ONE MUST IMAGINE EXPERTS HAPPY: REBALANCING NEURAL ROUTERS VIA CONSTRAINED OPTIMIZATION

Kushal Thaman
Department of Computer Science
Stanford University
Stanford, CA 94305, USA
kushalt@stanford.edu

ABSTRACT

Mixture-of-Experts models promise scalable capacity by routing tokens to a sparse set of expert networks, but imbalanced routing (i.e. routing collapse) can degrade performance and hinder distributed expert parallelism. Conventional remedies add an auxiliary load-balancing loss, but this introduces conflicting gradients that hamper primary optimization. Recent bias-based routing strategies avoid auxiliary losses by dynamically adjusting per-expert biases that are updated using the sign of deviation from mean load. This requires careful tuning, is still susceptible to load fluctuations and suboptimal load utilization, and can lead to oscillations ("gating thrash") if mistuned. We propose *Dual Unified Ascent for Load-balancing* (DUAL), a technique that recasts router load balancing as a constrained optimization problem. By learning per-expert bias updates derived from Lagrange dual variables, DUAL adjusts gating bias increments proportional to each expert's load error with a damping term that prevents bias overshoot, combined with a differentiable router using a sparsemax function for the gating logits. Experimental results on 330M-parameter language models demonstrate that DUAL attains more uniform expert utilization without sacrificing performance, consistently reduces router imbalance, and slightly outperforms the performance of state-of-the-art Mixture-of-Expert techniques.

1 INTRODUCTION

Scaling language models via Mixture-of-Experts (MoE) increases capacity without a linear rise in computation by routing each token to a sparse gated subset of experts (Shazeer et al. (2017)), but ensuring that tokens are evenly distributed across experts remains a critical challenge. For an input token $\mathbf{u}_t$, the standard MoE layer computes:

$$\mathbf{h}_t = \mathbf{u}_t + \sum_{i \in \mathcal{K}_t} g_{i,t}\, \mathrm{FFN}_i(\mathbf{u}_t),$$

where $\mathcal{K}_t$ is the set of experts selected by a gating network and

$$g_{i,t} = \frac{s_{i,t}}{\sum_{j \in \mathcal{K}_t} s_{j,t}}, \quad s_{i,t} = \sigma(\mathbf{u}_t^\top \mathbf{e}_i).$$

Imbalanced routing leads to routing collapse with a few experts consistently overloaded while others remain underutilized, and to severe computational bottlenecks in distributed settings. Early solutions proposed auxiliary load-balancing losses (with Switch Transformer, described in Fedus et al. (2021) and GShard, described in Lepikhin et al. (2020)) to penalize disparities in expert usage. Although these losses do enforce balance, they inject interfering gradients that degrade the main task optimization. Recent auxiliary-loss-free methods eliminate these interfering gradients by adjusting per-expert biases heuristically (Dai et al. (2024)), e.g. with $s'_{i,t} = s_{i,t} + b_i$ and update these biases with a simple sign update $b_i \leftarrow b_i + \gamma\ \mathrm{sign}(\bar{c} - c_i)$, where c_i is the token count for expert i and $\bar{c}$ is the average load. Although effective compared to auxiliary strategies, this update uses only the sign and ignores the magnitude of the error.

