STOCHASTIC ONLINE LINEAR PROGRAMMING FOR RESOURCE-AWARE EXPERT ROUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-experts (MoE) models decouple parameter count from per-token compute, but require a router that matches tokens to experts while balancing the load across experts. We formulate MoE routing as a *stochastic online linear program* (SOLP), which allows a flexible formulation of matching-quality versus expert-balance tradeoffs through linear constraints. Specifically, while our formulation easily admits the traditional cardinality constraints on experts-per-token or tokens-per-expert, it can also capture substantially more flexible aggregate constraints on expert utilization across sequences, layers or even a whole training batch. Furthermore, we provide an elegant dual update algorithm, which finds a near-optimal solution as we establish in our main theoretical guarantee. Empirically, we evaluate various instantiations of our general SOLP framework across a seven-scale ladder built on the Gemma architecture, ranging from 183M to 6.59B parameters (6 to 30 layers), and show that the flexibility afforded by our formulation results in better validation perplexity across this ladder, when compared with standard Top-k routing approaches. We further show additional benefits of our dual update algorithm, where the dual parameters can be cheaply retrained for a new set of constraints at inference-time, yielding a powerful mechanism for inference time compute optimization, without any significant quality losses.

1 INTRODUCTION AND CONTRIBUTION

The scaling paradigm for large language models has been significantly boosted by the development of mixture-of-experts (MoE) techniques (Jacobs et al., 1991; Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022). At a high level, MoE models allow tokens in the feed-forward layers to be processed not by a single parameterized function but instead by a *sparse mixture* of functions, known as experts. This provides the model designer with a powerful lever: one can increase the total number of parameters—and hence the representational capacity—of the network with only a modest increase in the FLOPs required to process a single token. This decoupling of parameter count from per-token compute has provided enormous benefits, enabling models such as Mixtral (Jiang et al., 2024), DeepSeek-V2/V3 (DeepSeek-AI, 2024; 2025), and others to achieve strong performance at substantially reduced training cost.

On the other hand, MoE models introduce a fundamental new challenge, the *routing problem*: given a sequence of input tokens and a set of experts at each layer, how should tokens be assigned to these experts? One major tool researchers have used to address this challenge is a learned router scoring mechanism, where each token is assigned a score indicating its affinity for each expert. With such scores available, a natural choice is to simply route each token to the k experts with the highest affinity, known as *top-k* or *tokens choose* routing (Shazeer et al., 2017). However, this approach has the unfortunate side-effect of *load imbalance*, where a small number of "popular" experts receive the majority of tokens while others sit idle (Shazeer et al., 2017; Lepikhin et al., 2021; Zoph et al., 2022), which can lead to training slowdowns (devices hosting popular experts become bottlenecks) and wasted model capacity (underutilized experts fail to specialize; see Riquelme et al. 2021; Chi et al. 2022). Such challenges can be mitigated with the use of *auxiliary load-balancing regularization* (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022; Zoph et al., 2022), while other researchers have proposed alternative approaches to expert routing (Riquelme et al., 2021; Zhou et al., 2022; Wang et al., 2024; Han & Zhong, 2025; Sun et al., 2026). One such approach, for example, is the *experts choose* strategy (Zhou et al., 2022), a "transpose" of tokens choose where each expert is assigned its highest-affinity tokens; this approach can create imbalance at the token level, and worse, it breaks the causal structure required by autoregressive models.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Taking a step back, we can view the MoE routing problem as a *constrained resource allocation problem*: we want to maximize the affinity of assigned experts while ensuring that each token is operated on by a suitable number of experts, no expert is over-assigned, and each expert receives sufficient load to remain viable. Through this lens we can formulate routing directly as a *linear program* (LP) to design principled assignment strategies, a perspective that recent literature has begun to explore (Lewis et al., 2021; Zhou et al., 2022; Wang et al., 2024; Han & Zhong, 2025; Sun et al., 2026). The excellent recent work of Han & Zhong (2025), for example, shows that DeepSeek's *auxiliary-loss-free load-balancing* (ALF-LB; Wang et al. 2024) can be understood as a primal-dual optimization algorithm operating across training steps with provable convergence guarantees.

In the present paper we will take this LP perspective significantly further, and indeed we show both theoretically and empirically that the primal-dual framework opens up a rich landscape of possibilities for allocating compute more efficiently in transformer models while simultaneously respecting hardware constraints and preserving fundamental properties such as autoregressive causality. Our framework, *Stochastic Online Linear Programming (SOLP) Routing*, is built around five core ideas:

- **Multi-dimensional constraints and index relaxation** (Sections 2–3): We model routing over a five-dimensional binary assignment tensor $X[t, e, l, b, n] \in \{0, 1\}$ indexing tokens t, experts e, layers l, batch sequences b, and training steps n. Every routing rule—existing heuristics such as Top-k (Shazeer et al., 2017), Expert Choice (Zhou et al., 2022), BASE Layers (Lewis et al., 2021), and ALF-LB (Wang et al., 2024), as well as our own—corresponds to constraining the values of X by fixing some axis indices while averaging over other axes. The Top-k rule, for example, pins the number of experts per token at k, i.e., $\sum_e X[t, e, l, b, n] = k$, but we can instead relax this constraint to spread expert FLOPs heterogeneously across the sequence, the batch, or even across the layers of the network.
- **Flexible optimization objectives** (Section 4): Alongside the standard linear objective of maximizing token–expert affinity scores, the same LP constraint structure supports an adaptive *top-P probability-mass* rule akin to nucleus sampling (Holtzman et al., 2020), where the target $P = 1 - \tau$ is learned directly via the dual variable of the compute budget to guarantee k experts on average.
- **Dual variables as shadow prices** (Section 4): Resource constraints are enforced through additive dual variables, or "prices," which act as per-expert score biases ($\lambda_{e,l}$, pricing congestion) and scalar admission thresholds (τ_l, pricing compute, see Sun et al., 2026; Raposo et al., 2024). We prove that a scalar price cannot reorder a fixed-cardinality choice (Proposition 2 in Appendix B), explaining why dual pricing only unlocks quality gains when paired with cardinality relaxation.
- **Decoupling training and inference constraints across timescales** (Section 4): During pre-training over diverse minibatches, we average constraints across batches and steps ($\bar{\Sigma}b, \bar{\Sigma}n$; Algorithm 1), avoiding sequential token scans and allowing individual sequences to specialize to domain-relevant experts. At inference time, however, hardware buffers demand strict within-step load balance; because SOLP separates learned router weights from dual control, we can swap in a causal token-level algorithm (`Sequence-OLP` or `Batch-OLP`; Algorithm 2) on the *same frozen checkpoint* to dynamically rebalance token–expert assignments and cut inference buffer overloads with negligible quality loss.
- **Compositional regret and violation bounds** (Section 5): While online routing approximates the offline LP optimum, our main theorem (Theorem 8 in Appendix D and Corollary 1 in Section 5) proves that score shortfall and constraint violations decompose *additively* across resource groups and decay at rate $\mathcal{O}(1/\sqrt{U})$ over U dual price updates—yielding $\mathcal{O}(1/\sqrt{N})$ across N training steps and $\mathcal{O}(1/\sqrt{T})$ across T inference tokens—even when each update aggregates an entire batch of routing decisions.

