Optimal Expert-Attention Allocation in Mixture-of-Experts: A Scalable Law for Dynamic Model Design

Junzhuo Li[†‡], **Peijie Jiang**[¶], **Changxin Tian**[¶], **Jia Liu**[¶],
Zhiqiang Zhang[¶], and **Xuming Hu**[†‡*]
[†]The Hong Kong University of Science and Technology (Guangzhou)
[‡]The Hong Kong University of Science and Technology
[¶]Ant Group
jz.li@connect.hkust-gz.edu.cn
xuminghu@hkust-gz.edu.cn

Abstract

This paper presents a novel extension of neural scaling laws to Mixture-of-Experts (MoE) models, focusing on the optimal allocation of compute between expert and attention sub-layers. As MoE architectures have emerged as an efficient method for scaling model capacity without proportionally increasing computation, determining the optimal expert-attention compute ratio becomes critical. We define the ratio r as the fraction of total FLOPs per token dedicated to the expert layers versus the attention layers, and explore how this ratio interacts with the overall compute budget and model sparsity. Through extensive experiments with GPT-style MoE Transformers, we empirically find that the optimal ratio r^* follows a power-law relationship with total compute and varies with sparsity. Our analysis leads to an explicit formula for r^*, enabling precise control over the expert-attention compute allocation. We generalize the Chinchilla scaling law by incorporating this architectural parameter, providing a new framework for tuning MoE models beyond size and data. Our findings offer practical guidelines for designing efficient MoE models, optimizing performance while respecting fixed compute budgets.

1 Introduction

Modern large language models (LLMs) are increasingly trained under strict and explicit compute budgets. In industrial settings, model developers must operate within fixed GPU resources, training windows, and inference throughput constraints, making efficient use of computation a primary design objective rather than an afterthought . Mixture-of-Experts (MoE) (Shazeer et al., 2017; Fedus et al., 2021; Lepikhin et al., 2021; Jiang et al., 2024; Dai et al., 2024; Muennighoff et al., 2024; Qwen, 2024) architectures have emerged as a practical solution in this regime, enabling substantial parameter growth while keeping per-token computation nearly constant through sparse expert activation.

However, adopting MoE architectures introduces new architectural decisions that are largely absent in dense Transformers (Vaswani, 2017). Beyond choosing the number of experts or the activation sparsity, practitioners must implicitly decide how computation is distributed *inside* the model. In particular, how much of the available compute should be allocated to attention layers versus expert (feed-forward) layers remains an open question. In practice, this allocation is often inherited from dense Transformer designs or tuned heuristically, despite the fact that expert layers can dominate the compute budget in large-scale MoE models.

From a systems perspective, this raises a fundamental yet underexplored question: *given a fixed training compute budget, what is the optimal way to allocate computation between attention and expert components in an MoE Transformer?* We characterize this trade-off using the FLOPs ratio $r = C_E/C_A$, which directly controls where the majority of per-token computation is spent. Unlike traditional hyperparameters, r governs a first-order resource allocation decision and can substantially affect model performance at scale.

Existing neural scaling laws (Kaplan et al., 2020; Henighan et al., 2020; Hoffmann et al., 2022; Chowdhery et al., 2023) provide powerful guidance for allocating compute across model size and data. The Chinchilla scaling law, for example, prescribes an optimal balance between parameters and training tokens under a fixed compute budget. Recent extensions to MoE models further incorporate sparsity and expert count into this framework (Ludziejewski et al., 2024; Abnar et al., 2025; Wang et al., 2024). However, these approaches implicitly assume a fixed internal compute allocation, leaving the expert–attention trade-off unmodeled. As a result, current scaling laws offer limited guid-

*Corresponding author

ance on how MoE architectures themselves should evolve with increasing compute.

In this work, we show that the optimal expert–attention compute allocation is not fixed, but instead follows a predictable scaling behavior. Through controlled experiments across multiple model scales and sparsity regimes, we demonstrate that the optimal FLOPs ratio r^* increases as a power law of total training compute, with scaling coefficients that depend systematically on sparsity. Building on this observation, we incorporate the expert–attention trade-off into a unified scaling law, extending the Chinchilla framework to account for internal architectural allocation. Our results provide practical, compute-aware guidelines for designing MoE models that achieve better performance under fixed resource budgets.