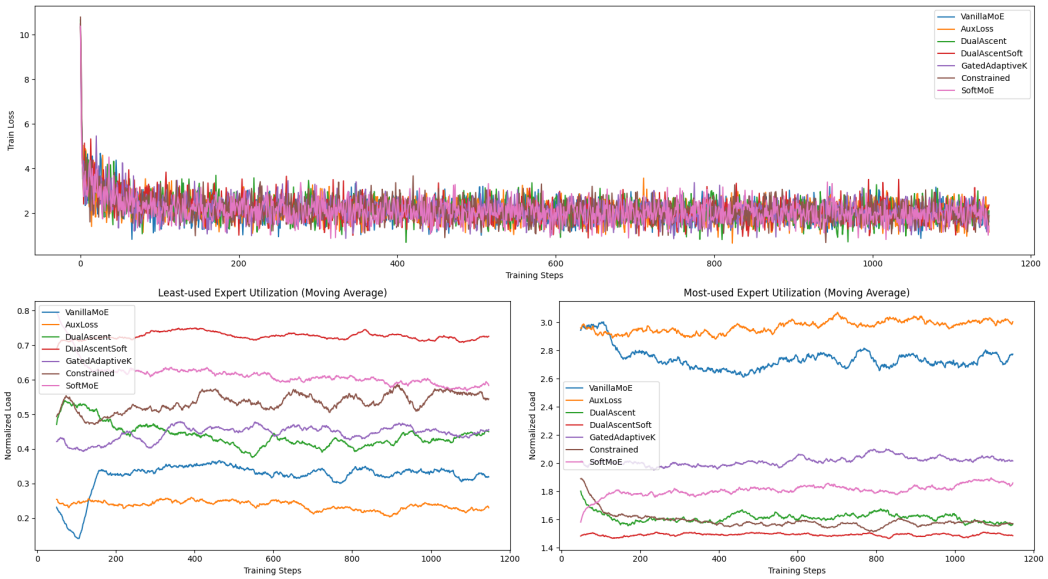


Figure 1: Comparison of different methods for MoE load balancing. Each method converges roughly to the same training loss and model performance, but we demonstrate a large disparity in the success of load balancing: our proposed DualAscent technique keeps all experts active, closer to the idea normalized load ($r_i = \frac{c_i}{\bar{c}}$ for expert i, and $\bar{c}$ is the average token count) of 1 for the least and most used experts, compared to other techniques that allow some experts to be underutilized.

Further, we found that even this strategy exhibits various limitations: it often yields a suboptimal maximal and minimal expert usage ratio (with overloaded experts persisting in various regimes), and these bias update rules—commonly based on sign or proportional corrections—lack a rigorous mathematical foundation. Such heuristically assigned expert bias updates often result in unstable or oscillatory behavior, and fail to guarantee convergence to a truly balanced load distribution.

2 METHODS

2.1 CONSTRAINED OPTIMIZATION FORMULATION

We view token-to-expert assignment as minimizing the model's negative log-likelihood while enforcing balanced expert usage and capacity limits. For a batch of N tokens where each token selects K experts on average, the ideal (target) load per expert is $n^* = \frac{N \cdot K}{E}$, with n_j denoting the number of tokens assigned to expert j. Because routing decisions are discrete, we relax them to soft assignments $p_{i,j} \in [0, 1]$ (with $\sum_j p_{i,j} = 1$ and at most K nonzero entries per token). We use sparsemax instead of softmax to promote sparsity. Thus, the expected load per expert is $n_j = \sum_i p_{i,j}$, and perfect balance means $n_j \approx n^*$ for all j. We express the constrained problem via the Lagrangian:

$$\mathcal{L} = L_0(\{p_{i,j}\}) + \sum_{j=1}^{E} \lambda_j (n_j - n^*) + \sum_{j=1}^{E} \mu_j (n_j - C_j),$$

where L_0 is the primary loss (e.g. negative log-likelihood), λ_j are Lagrange multipliers enforcing balance, and μ_j penalize capacity violations (active when $n_j > C_j$). Intuitively, λ_j acts as an expert-specific bias: if $n_j > n^*$, a positive λ_j increases the cost, discouraging further assignments; if $n_j < n^*$, a negative λ_j incentivizes more routing. At optimality, the KKT conditions yield that for any token i and experts j, k in its support,

$$\frac{\partial L_0}{\partial p_{i,j}} + \lambda_j + \mu_j = \frac{\partial L_0}{\partial p_{i,k}} + \lambda_k + \mu_k.$$