In Section 6, we train a controlled seven-rung ladder of Gemma-4 based MoE language models on C4 spanning 183M to 6.59B parameters (6 to 30 layers) with six paired seeds per arm. The training results confirm the theory step by step: replacing the auxiliary loss with fixed-k dual prices (`TopK-OLP`) is a null through 1.32B, whereas relaxing per-token cardinality at matched routed compute ($k_{\text{tar}} = 4.00$) improves held-out loss at every scale. Pooling the compute budget across layers (`VarK-Global`) scales monotonically with depth, spontaneously discovering the $2.6\times$ ascending depth schedule predicted by an offline LP oracle on frozen router scores (Appendix E.2). This recovers a substantial fraction of a model scale step at zero additional compute, while enabling post-training compute elasticity and zero-shot inference rebalancing via `Sequence-OLP` and `Batch-OLP`.

So far we surveyed the progression from auxiliary losses through loss-free biasing to the LP formulation. Next, we discuss several additional connections in more depth.

Token choice vs. expert choice. Top-k routing (Shazeer et al., 2017) and Expert Choice (Zhou et al., 2022) represent dual views of the same assignment problem: one fixes the number of experts per token, the other fixes the number of tokens per expert. When both capacities are constrained simultaneously, the problem becomes a balanced assignment, as in BASE Layers (Lewis et al., 2021). SOLP generalizes all three by allowing both token-side and expert-side constraints to be soft, with dual variables mediating their tradeoff.

Losses vs. prices. Auxiliary losses (Lepikhin et al., 2021; Fedus et al., 2022; Zoph et al., 2022; Chi et al., 2022) alter routing indirectly by producing gradients that change the router's parameters over many training steps. SOLP uses shadow prices (dual variables) added directly to router scores, modifying the routing decision *immediately* without changing network weights. ALF-LB (Wang et al., 2024; Han & Zhong, 2025) similarly adjusts a per-expert bias at each training step under fixed top-k selection. Our framework shows that such biases are formally dual variables of step-level expert balance constraints ($\mathsf{Bal}_{e,l}$); in our experiments, the fixed-k variant `TopK-OLP` directly instantiates this ALF-LB dual update (reproducing its gains over auxiliary losses at larger scales), while extending the same dual mechanism to relaxed per-token cardinality and cross-layer compute budgets (`VarK` and `TopP`) roughly doubles the quality improvement.

The optimization foundation. Our price updates are instances of online gradient descent (Zinkevich, 2003; Shalev-Shwartz, 2012; Hazan, 2016). Dual mirror descent provides guarantees for stochastic online resource allocation (Balseiro et al., 2020; 2023); we build on that foundation for the routing setting, with the additional structure of multiple resource groups and update schedules. Online LP (Agrawal & Devanur, 2014) and OCO with long-term constraints (Mahdavi et al., 2012; Jenatton et al., 2016; Yu & Neely, 2020) are also closely related, though they use different comparators and feasibility definitions. We state our allocation benchmark and assumptions explicitly.

Thresholds and variable computation. Several recent methods route a variable number of experts per token, echoing adaptive computation (Graves, 2016; Dehghani et al., 2019): Expert Threshold Routing (Sun et al., 2026) maintains an exponential moving average quantile threshold per expert, while DTop-p (Jin et al., 2025) uses a proportional–integral controller to adjust a nucleus-style probability threshold p per layer. The SOLP framework provides a unifying optimization view for understanding and extending both approaches: although neither was originally formulated as a linear program, our framework reveals that score-threshold routing (`VarK`) and adaptive top-P routing (`TopP`) emerge as exact Lagrangian duals of the *same* linear constraint hierarchy under two distinct primal objectives (Section 4 and Appendix C). This primal–dual perspective naturally equips both mechanisms with joint load-balance and compute-budget control, per-token guardrails $[k_{\min}, k_{\max}]$, cross-layer budget pooling ($\mathsf{Bud}_{\varnothing}$, generalizing binary depth allocation such as Mixture-of-Depths Raposo et al., 2024 and LayerSkip Elhoushi et al., 2024), and finite-sample regret and violation guarantees.

2 Mixture-of-Experts Layers and the Assignment Tensor

Mixture-of-experts (MoE) layer. Consider a transformer with L layers indexed by $l \in [L] := \{1, \dots, L\}$, processing batches of B sequences (indexed by $b \in [B]$) of length T tokens (indexed by $t \in [T]$) across N training or serving steps (indexed by $n \in [N]$). At layer l, there are E expert feed-forward networks $\{F_{e,l} : \mathbb{R}^d \to \mathbb{R}^d\}_{e=1}^{E}$. At the lth layer of the transformer, every token t is represented by a hidden state vector $u_{t,l,b,n} \in \mathbb{R}^d$, and in order to estimate the token–expert affinity a linear router with weight matrix $W_{r,l} \in \mathbb{R}^{E \times d}$ computes raw router logits and normalized routing probabilities:

$$z_{t,e,l,b,n} = (W_{r,l} u_{t,l,b,n})_e, \qquad q_{t,e,l,b,n} = \mathrm{softmax}(z_{t,:,l,b,n})_e = \frac{\exp(z_{t,e,l,b,n})}{\sum_{e'=1}^{E} \exp(z_{t,e',l,b,n})}. \tag{1}$$

From (1), the router produces affinity scores $c_{t,e,l,b,n} \in \mathbb{R}$ (e.g., routing probabilities $c_{t,e,l,b,n} = q_{t,e,l,b,n}$, log-probabilities $c_{t,e,l,b,n} = \log q_{t,e,l,b,n}$, or raw logits $c_{t,e,l,b,n} = z_{t,e,l,b,n}$).