2 Theoretical Motivation

This section provides a minimal theoretical rationale for why the optimal compute allocation between attention and expert components in MoE models should vary with scale. Rather than proposing a detailed mechanistic model, our goal is to capture the essential trade-off underlying expert–attention compute allocation and to motivate the empirical scaling behavior observed in later sections.

2.1 Compute Allocation in MoE Transformers

In an MoE Transformer, the dominant per-token computation arises from two sources: self-attention and expert (feed-forward) layers. Let C_A denote the FLOPs consumed by attention and C_E the FLOPs consumed by expert layers. Under a fixed per-token compute budget $C = C_A + C_E$, their relative allocation is fully characterized by the FLOPs ratio $r = C_E/C_A$.

In addition to compute allocation, MoE models are defined by their activation sparsity. We characterize sparsity by the fraction of inactive experts, $S = (E - E_{\text{act}})/E$, where E is the total number of experts and E_{act} is the number of experts activated per token. Sparsity controls how expert capacity is utilized: lower sparsity implies that more experts contribute computation per token, while higher sparsity concentrates compute on fewer experts.

Unlike dense Transformers, MoE architectures expose both r and S as explicit design choices. Fixing either implicitly constrains the effectiveness of the other. As a result, optimal compute allocation cannot be determined independently of sparsity.

2.2 Diminishing Returns under Sparse Expert Activation

Allocating additional compute to either attention or expert layers yields diminishing returns. However, in MoE models, the rate of diminishing returns for expert computation depends critically on sparsity. When sparsity is low (i.e., more experts are activated), additional expert compute can be distributed across multiple specialized subnetworks, leading to higher marginal gains. In contrast, under high sparsity, expert computation is concentrated on a small set of experts, and additional compute is more likely to saturate existing representations.

This asymmetry implies that the effective elasticity of expert compute is sparsity-dependent. While attention computation primarily benefits global token interaction modeling and is largely insensitive to sparsity, expert computation derives its benefit from both capacity and diversity, the latter being directly controlled by S.

2.3 Implications for Optimal Allocation

Under a fixed compute budget, optimal performance is achieved by balancing the marginal utility of attention and expert computation. Because sparsity modulates the effectiveness of expert compute, the optimal FLOPs ratio r^* must depend on both total compute C and sparsity S. A minimal consequence of this interaction is that r^* is not constant, but instead follows a scale-dependent law of the form

$$r^*(C, S) = \alpha(S) C^{\beta(S)}, \tag{1}$$

where the sparsity-dependent coefficients reflect how expert utilization changes under different activation regimes.

This formulation yields a clear, testable prediction: models with lower sparsity should benefit more from allocating additional compute to expert layers as scale increases, while highly sparse models should favor relatively greater attention capacity. In the following sections, we empirically validate this prediction and quantify how sparsity shapes the scaling behavior of r^*.

3 Empirical Scaling Behavior of the Optimal Compute Allocation

Motivated by the sparsity-aware allocation rationale developed in Section 2, we now empirically in-

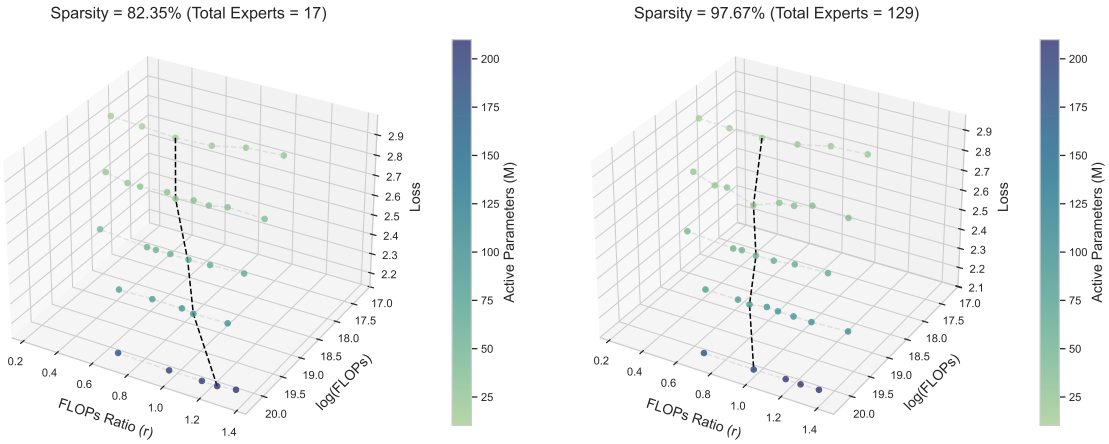


Figure 1: Loss as a function of FLOPs ratio and total compute. Black dashed lines trace optimal r^*. Color indicates active parameters. Low-sparsity models (left) favor higher r^* at scale.