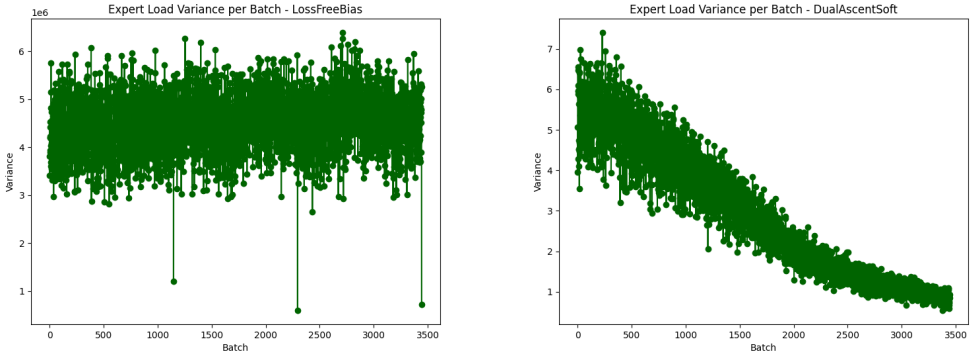


Figure 2: Batch-wise expert load variance for (left) LossFreeBias and (right) DualAscentSoft methods. Lower variance implies more uniform token distribution across experts. The sign-based update (LossFreeBias) exhibits higher and more erratic variance, while DualAscentSoft steadily drives variance down over training (batch index) and demonstrates stable convergent load balancing.

Differentiating the Lagrangian with respect to the dual variable (bias) λ_j

$$\frac{\partial \mathcal{L}}{\partial \lambda_j} = n_j - n^*$$

This gradient reflects how far expert j's load deviates from the target a natural update for λ_j is to move in the direction of this gradient $\lambda_j \leftarrow \lambda_j + \eta(n_j - n^*)$. This update step iteratively reduces load imbalance by increasing the bias for underloaded experts (encouraging more assignments) and decreasing it for overloaded ones, thus driving the system toward the balance constraint $n_j \approx n^*$.

2.2 Dual Ascent Bias Update

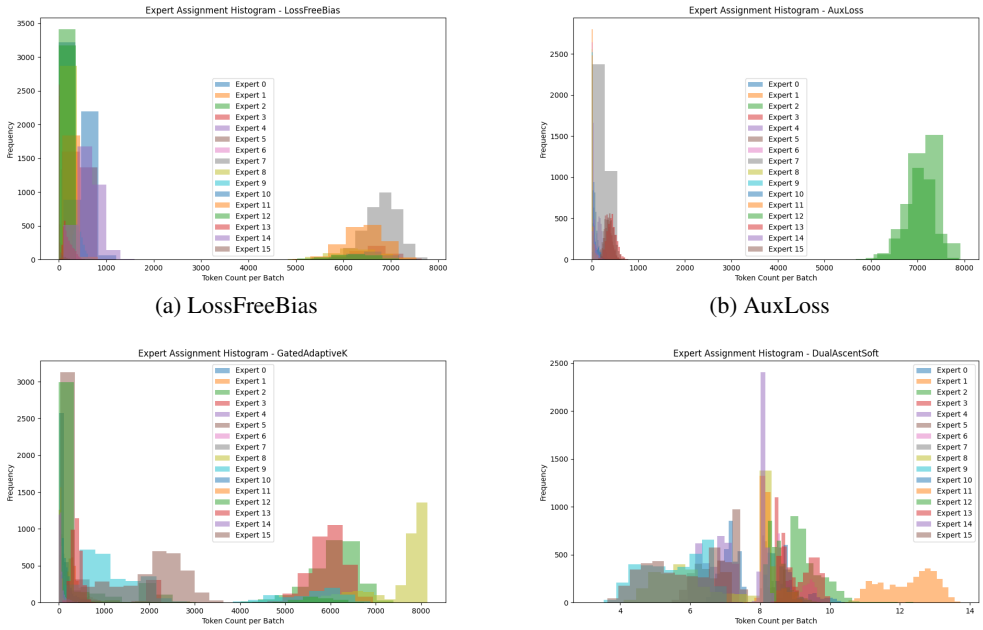
This formulation motivates our proposed *Dual Unified Ascent for Load-Balancing* (DUAL) bias update: we treat the biases as dual variables to iteratively drive each expert's load toward n^*.