The assignment tensor. Routing decisions are represented by the five-dimensional binary *assignment tensor* $X \in \{0,1\}^{T \times E \times L \times B \times N}$, where $X[t, e, l, b, n] = 1$ if and only if token t in sequence b at layer l and step n is routed to expert e. Given X, the MoE layer evaluates only the selected experts

$(X[t, e, l, b, n] = 1)$ and outputs the gated sum

$$y_{t,l,b,n} = \sum_{e=1}^{E} X[t, e, l, b, n]\, g_{t,e,l,b,n}\, F_{e,l}(u_{t,l,b,n}) \ ,$$

where $g_{t,e,l,b,n} \geq 0$ are the combine weights, typically chosen as $g_{t,e,l,b,n} = q_{t,e,l,b,n}$ although many alternatives have been proposed such as the L2-normalized $g_{t,e,l,b,n} = q_{t,e,l,b,n}^{1/2} / \sum_{e'} q_{t,e',l,b,n}^{1/2}$.

The typical MoE routing algorithm selects a fixed number k of experts for each token—tokens choose (Shazeer et al., 2017) is the canonical example—so that $\sum_{e=1}^{E} X[t, e, l, b, n] = k$ for all t, l, b, n. In this work, we instead specify a *target* k_{tar} denoting the target average number of active experts per token per layer, together with optional per-token guardrails $k_{\min}$ and $k_{\max}$ ($0 \leq k_{\min} \leq k_{\text{tar}} \leq k_{\max} \leq E$, where setting $k_{\min} = 0, k_{\max} = E$ removes the pointwise guardrail altogether) bounding the minimum and maximum number of experts any single token may be assigned at a layer.

3 Constraint Notation and Taxonomy of Routing Methods

Index annotations ($\cdot$ vs. $\bar{\Sigma}$). Every linear constraint we consider marginalizes $X[t, e, l, b, n]$ by averaging over a subset of its five slots while holding the remaining slots free. We write only two markers in each slot: a *free* index variable (t, e, etc.), or equivalently a simple "$\cdot$", or we use the symbol $\bar{\Sigma}$ which means the left hand side of the constraint is averaged over this axis. We name each constraint row $\mathsf{Name}_{\text{free slots}}$, so that its subscript explicitly lists the free $(\cdot)$ slots; the number of dual variables attached to any row is then the product of the cardinalities of the slots in its subscript.

The three constraint families (Card, Bud, Bal). Building off of previous work, all methods in this paper are built from three constraint families on $X[t, e, l, b, n]$.

- $\mathsf{Card}_{t,l,b,n}$ **(Per-token cardinality guardrails, un-averaged):**

$$X[t,\ \bar{\Sigma}e,\ l,\ b,\ n] \in \left[\tfrac{k_{\min}}{E},\ \tfrac{k_{\max}}{E}\right] . \tag{2}$$

  Since $X[t, \bar{\Sigma}e, l, b, n] = \frac{1}{E}\sum_{e=1}^{E} X[t, e, l, b, n]$, (2) states that *every individual token* (t, l, b, n) *must select between* $k_{\min}$ *and* $k_{\max}$ *experts*. Most prior work has assumed $k_{\min} = k_{\max} = k_{\text{tar}}$.

- Bud_l **and** $\mathsf{Bud}_{\varnothing}$ **(Pooled compute budget, averaged):**

$$\mathsf{Bud}_l :\ X[\bar{\Sigma}t,\ \bar{\Sigma}e,\ l,\ \bar{\Sigma}b,\ \bar{\Sigma}n] = \tfrac{k_{\text{tar}}}{E} , \qquad \mathsf{Bud}_{\varnothing} :\ X[\bar{\Sigma}t,\ \bar{\Sigma}e,\ \bar{\Sigma}l,\ \bar{\Sigma}b,\ \bar{\Sigma}n] = \tfrac{k_{\text{tar}}}{E} .$$

  In English, Bud_l requires the average expert usage across tokens, batch sequences, and steps to equal k_{tar} experts per token *at each layer* l *separately*. The relaxed version of this constraint is $\mathsf{Bud}_{\varnothing}$, where we require only that the *network-wide average* across all L layers equal k_{tar} experts per token, allowing compute to move across depth.

- Bal **(Expert load balance, averaged over tokens):**

$$\mathsf{Bal}_{e,l} :\ X[\bar{\Sigma}t,\ e,\ l,\ \bar{\Sigma}b,\ \bar{\Sigma}n] = \tfrac{k_l}{E} \qquad \left(\text{where } k_l \coloneqq \textstyle\sum_e X[\bar{\Sigma}t,\ e,\ l,\ \bar{\Sigma}b,\ \bar{\Sigma}n]\right) ,$$

  with within-step online variants $\mathsf{Bal}_{e,l,b,n}$ $(X[\bar{\Sigma}t, e, l, b, n] = k_l/E)$ and $\mathsf{Bal}_{e,l,n}$ $(X[\bar{\Sigma}t, e, l, \bar{\Sigma}b, n] = k_l/E)$.[1] In English, Bal requires every expert $e \in [E]$ at layer l to receive an equal $1/E$ share of layer l's traffic. When layer l's budget is fixed by Bud_l (or $\mathsf{Card} = k_{\text{tar}}/E$), $k_l = k_{\text{tar}}$, so the right-hand side is simply k_{tar}/E. When compute is shared across layers ($\mathsf{Bud}_{\varnothing}$), writing the right-hand side as layer l's own average $k_l/E = X[\bar{\Sigma}t, \bar{\Sigma}e, l, \bar{\Sigma}b, \bar{\Sigma}n]$ enforces exact $1/E$ balance among the E experts *within* layer l while allowing k_l itself to vary across layers subject to $\frac{1}{L}\sum_{l=1}^{L} k_l = k_{\text{tar}}$.

Taxonomy of prior and proposed routing methods. As summarized in Table 1, existing MoE routing rules correspond to specific choices of which slots in $X[t, e, l, b, n]$ are held free $(\cdot)$ versus averaged $(\bar{\Sigma})$. Table 2 then presents our routing taxonomy as progressive relaxations of these constraints. For direct comparison, the first two rows of Table 2 repeat the exact fixed-k baselines from Table 1—`TopK+Aux` (Fedus et al., 2022) and `TopK-OLP` (the linear dual-ascent form of

[1] We are slightly abusing notation here, as k_l is *not* a constant; on the other hand, having a linear function of $X[\cdots]$ on the right hand side of the constraint is perfectly valid in a linear program.

ALF-LB; Wang et al. 2024; Han & Zhong 2025), where $\mathsf{Card}_{t,l,b,n} = k_{\text{tar}}/E$ is enforced exactly by top-k selection and $\mathsf{Bal}_{e,l}$ is enforced via learned per-expert dual prices. Subsequent rows relax $\mathsf{Card}_{t,l,b,n}$ to the guardrail interval $[k_{\min}/E, k_{\max}/E]$ and control average compute via a pooled budget constraint, either per layer (Bud_l in `VarK-Layer` and `TopP-Layer`) or globally across all layers ($\mathsf{Bud}_\varnothing$ in `VarK-Global` and `TopP-Global`). Crucially, `VarK` and `TopP` share the *exact same constraint rows* in Table 2, and differ only in their primal routing objective: `VarK` maximizes standard token–expert affinity scores (Section 4), whereas `TopP` maximizes a rank-slot tail-coverage objective that induces adaptive top-P nucleus routing with $P = 1 - \tau$ (Appendix C). Finally, `Sequence-OLP` and `Batch-OLP` tighten Bal to the sequence/batch level, useful for inference-time load balancing.

