates experts uniformly over the full dataset. Additionally, γ is a hyperparameter that may need to be tuned, though the original authors recommend 0.001 since it provides a good balance between balancing while preventing fluctuations later in training.

Other work [10] use LBL in conjunction with LF for batch-wise load balancing, as they find that it can result in substantial imbalance in expert use sequence-wise. We do not include these results in earlier charts due to this extreme imbalance. Instead, in this section, we explore whether a combination with SIMBAL works similarly to LBL to improve sequence-wise balancing.

While the original authors of LF use sigmoid gating (over our softmax gating), we find that softmax gating is substantially more common in state-of-the-art work. Thus, to maximize relevance (regardless of performance), we additionally compare with softmax gating. The training setup for MoE-M remains identical to Section 3.3 otherwise.

We evaluated the balancing capabilities of this method using the MoE-M configuration, comparing its performance against both LBL and SIMBAL. We summarize our results in Table 12. We find that sigmoid gating leads to significant degradation in sequence-wise balance, especially compared to using only SIMBAL or LBL (as seen in Table 2). In exchange, there was a minor and possibly

Table 12: Model setup and performance. Sequence-wise Expert Utilization (SEU) is computed as the mean over the fraction of activated experts within a sequence. SIMBAL can improve sequence-wise balance without significant performance degradation, sometimes improving performance. All models use all experts throughout the full validation set, LF is the least balanced per-batch. While LBL asserts near-perfect balance, it also causes substantial perplexity degradation.

Model	MoE-M	MoE-M	MoE-M	MoE-M	MoE-M	MoE-M
Gating	Softmax	Softmax	Softmax	Sigmoid	Sigmoid	Sigmoid
Balancing	LF	LF+LBL	LF+SIMBAL	LF	LF+LBL	LF+SIMBAL
Perplexity ↓	13.708	14.154	13.695	13.618	14.015	13.637
SEU ↑	0.505	0.997	0.755	0.381	0.997	0.476

statistically insignificant (using the deviation values from Section 4.2. This is not ideal, as with larger models, when using model parallelism, extra consideration may be needed to ensure full utilization of all devices. Using LBL mitigates some of this, but leads to a substantial degradation in performance.

G Implementation Details

Here we provide some implementation details related to the auxiliary losses used in the paper in Figure 6. For our LBL baseline, we use an open-source repository implementation based on [38], available at lucidrains/st-moe-pytorch on GitHub. For both, we multiply the output of the function by the scaling coefficient if/where applicable during training. These losses can then be added to the final model loss (by adding them), or included using the AddAuxiliaryLoss autograd trick used in DeepSeek's modeling_deepseek.py on HuggingFace. Please note that in PyTorch, matrices are transposed relative to our notation, so the matmul is valid.

```python
import torch
from einops import reduce

# LBL
def balance_loss(gates: torch.Tensor) -> torch.Tensor:
    batch_size, num_tokens, num_experts = gates.shape

    # bal_loss = E * sum(f_i * P_i), expert i
    expert_mask = gates > 0.0
    f_i = reduce(expert_mask.float(), "b t e -> b e", "mean")
    P_i = reduce(gates, "b t e -> b e", "mean")
    loss_per_batch = num_experts * torch.sum(f_i * P_i, dim=-1)
    return loss_per_batch.mean()

# SimBal
def simbal_loss(router_linear, p=1):
    w = router_linear.weight
    w_ortho = torch.matmul(w, w.T)
    eye = torch.eye(w.shape[0], device=w.device)
    loss = torch.norm(w_ortho - eye, p=p)
    return loss
```

Figure 6: Python implementations of the LBL and SimBal loss functions.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We back all claims with their own experiments.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include a limitations section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: We primarily provide a mathematical intuition behind the reason load balancing works and provide empirical evidence, this is not a theory paper.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Hyperparameters, architectural details, datasets, and loss function implementations are provided.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We provide implementations for the loss functions in Section G and use open-source repositories as a base for the project, as we describe in 3.3. However, certain tools used to execute the experiments cannot be open-sourced.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Yes, in Section 3.3.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We explicitly address statistical significance with a Cohen's d value in Section 4.2.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the code of ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our work is more foundational in nature rather than application-driven.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We defer dataset release safety risks to the original authors of the DCLM paper.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All licences are respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not release new assets. The code in the appendix is documented.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were used as a writing and programming aid at times. However, they did not create the methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.