$$b_i \leftarrow b_i + \eta\Big((\bar{c} - c_i) - \lambda\, b_i\Big).$$

where η is the step size and λ is a damping factor. This scales the correction with the actual load difference $(\bar{c} - c_i)$, the $-\lambda\, b_i$ term prevents bias values from growing unbounded. The motivation for a proportional update over a sign one as proposed in Dai et al. (2024) is that a sign update ignores the magnitude of the imbalance. A slight deviation from average load receives the same correction as a severe one, so biases can drift indefinitely and oscillate. If c_i is barely below $\bar{c}$, the sign update causes the same bias increment to be applied as if c_i were far below $\bar{c}$. This can lead to so-called *chattering* or *thrashing* around the equilibrium point. Since the step size is independent of the error magnitude, biases can drift indefinitely without a diminishing force bringing b_i back to zero once $|b_i|$ grows large. From a constrained optimization perspective, sign updates are effectively a subgradient for an L_1-type penalty with discontinuous corrections and does not solve for a precise load-balancing equilibrium. DUAL is a proportional update that behaves like a gradient step on an L_2-style objective which can produce a stable, smoothly adjusting solution converging to a well-defined fixed point.

2.3 Differentiable Routing via Sparsemax

Routing decisions, as in Dai et al. (2024) are typically non-differentiable and use a straight-through estimator or rely on the auxiliary loss to provide a continuous training signal to the gating network. To allow direct end-to-end training of the gating network without auxiliary losses, we adopt the sparsemax activation function for the router. Sparsemax (Martins & Astudillo (2016)) is a soft alternative to softmax that produces sparse probability outputs. Given a vector of gating logits for



(a) LossFreeBias (b) AuxLoss

(c) GatedAdaptiveK (d) DualAscentSoft

Figure 3: (a) LossFreeBias, (b) AuxLoss, (c) GatedAdaptiveK, (d) DualAscentSoft. Each histogram visualizes the distribution of token counts assigned to an expert in a single batch aggregated over training. Narrower and more centralized histograms indicate more uniform usage. DualAscentSoft yields visibly tighter distributions, while other methods yield multiple batches fed to experts with 0 token counts, a symptom of severe underutilization.

a token i, $\mathbf{z}_i = (z_{i,1}, \ldots, z_{i,E})$ (after adding the expert biases b_j to the raw scores), the sparsemax function is defined as:

$$\text{sparsemax}(\mathbf{z}_i) = \arg\min_{\mathbf{p}_i \in \Delta^E} \|\mathbf{p}_i - \mathbf{z}_i\|^2,$$

where $\Delta^E = \{\mathbf{p} \in \mathbb{R}^E_{\geq 0} : \sum_j p_j = 1\}$ is the probability simplex. In our setup, for a token with hidden representation h, let the gating network output be $G(h)$. We compute the sparse, adjusted routing probabilities as $\pi(h) = \text{sparsemax}\Big(G(h) + b\Big)$.

The result of sparsemax is that a token's probability mass is concentrated on the top-scoring experts, and lower-scoring experts receive exactly zero assignment. Sparsemax projects low-confidence experts to zero weight instead of assigning every expert a positive fraction (however small) as with softmax.

3 Experiments & Results

3.1 Setup

We use the **Wikitext-2-v1** dataset (Merity et al. (2016)) as our primary corpus. Texts are tokenized with the `bert-base-uncased` tokenizer, yielding a vocabulary of 30,522 tokens. For our large-scale runs, we set the maximum sequence length to 1024 tokens, so that each training example is either truncated or padded to 1024 tokens. We employ a Transformer-based language model in which the feed-forward layers are replaced with MoE layers.

We evaluate a number of variants to compare the effectiveness of the proposed techniques, comparing:

- **LossFreeBias:** This is our baseline sign update bias method as implemented in Dai et al. (2024).

MoE Variant	Validation Loss
`LossFreeBias`	1.2995
`AuxLoss`	1.3029
`DualAscent`	1.2975
`DualAscentSoft`	**1.2709**
`GatedAdaptiveK`	1.2894
`Constrained`	1.3070
`JustSparsemax`	1.2985

Table 1: DualAscentSoft method achieves a marginally better validation perplexity while also considerably improving load-balancing on WikiText.