Table 1: Prior routing methods in the index-annotation format.

Method	Constraint	t	e	l	b	n	RHS
Top-k + aux loss (Fedus et al., 2022; Shazeer et al., 2017)	$\mathsf{Card}_{t,l,b,n}$	$\cdot$	$\bar{\Sigma}$	$\cdot$	$\cdot$	$\cdot$	$= k_{\text{tar}}/E$
Experts choose (Zhou et al., 2022)	$\mathsf{Bal}_{e,l,b,n}$	$\bar{\Sigma}$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$= k_{\text{tar}}/E$
BASE Layers (Lewis et al., 2021)	$\mathsf{Card}_{t,l,b,n}$	$\cdot$	$\bar{\Sigma}$	$\cdot$	$\cdot$	$\cdot$	$= k_{\text{tar}}/E$
	& $\mathsf{Bal}_{e,l,b,n}$	$\bar{\Sigma}$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$= k_{\text{tar}}/E$
ALF-LB (Wang et al., 2024; Han & Zhong, 2025)	$\mathsf{Card}_{t,l,b,n}$	$\cdot$	$\bar{\Sigma}$	$\cdot$	$\cdot$	$\cdot$	$= k_{\text{tar}}/E$
	& $\mathsf{Bal}_{e,l}$	$\bar{\Sigma}$	$\cdot$	$\cdot$	$\bar{\Sigma}$	$\bar{\Sigma}$	$= k_{\text{tar}}/E$
Expert threshold (Sun et al., 2026)	*no constraint rows ($\tau = \theta$ fixed by hand; neither k_t nor k_{tar} controlled)*						

Table 2: Our trained and inference-time routing methods as conjunctions (&) of Card, Bud, and Bal. The top two rows repeat the fixed-k baselines from Table 1—`TopK+Aux` (Fedus et al., 2022) and `TopK-OLP` (the linear dual-ascent form of ALF-LB; Wang et al. 2024; Han & Zhong 2025)—which subsequent rows progressively relax. `VarK` and `TopP` share identical constraint rows and differ only in their primal objective (Section 4 vs. Appendix C). In the global cross-layer arms (`VarK-Global`, `TopP-Global`), $k_l/E := X[\bar{\Sigma}t, \bar{\Sigma}e, l, \bar{\Sigma}b, \bar{\Sigma}n]$ is layer l's realized average, allowing compute k_l to move across layers while balancing experts within each layer.

Method	Constraint	t	e	l	b	n	RHS
`TopK+Aux`	$\mathsf{Card}_{t,l,b,n}$	$\cdot$	$\bar{\Sigma}$	$\cdot$	$\cdot$	$\cdot$	$= k_{\text{tar}}/E$
	& *soft auxiliary loss penalty*						
`TopK-OLP` (ALF-LB)	$\mathsf{Card}_{t,l,b,n}$	$\cdot$	$\bar{\Sigma}$	$\cdot$	$\cdot$	$\cdot$	$= k_{\text{tar}}/E$
	& $\mathsf{Bal}_{e,l}$	$\bar{\Sigma}$	$\cdot$	$\cdot$	$\bar{\Sigma}$	$\bar{\Sigma}$	$= k_{\text{tar}}/E$
`VarK-Layer`	$\mathsf{Card}_{t,l,b,n}$	$\cdot$	$\bar{\Sigma}$	$\cdot$	$\cdot$	$\cdot$	$\in [k_{\min}/E,\ k_{\max}/E]$
`TopP-Layer`	& Bud_l	$\bar{\Sigma}$	$\bar{\Sigma}$	$\cdot$	$\bar{\Sigma}$	$\bar{\Sigma}$	$= k_{\text{tar}}/E$
	& $\mathsf{Bal}_{e,l}$	$\bar{\Sigma}$	$\cdot$	$\cdot$	$\bar{\Sigma}$	$\bar{\Sigma}$	$= k_{\text{tar}}/E$
`VarK-Global`	$\mathsf{Card}_{t,l,b,n}$	$\cdot$	$\bar{\Sigma}$	$\cdot$	$\cdot$	$\cdot$	$\in [k_{\min}/E,\ k_{\max}/E]$
`TopP-Global`	& $\mathsf{Bud}_\varnothing$	$\bar{\Sigma}$	$\bar{\Sigma}$	$\bar{\Sigma}$	$\bar{\Sigma}$	$\bar{\Sigma}$	$= k_{\text{tar}}/E$
	& $\mathsf{Bal}_{e,l}$	$\bar{\Sigma}$	$\cdot$	$\cdot$	$\bar{\Sigma}$	$\bar{\Sigma}$	$= k_l/E$
`Sequence-OLP`	$\mathsf{Bal}_{e,l,b,n}$	$\bar{\Sigma}$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$= k_l/E$
`Batch-OLP`	$\mathsf{Bal}_{e,l,n}$	$\bar{\Sigma}$	$\cdot$	$\cdot$	$\bar{\Sigma}$	$\cdot$	$= k_l/E$

4 The Routing LP, Dual Objective and the SOLP Algorithm

We start by considering the standard affinity objective and its dual on a single layer l (suppressing (l, b, n) and writing $x_{t,e} = X[t, e, l, b, n] \in [0, 1]$), and then note how the global cross-layer formulation ($\mathsf{Bud}_\varnothing$) modifies the program.

Primal LP. Given exogenous router scores $c_{t,e} \in \mathbb{R}$, the per-layer routing LP maximizes total affinity subject to Bud_l, Card, and $\mathsf{Bal}_{e,l}$ (multiplying both sides of the normalized constraints by E or TE so duals are per-assignment):

$$\max_{x \in [0,1]^{T \times E}} \frac{1}{T} \sum_{t=1}^{T} \sum_{e=1}^{E} c_{t,e}\, x_{t,e} \qquad \text{s.t.} \qquad \begin{aligned} \tfrac{1}{T}\textstyle\sum_{t,e} x_{t,e} &\le k_{\text{tar}} && (\tau \ge 0,\ \mathsf{Bud})\,, \\ k_{\min} \le \textstyle\sum_{e} x_{t,e} &\le k_{\max}\ \forall t && (\kappa_t^{\pm} \ge 0,\ \mathsf{Card})\,, \\ \tfrac{1}{T}\textstyle\sum_{t} x_{t,e} &= k_l/E\ \forall e && (\lambda_e \in \mathbb{R},\ \mathsf{Bal})\,, \end{aligned} \tag{3}$$

where we annotate each set of constraints with the corresponding dual variables, discussed next. We recall that k_l can be either a constant or based on X in the global variants.

Dual LP and threshold selection rule. Introducing the dual variables $(\tau, \lambda, \kappa^{\pm})$ and maximizing the Lagrangian pointwise over $x \in [0,1]^{T\times E}$ yields the exact dual LP:

$$\min_{\tau\geq 0,\, \lambda\in\mathbb{R}^E,\, \kappa^{\pm}\geq 0} \tau k_{\text{tar}} + \frac{k_l}{E}\sum_{e=1}^{E}\lambda_e + \frac{1}{T}\sum_{t=1}^{T}\Big(\kappa_t^{+}k_{\max} - \kappa_t^{-}k_{\min} + \sum_{e=1}^{E}\left[c_{t,e} - \lambda_e - \tau - \kappa_t^{+} + \kappa_t^{-}\right]_{+}\Big), \tag{4}$$

where $[x]_+ = \max(x, 0)$. *Remark (Per-layer* Bud_l *vs. global* $\mathsf{Bud}_{\varnothing}$*):* In the per-layer program (Bud_l), $k_l = k_{\text{tar}}$ is a fixed constant in (3)–(4). In the global cross-layer program ($\mathsf{Bud}_{\varnothing}$), the objective and budget constraint average over all $l \in [L]$ with a single shared scalar τ, while $\mathsf{Bal}_{e,l}$ substitutes the linear expression $k_l/E = \frac{1}{TE}\sum_{t,e'} x_{t,e',l}$ into (3); in the Lagrangian, this simply centers each layer's multipliers ($\tilde{\lambda}_{e,l} := \lambda_{e,l} - \frac{1}{E}\sum_{e'}\lambda_{e',l}$, so $\sum_e \tilde{\lambda}_{e,l} = 0$) and drops the $\frac{k_l}{E}\sum_e \lambda_e$ term from (4) while leaving the primal selection rule below unchanged.

The corresponding primal solution is obtained by taking the top few maximizers of the Lagrangian adjusted score, defined as

$$s_{t,e} = c_{t,e} - \lambda_e - \tau\,.$$

The number of experts chosen for a token is simply obtained by minimizing (4) over the local box multipliers $(\kappa_t^{+}, \kappa_t^{-})$ in closed form, so that for each token t, we select the top k_t experts ranked by $s_{t,e}$, where

$$k_t = \text{clip}\big(\big|\{e : s_{t,e} > 0\}\big|,\ k_{\min},\ k_{\max}\big)\,, \tag{5}$$

and $\text{clip}(x, a, b) = \min(\max(a, x), b)$. Thus, the routing assignment given a set of dual variables bears a remarkable similarity to top-k routing, but with two critical differences. First, the router scores are dynamically adjusted by the dual variables and allow global tradeoffs across tokens, layers etc. Second, the flexibility in number of chosen experts per token allows adaptive computation per token, which is critical for balance constraints without an equality on the cardinality constraint, as highlighted by Expert Choice (Zhou et al., 2022).

A Top-P routing variant: In Appendix C, we show another version of the LP with a different primal objective (same constraints) that leads to a top-P style routing, allocating an adaptive expert confidence budget to each token, rather than an adaptive expert count budget as in the choice of k_t above. We also show the merits of this alternative in our experiments.

In Appendix B, we prove some structural properties of the dual objective that shed further light on the role of the various dual parameters and their effect on the primal assignments.

The SOLP Algorithm to Update Dual Variables. The final ingredient to specify the SOLP algorithm is a recipe for updating the dual parameters. Definition (5) already handles the setting of $\kappa^{\pm}$. For λ and τ, we initialize them arbitrarily and then update them at each training step n using an online gradient descent (OGD) style update on the dual objective (4), resulting in

$$\lambda_{e,l} \leftarrow \lambda_{e,l} + \eta_\lambda \cdot (X[\bar{\Sigma}t, e, l, \bar{\Sigma}b, n] - k_l/E), \tag{6}$$

and a similar update for τ. We give a detailed implementation in Appendix A, which shows that the algorithm is readily implemented in terms of standard Python constructs. The appendix also gives implementations for the Sequence/Batch-OLP variants, which differ in some details as they perform dual updates as they process the tokens along a sequence, adapting the dual variables per sequence, rather than in chunks after each training batch of examples.

5 Theoretical Guarantees

We now dive into why the SOLP algorithm is guaranteed to obtain a near-optimal approximation to the core LP. The key insight of this section is that online gradient methods can efficiently track the optimal dual multipliers of an LP whose constraint families operate on orthogonal axes of the model.

The online vs. hindsight problem. We begin by explaining the core tension that our theory resolves and what the resulting bounds mean in practice. Ideally, to route a total of H tokens while respecting shared compute budgets (Bud) and expert capacity limits (Bal), we would solve the offline linear

program that observes all H score vectors in advance in Equation (3). We denote the offline LP value by OPT_H. In reality, a router must commit to each routing decision $x_i \in \{0,1\}^E$ on the fly using running dual variables (λ_i, τ_i). Furthermore, dual variables are updated lazily every K tokens across $U := H/K$ update steps: in training (Algorithm 1 in Appendix A), duals update once per batch of B sequences of length T ($H = NBT,\ K = BT,\ U = N$), whereas in inference (Algorithm 2 in Appendix A), duals update at each token step under batch routing ($H = BT,\ K = B,\ U = T$) or single-sequence streaming ($H = T,\ K = 1,\ U = T$). When routing greedily using running dual variables instead of solving the hindsight LP, two errors can accumulate over the H decisions: the normalized score shortfall $\mathrm{Reg}_H := \frac{1}{H}(\mathrm{OPT}_H - \sum_{i=1}^{H} c_i^\top x_i)$, and the normalized resource violations $\frac{1}{H}\,\mathrm{Viol}_H^{\mathsf{bud}} := \frac{1}{H}\left[\sum_{i=1}^{H} h_i^{\mathsf{bud}}(x_i)\right]_+$ and $\frac{1}{H}\left\|\mathrm{Viol}_H^{\mathsf{bal}}\right\|_2 := \frac{1}{H}\left\|\sum_{i=1}^{H} h_i^{\mathsf{bal}}(x_i)\right\|_2$. Here, $h_i^{\mathsf{bud}}(x_i) := \mathbf{1}^\top x_i - k_{\mathrm{tar}} \in \mathbb{R}$ measures per-token compute budget excess, while $h_i^{\mathsf{bal}}(x_i) := x_i - \frac{k_l}{E}\mathbf{1} \in \mathbb{R}^E$ (or $x_i - \frac{k_{\mathrm{tar}}}{E}\mathbf{1}$) measures per-expert allocation discrepancy, directly matching the constraints of the primal routing LP in (3).