- **AuxLoss:** This is an MoE with a standard mean-squared auxiliary load-balancing loss.
- **DualAscent:** Here we implement Dual Ascent per-expert bias update.
- **DualAscentSoft:** This is the proposed **DUAL** technique: a DualAscent update combined with the differentiable router.
- **GatedAdaptiveK:** Beyond static top-K routing, we implemented this variant that dynamically adjusts the number of active experts per token. When the margin between the K^{th} and $(K+1)^{\text{th}}$ logits is small, the router activates an extra expert.
- **Constrained:** We test a variant where we enforce explicit capacity constraints by hard-capping the maximum number of tokens per expert such that no expert is overloaded beyond its capacity, i.e. dropping excess tokens so that n_j never exceeds some C_j.
- **JustSparsemax:** To isolate the effect of routing sparsity, we also evaluate a variant that uses sparsemax-based routing with fixed (zero) biases.

4 Conclusion

In this work, we presented a novel auxiliary-loss-free load balancing strategy for Mixture-of-Experts (MoE) models termed *Dual Unified Ascent for Load-Balancing* (DUAL). We recast the expert load balancing as a constrained optimization problem, updating learnable bias terms via a dual ascent rule with damping and a differentiable sparsemax-based gating function. Our experiments on a 330M-parameter MoE model trained on 10-100B tokens demonstrate that DUAL achieves slightly lower validation perplexity while maintaining a near-ideal load distribution and reduced batch-wise load variance.

5 Limitations

The dual ascent update relies on carefully tuned hyperparameters, and insufficient tuning can still lead to oscillatory bias adjustments or cause slow convergence. While our results are promising, the current experiments are performed at a moderate scale. Future work will extend this investigation to larger models and diverse datasets, further validating the robustness and scalability of our approach in distributed, expert-parallel settings. A wider suite of evaluations (such as testing on Humanity's Last Exam (Phan et al. (2025))) is a next step, and although the dual ascent soft updates reduce load variance considerably, some residual imbalance can persist in highly heterogeneous data scenarios, suggesting room for further refinement.

REFERENCES

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

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2101.03961.

Dmytro Lepikhin et al. Gshard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Machine Learning (ICML)*, 2020. URL https://arxiv.org/abs/2006.16668.

André FT Martins and Ramón Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 1614–1623. PMLR, 2016.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, Michael Choi, Anish Agrawal, Arnav Chopra, ..., Alan Zhou, Aidan Wu, Jason Luo, Anwith Telluri, Summer Yue, Alexandr Wang, and Dan Hendrycks et al. Humanity's last exam, 2025. URL https://arxiv.org/abs/2501.14249.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey Hinton, and Naveen Rao. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://arxiv.org/abs/1701.06538.

A APPENDIX: DUAL ASCENT SOFT BIAS UPDATE

Algorithm 1 summarizes our dual ascent soft update: This update ensures that experts that are over-utilized receive a negative adjustment while under-utilized ones get a positive correction, all without interfering with the main gradient flow.

A.1 CONSTRAINED OPTIMIZATION FORMULATION

Formulation as Constrained Optimization We formulate the token-to-expert assignment as an optimization problem with the objective of minimizing the model's negative log-likelihood subject to the condition that each expert sees an equal share of tokens and does not exceed its capacity. For a given batch of tokens, let n_j be the number of tokens assigned to expert j. In an ideal balanced routing, n_j should be close to the *target load* $n^* = N \cdot K/E$ (with batch size N, and each token selects K experts on average) or at least not exceed a capacity C_j (which may be set to $C_j = \lceil n^* \rceil$ or $n^*(1+\epsilon)$ for some tolerance ϵ). The routing decisions are discrete (each token either goes to an expert or not), so directly enforcing these constraints is combinatorially hard. We relax the problem by considering soft assignments $p_{i,j} \in [0,1]$ indicating the fraction or probability of token i's assignment to expert j. Each token's assignments must satisfy $\sum_j p_{i,j} = 1$ (as each token is fully allocated across experts) and at most K of the $p_{i,j}$ can be non-zero (sparsity). In a conventional softmax gating, $p_{i,j}$ would be a dense probability distribution; we will instead use sparsemax to ensure most $p_{i,j}$ are zero. The expected load of expert j is then summing over tokens in the batch $n_j = \sum_i p_{i,j}$. Perfect balance means $n_j \approx n^*$ for all j.