Why standard OGD regret is not enough—and how our theory bridges the gap. Running projected Online Gradient Descent (OGD) on the dual variables guarantees sublinear *dual regret* against any fixed dual vector (Lemma 7 in Appendix D). However, low dual regret does not automatically guarantee small primal resource violations or small primal score shortfall. In fact, on arbitrary adversarial score sequences, Proposition 11 proves an information-theoretic impossibility: *any* online router achieving sublinear resource violation $o(H)$ must suffer linear score shortfall $\Omega(H)$. Under stationary score distributions (or under non-stationary drift where the total drift is bounded by Δ_H, which adds a mild additive penalty of at most $2\Delta_H$; Proposition 10), stochastic OLP duality guarantees that dual OGD tracks the optimal dual multipliers of the offline LP. Crucially, executing dual OGD with vanishing regret requires that the optimal dual multipliers $(\tau^\star, \lambda^\star)$ remain bounded. By exploiting the geometry of the routing polytope and the balance margin $\min\{k_{\mathrm{tar}}/E, 1 - k_{\mathrm{tar}}/E\}$, Proposition 6 in Appendix D proves universal bounds: $\|\lambda^\star\|_2 \le B_{\mathsf{bal}} := \frac{R_{\max}}{\min\{k_{\mathrm{tar}}/E,\, 1-k_{\mathrm{tar}}/E\}}$ and $\tau^\star \le B_{\mathsf{bud}} := 1 + B_{\mathsf{bal}}$ (with $\tau^\star \equiv 0$ when per-token cardinality is fixed), where $R_{\max} := \sup_{x\in\mathcal{X}} \mathbb{E}[c_i^\top x] \le k_{\max}$ denotes the maximum expected affinity per token. As long as the dual step sizes and projection sets are tuned using these problem parameters (as detailed in Appendix D), the dual trajectory tracks the optimal multipliers, yielding simultaneous finite-sample guarantees on both primal metrics:

Corollary 1 (Realized Convergence Rates for MoE Routing)**.** *Suppose the routing score vectors c_i are drawn from a stationary distribution $\mathcal{D}$. There exist dual step sizes and projection sets depending only on the problem parameters $(E, k_{\mathrm{tar}}, k_{\min}, k_{\max}, R_{\max})$ such that over $U = H/K$ update steps, the expected normalized score shortfall and resource violations satisfy*

$$\mathbb{E}[\mathrm{Reg}_H] \le \mathcal{O}\left(\frac{\mathcal{C}_{\mathrm{MoE}}}{\sqrt{U}}\right), \quad \mathbb{E}\left[\frac{\mathrm{Viol}_H^{\mathsf{bud}}}{H}\right] \le \mathcal{O}\left(\frac{\mathcal{C}_{\mathrm{MoE}}}{\sqrt{U}}\right), \quad \mathbb{E}\left[\frac{\left\|\mathrm{Viol}_H^{\mathsf{bal}}\right\|_2}{H}\right] \le \mathcal{O}\left(\frac{\sqrt{E}\,\mathcal{C}_{\mathrm{MoE}}}{\sqrt{U}}\right), \tag{7}$$

where $\mathcal{C}_{\mathrm{MoE}} := B_{\mathsf{bud}}(k_{\max} - k_{\min}) + B_{\mathsf{bal}}\sqrt{k_{\max}}$ captures the architectural dual scale. In particular, this guarantees a convergence rate of $\mathcal{O}(1/\sqrt{N})$ across N training batches for training (Algorithm 1 in Appendix A, where $U = N$), and a convergence rate of $\mathcal{O}(1/\sqrt{T})$ across sequences of length T for both batched inference (`Batch-OLP`, where $U = T$) and streaming inference (`Sequence-OLP`, where $U = T$, Algorithm 2 in Appendix A).

Discussion of the theoretical guarantees. Corollary 1 highlights two structural properties of our dual routing framework: first, *additive composition across constraints*: the regret bound decomposes additively across the budget (Bud) and load-balancing (Bal) constraint groups. Because dual updates separate budget and balance enforcement into independent subgradients, introducing expert load balancing incurs no multiplicative penalty on budget tracking, and vice versa. Second, *the price of batched updates*: updating dual variables lazily every K tokens incurs a cumulative drift penalty scaling as $\sqrt{KH} = K\sqrt{U}$. When normalized by the horizon $H = KU$, the factor of K cancels out ($K\sqrt{U}/(KU) = 1/\sqrt{U}$), preserving the $\mathcal{O}(1/\sqrt{U})$ rate in the number of dual gradient updates U. The complete Master Allocation Theorem for general update partitions (Theorem 8), the distribution drift analysis (Proposition 10), the adversarial lower bound (Proposition 11), and all technical proofs are deferred to Appendix D.

6 EXPERIMENTS

Experimental setup, metrics, and convergence. We construct a controlled seven-rung scaling ladder based on the Gemma-4 MoE architecture spanning 183M to 6.59B parameters ($L \in \{6, 12, 18, 24, 30\}$ layers; $E = 32$ routed experts plus one shared dense branch per layer, nominal target $k_{\text{tar}} = 4.0$, and guardrails $[k_{\min}, k_{\max}] = [2, 8]$; Table 4 in Appendix E). Every model is trained from random initialization on English C4 for 20,000 steps (16,384 tokens/step, 327.7M tokens/run) under a cosine learning-rate schedule across $n = 6$ paired seeds per method; realized compute stays within 0.36% of budget ($[3.986, 4.029]$ experts/token). On 327,680 held-out C4 tokens, we evaluate (i) validation cross-entropy loss in *nats per token*[2] and (ii) expert load imbalance via the **coefficient of variation** $\text{CV} = \sigma_e/\mu_e$ across the E experts ($\text{CV} = 0$ is uniform) and **peak expert buffer load** $\max_e \text{load}_e/(1/E)$. Because all scales train for 20,000 steps rather than a Chinchilla-proportional token budget (Hoffmann et al., 2022), we verify in Appendix E (Figure 3) that all seven loss curves flatten by step 20,000: over 99.5% of the post-warmup loss drop occurs by step 15,000, and the residual change over the final 5,000 steps (0.0014–0.0092 nats) is several times smaller than the routing gains below.