We write the Lagrangian of the constrained routing for one batch as:

$$\mathcal{L} = L_{\text{main}}(\{p_{i,j}\}) \; + \; \sum_{j=1}^{E} \lambda_j \big(n_j - n^*\big) \; + \; \sum_{j=1}^{E} \mu_j \big(n_j - C_j\big),$$

where L_{main} is the primary training objective (e.g. negative log-likelihood of the batch given the expert assignments), λ_j are Lagrange multipliers enforcing the balance constraints, and μ_j are Lagrange multipliers enforcing the capacity constraints. The μ_j terms only activate when $n_j > C_j$ and can be seen as slack variables penalties. In practice, we handle capacity with a hard algorithmic step, as described later, but we include μ_j here for completeness of the formulation.

Intuitively, λ_j can be interpreted as an expert-specific "cost" or bias: if expert j is over-assigned ($n_j > n^*$), then to satisfy the constraint the term $\lambda_j(n_j - n^*)$ will increase the Lagrangian (if λ_j is positive), making it beneficial to reduce assignments to j. Conversely, if j is under-assigned ($n_j < n^*$), a properly adjusted λ_j would be negative, encouraging more assignments to that expert. At optimum, the Karush–Kuhn–Tucker conditions would require that the gradient of $\mathcal{L}$ with respect to the assignment probabilities $p_{i,j}$ is zero for interior solutions. This yields a condition on the effective gating scores: for any token i and any two experts j and k that are both in the support of p_i (i.e. could be selected), we would have

$$\frac{\partial L_{\text{main}}}{\partial p_{i,j}} + \lambda_j + \mu_j = \frac{\partial L_{\text{main}}}{\partial p_{i,k}} + \lambda_k + \mu_k.$$

A.2 Model Architecture

Our backbone is a Transformer-based language model in which the standard feed-forward network (FFN) is replaced by an MoE layer. The key architectural specifications for our *big model* configuration are as follows:

- **Embedding Layer:**
  - Vocabulary size: 30,522
  - Embedding dimension: 2048
- **MoE Layer:**
  - Hidden dimension: 4096
  - Number of experts: 16 (we also experiment with variants such as 64 experts)
  - Top-K selection: $K = 2$
  - Load-balancing method: We compare several variants, including our baseline *LossFreeBias* (sign update), *AuxLoss*, *DualAscent*, *DualAscentSoft* (our key contribution), *GatedAdaptiveK*, *Constrained*, and *JustSparsemax*.
- **Residual Projection and Output Layer:**
  - A residual projection is applied if the embedding and hidden dimensions differ.
  - The output layer maps from the hidden dimension (4096) back to the vocabulary size.