6.1 FINDING 1: RELAXING THE LP IMPROVES QUALITY ACROSS SCALE (FIGURE 1)

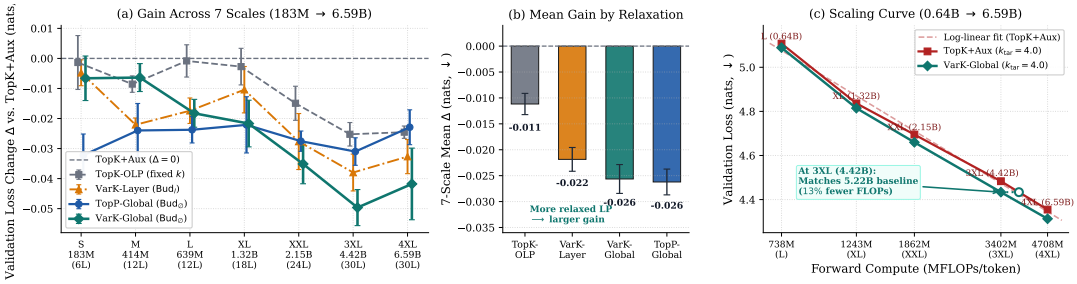

Figure 1: **Relaxing the routing LP improves held-out validation loss across 7 Gemma-4 MoE scales. (a)** Seed-paired validation loss difference Δ (± 1 SE, $n = 6$ seeds) vs. tuned `TopK+Aux` ($\Delta = 0$ at $k_{\text{tar}} = 4.0$). **(b)** Seven-scale mean validation loss reduction: relaxing per-token cardinality and pooling the compute budget across layers monotonically improves quality. **(c)** Validation loss vs. active forward compute (MFLOPs/token) from 0.64B to 6.59B; dashed red line shows the log-linear power-law fit to `TopK+Aux`.

Figure 1 evaluates the progressive constraint relaxations of Table 2 across all seven model scales (full values in Table 6, Appendix E):

- **Progressive LP relaxation drives a monotone quality improvement:** Averaged across all seven scales (Figure 1(b)), each LP relaxation yields a larger loss reduction over `TopK+Aux`: replacing the auxiliary penalty with dual expert prices under fixed top-k (`TopK-OLP`) averages $\mathbf{-0.011}$ nats; relaxing per-token cardinality to $[2/E, 8/E]$ per layer (`VarK-Layer`) doubles the gain to $\mathbf{-0.022}$ nats; and pooling layer budgets into a global budget $\text{Bud}_\varnothing$ (`VarK-Global`, `TopP-Global`) reaches $\mathbf{-0.026}$ nats.
- **Fixed-k constraint (`TopK-OLP` / ALF-LB) is neutral at small scale and improves at ≥ 2.15B:** `TopK-OLP` enforces exact cardinality ($\text{Card}_{t,l,b,n} = k_{\text{tar}}/E$) and step-level balance ($\text{Bal}_{e,l}$), directly instantiating the primal–dual ALF-LB update (Wang et al., 2024; Han & Zhong, 2025). Consistent with Proposition 2 (Appendix B)—which proves that scalar duals τ cannot alter fixed-k choices—`TopK-OLP` is neutral from 183M through 1.32B (Figure 1(a), gray curve; $|\Delta| \leq 0.0027$ nats at S, L, XL), and eliminates auxiliary gradient interference at 2.15B–6.59B to gain -0.015 to -0.025 nats ($t = -2.6$ to -12.0), consistent with Wang et al. (2024).
- **Scaling efficiency of global budget pooling:** The benefit of cross-layer pooling (`VarK-Global`) grows with depth as routed experts account for a larger share of total FLOPs (3.2% at 183M to 25.1% at 6.59B), reaching $\mathbf{-0.0497}$ nats ($t = -8.3$) at 4.42B and $\mathbf{-0.0418}$ nats at 6.59B.

[2]Since perplexity $= \exp(\text{loss})$, $\Delta = -0.05$ nats is a $\sim$4.9% perplexity reduction.

Projecting the 4.42B `VarK-Global` loss onto the `TopK+Aux` scaling curve (Figure 1(c)) matches an interpolated 5.22B baseline—saving **13**% of total forward FLOPs at zero extra cost.

6.2 Summary of Additional Empirical Findings (Figure 2)

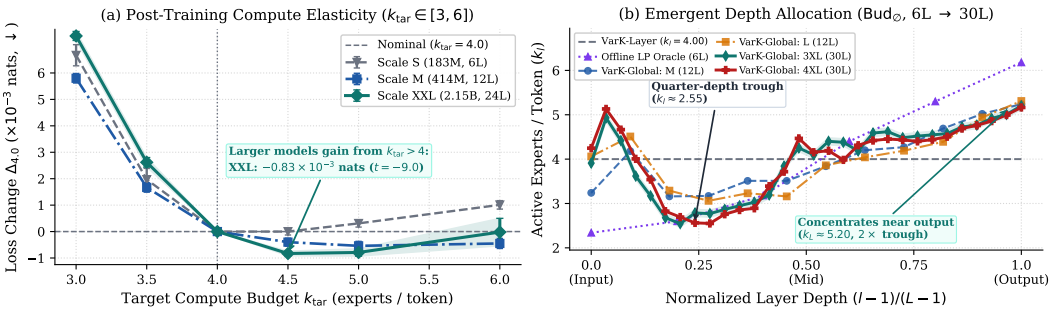

Figure 2: **Post-training compute elasticity and emergent depth allocation.** **(a)** Validation loss change $\Delta_{4.0}$ ($\times 10^{-3}$ nats, ± 1 SE, $n = 6$ seeds) when adapting a trained `TopP-Layer` checkpoint ($k_{\text{tar}} = 4.0$) to target budgets $k_{\text{tar}} \in [3.0, 6.0]$ for 1,000 continuation steps across Scales S (183M), M (414M), and XXL (2.15B). **(b)** Realized active experts/token k_l vs. normalized layer depth $(l-1)/(L-1)$ under global pooling (`VarK-Global`, Bud$_\varnothing$) across 12L and 30L models, compared to flat per-layer budgets (`VarK-Layer`, $k_l = 4.00$) and the offline LP oracle (Table 5).

For complete experimental details, ablation tables, and additional figures, we refer the reader to Appendix E; here we summarize three further empirical findings. First, Figure 2(a) (detailed in Appendix E.1) shows that because k_{tar} in Bud is a continuous dual target rather than a discrete architectural hyperparameter, a trained `TopP-Layer` checkpoint ($k_{\text{tar}} = 4.0$) can be adapted post-training to any fractional budget $k_{\text{tar}} \in [3.0, 6.0]$ over 1,000 continuation steps (5% of training), matching every target within ± 0.005 experts/token; reducing compute ($k_{\text{tar}} \in \{3.0, 3.5\}$, saving 12.5%–25% of routed FLOPs) traces a smooth quality–compute tradeoff, while increasing compute ($k_{\text{tar}} > 4.0$) yields scale-dependent gains reaching $\mathbf{-0.83 \times 10^{-3}}$ nats ($t = -9.0$) at 2.15B (XXL). Second, Figure 2(b) (Appendix E.2) reveals why global budget pooling (Bud$_\varnothing$) excels in deeper networks: when all L layers compete under a single shared price τ (`VarK-Global`), the router spontaneously shifts capacity away from early-to-middle layers (dipping to a quarter-depth trough of $k_l \approx 2.55$ experts/token) and concentrates over $2\times$ as much compute near the output ($k_L \approx 5.20$ at 30 layers).