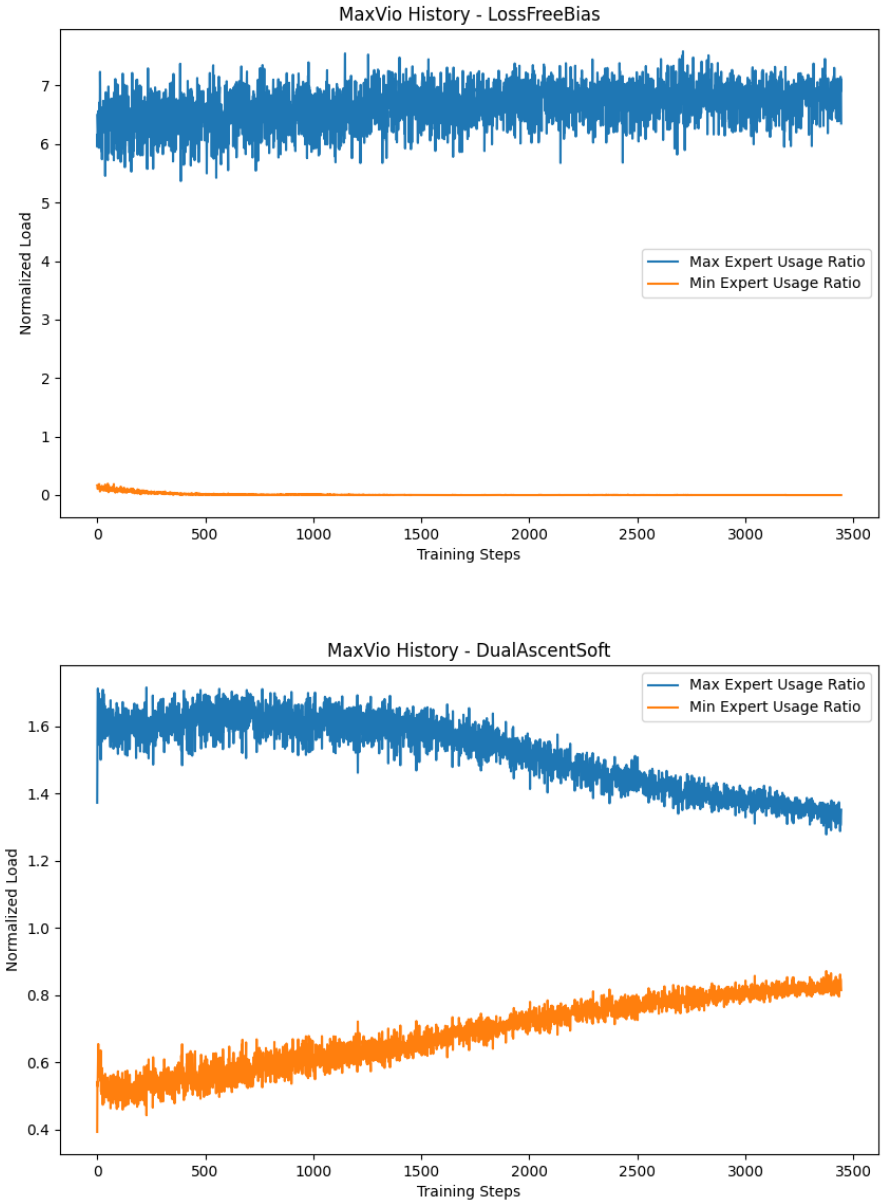
A.3 Training Details

All models are trained on the Wikitext-2-v1 corpus. For our large-scale experiments we simulate a setting with extended sequence lengths (1024 tokens) and use a training schedule designed for efficiency:

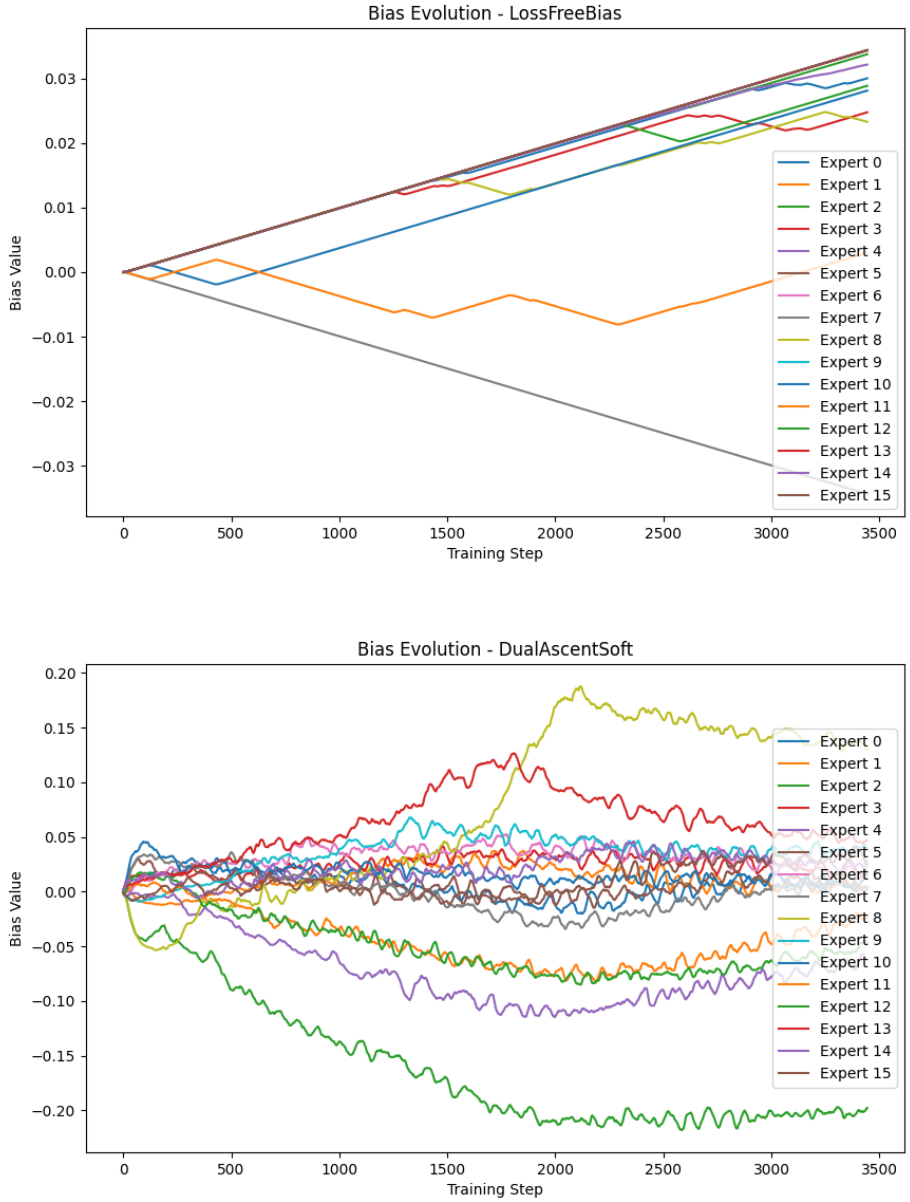
- **Number of Epochs:** We train for 5 epochs.
- **Batch Size:** 64 sequences per batch.
- **Learning Rate:** 1×10^{-3} with the Adam optimizer.
- **Auxiliary Loss Coefficient (if used):** 0.01.
- **Bias Update Parameters:** For methods employing external bias updates, we use an update rate of 1×10^{-5} and a momentum factor of 0.9. For our dual ascent updates, we set the dual ascent step size (η) to 1×10^{-5} and the damping factor (λ) to 1×10^{-2}.

If multiple GPUs are available, we leverage PyTorch's `DataParallel` to distribute computation. All experiments are run on our cluster setup using 4 A100 NVIDIA GPUs.

A.4 Additional results



(a) **Worst-case expert–load ratio over training.** (a) *LossFreeBias*: the maximum load ratio (blue) hovers between 6–7× the mean while the minimum ratio (orange) collapses to ≈ 0, showing persistent routing collapse and no sign of recovery. (b) *DualAscentSoft*: dual-ascent updates quickly damp the imbalance— the max ratio decays from $\approx 1.7 \rightarrow 1.35$ and the min climbs from $\approx 0.5 \rightarrow 0.85$—yielding a 4× tighter spread and a smooth convergence toward the ideal value 1.0.



(a) **Trajectory of learned per-expert biases** b_j. (a) *LossFreeBias*: fixed-step, sign-only updates drive almost perfectly linear bias drift with small amplitude ($|b_j| < 0.04$); the rule cannot generate sufficient leverage to correct the 6× load imbalance in figure. (b) *DualAscentSoft*: proportional dual-ascent updates move faster when an expert is far from target and slow as equilibrium is approached, producing damped oscillations that settle near $b_j \approx 0$ within a bounded range ($\approx \pm 0.2$). The adaptive behaviour lets the router respond aggressively to large errors while avoiding runaway bias growth.