Finally, in Appendix E.3 (Figure 4) we examine within-batch hardware load balance at inference time. Because Algorithm 1 updates expert balance duals $\lambda_{e,l}$ only once per training batch ($\bar{\Sigma}b, \bar{\Sigma}n$), open-loop evaluation of Algorithm 1 checkpoints exhibits $1.8\times$ higher single-batch load imbalance ($\text{CV}_{\text{batch}} \approx 0.119$, peak buffer load $1.27\times$ uniform) than `TopK+Aux` ($\text{CV}_{\text{batch}} = 0.064$, peak load $1.14\times$). Crucially, switching to causal within-step dual updates (Algorithm 2) on the *same frozen checkpoint* at inference time eliminates this gap without retraining: `Batch-OLP` (Bal$_{e,l,n}$) drives peak expert buffer load down to $\mathbf{1.105}\times$ (tighter than `TopK+Aux`) with only a $+0.0001$-nat loss change—preserving the full -0.022-nat quality lead—while `Sequence-OLP` (Bal$_{e,l,b,n}$) restores `TopK+Aux`-level balance even for single-sequence serving ($B = 1$) at a $+0.0009$-nat cost.

7 Conclusion

SOLP treats sparse routing as a multi-dimensional linear program over the assignment tensor $X[t, e, l, b, n]$, enforcing per-token guardrails (Card$_{t,l,b,n}$), pooled compute budgets (Bud$_l$, Bud$_\varnothing$), and expert load balance (Bal$_{e,l}$, Bal$_{e,l,n}$) via additive shadow prices (λ, τ). Our analysis shows that score shortfall and resource violations decompose additively across constraint groups and decay at rate $\mathcal{O}(1/\sqrt{U})$ over U dual updates ($\mathcal{O}(1/\sqrt{N})$ across N training steps and $\mathcal{O}(1/\sqrt{T})$ across T inference tokens), while our experiments across seven scales (183M to 6.59B) demonstrate that relaxing per-token cardinality and sharing compute across layers (`VarK-Global`, `TopP-Global`) unlocks substantial quality improvements at matched FLOPs, post-training compute elasticity, and inference-time batch load balancing.

Reproducibility statement

All formal claims have proofs in the main text or appendices. Appendix E specifies the model architectures across all seven scales, the offline LP allocation lattice, and the paired validation loss and compute elasticity tables. All language-model experiments train from random initialization on the public C4 corpus; no pretrained checkpoint is used.

AI use statement

The theoretical framework, algorithmic design, and empirical investigation in this project were developed by the authors over the course of more than two years. During manuscript preparation, LLM-based assistants were used to help polish and reorganize exposition, format tables and figures, check mathematical derivations, and assist with analysis and plotting scripts. All theoretical claims, proofs, experimental implementations, and empirical findings are the work of and were verified by the authors.

References

Shipra Agrawal and Nikhil R Devanur. Fast algorithms for online stochastic convex programming. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1405–1424, 2014.

Santiago Balseiro, Haihao Lu, and Vahab Mirrokni. Dual mirror descent for online allocation problems. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 613–628, 2020. URL https://proceedings.mlr.press/v119/balseiro20a.html.

Santiago Balseiro, Haihao Lu, and Vahab Mirrokni. The best of many worlds: Dual mirror descent for online allocation problems. *Operations Research*, 71(1):101–119, 2023. URL https://arxiv.org/abs/2011.10124.

Zewen Chi, Li Dong, Shaohan Huang, Shuming Ma, Saksham Singhal, Payal Bajaj, Xia Song, and Furu Wei. On the representation collapse of sparse mixture of experts. In *Advances in Neural Information Processing Systems*, volume 35, pp. 34600–34613, 2022.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model, 2024.

DeepSeek-AI. DeepSeek-V3 technical report, 2025.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

Mostafa Elhoushi, Akshat Shrivastava, Tatiana Likhomanenko, Anurag Wang, et al. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

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

Alex Graves. Adaptive computation time for recurrent neural networks, 2016.

X. Y. Han and Yuan Zhong. A theoretical framework for auxiliary-loss-free load balancing of sparse mixture-of-experts in large-scale AI models, 2025. URL https://arxiv.org/abs/2512.03915.

Elad Hazan. *Introduction to Online Convex Optimization*. MIT Press, 2016.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030, 2022. URL https://arxiv.org/abs/2203.15556.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1904.09751.

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

Rodolphe Jenatton, Jim Huang, and Cedric Archambeau. Adaptive algorithms for online convex optimization with long-term constraints. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 402–411, 2016. URL https://proceedings.mlr.press/v48/jenatton16.html.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, et al. Mixtral of experts, 2024.

Yongle Jin, Zhengyan Wu, Zhaoxuan Xie, Ning Ding, Zhiyuan Chen, Jian Zheng, Shuchang Bi, Zihan Hu, Peng Zou, Changze Yan, et al. Dtop-p: Adaptive routing for dynamic compute allocation in mixture of experts, 2025.

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. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2006.16668.

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

Mehrdad Mahdavi, Rong Jin, and Tianbao Yang. Trading regret for efficiency: Online convex optimization with long term constraints. *Journal of Machine Learning Research*, 13(81):2503–2528, 2012. URL https://www.jmlr.org/papers/v13/mahdavi12a.html.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models, 2024. URL https://arxiv.org/abs/2404.02258.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In *Advances in Neural Information Processing Systems*, volume 34, pp. 8583–8595, 2021.

Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

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

Hanchi Sun, Yixin Liu, Yonghui Wu, and Lichao Sun. Expert threshold routing for autoregressive language modeling with dynamic computation allocation and load balancing, 2026. URL https://arxiv.org/abs/2603.11535.

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.

Hao Yu and Michael J Neely. A low complexity algorithm with $o(\sqrt{T})$ regret and $o(1)$ constraint violations for online convex optimization with long term constraints. *Journal of Machine Learning Research*, 21:1–24, 2020.

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

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, pp. 928–936, 2003. URL https://www.martin.zinkevich.org/publications/ICML03.pdf.

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.