vestigate how the optimal expert–attention FLOPs ratio emerges in practice. Rather than evaluating a particular MoE architecture, our focus is on characterizing the behavior of the optimal allocation itself as a function of total training compute and sparsity. Specifically, we aim to answer three questions: (i) whether a well-defined optimal ratio r^* exists under fixed compute and sparsity, (ii) how this ratio evolves as the compute budget increases, and (iii) whether sparsity systematically modulates this evolution.

To this end, we perform controlled sweeps over the FLOPs ratio r while holding the per-token compute budget fixed, across multiple model scales and sparsity regimes. This experimental design isolates the effect of internal compute allocation from confounding factors such as parameter count or data scale. We show that the optimal ratio r^* is stable, scale-dependent, and exhibits consistent trends that form the empirical basis for the scaling laws developed in the following sections.

3.1 Existence of a Scale-Dependent Optimal Ratio

We first examine whether a well-defined optimal FLOPs ratio r^* exists under fixed training compute and sparsity. Figure 1 visualizes the training loss as a function of the FLOPs ratio r and total compute C for two representative sparsity levels. Across all compute budgets, the loss surface exhibits a clear and smooth minimum along the r dimension, forming a pronounced valley rather than a flat or noisy region. This observation indicates that r^* is a stable and well-defined quantity, rather than an artifact of random variation or overfitting.

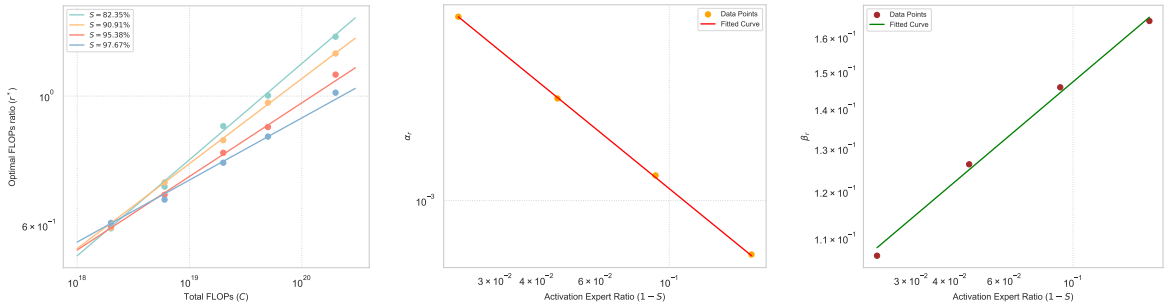
As the total compute increases, the location of this minimum shifts systematically toward larger values of r, indicating that allocating proportionally more compute to expert layers becomes increasingly beneficial at scale. Importantly, this shift is monotonic and smooth, suggesting a structured dependence on compute rather than isolated local effects. These trends are consistently observed across model sizes and training runs, confirming that the optimal ratio r^* is a robust property of the training regime.

Comparing the two sparsity settings in Figure 1 further reveals that sparsity strongly influences how rapidly the optimal ratio evolves with compute. In lower-sparsity models (left), where more experts are activated per token, the optimal r^* increases more steeply as compute grows. In contrast, under higher sparsity (right), the shift in r^* is more gradual, reflecting a reduced marginal benefit from additional expert computation. This qualitative difference provides direct empirical evidence that sparsity modulates the scale-dependent effectiveness of expert computation.

These results establish that a stable, scale-dependent optimum exists and that sparsity systematically reshapes its evolution.

3.2 Scaling of the Optimal Ratio with Compute

We next examine how the optimal FLOPs ratio r^* evolves with increasing training compute. For each



(a) Optimal ratio vs. C (b) Coefficient α_r vs. $1 - S$ (c) Exponent β_r vs. $1 - S$

Figure 2: **(a)** Relationship between the optimal FLOPs ratio r^* and total per-token compute C, fitted with the power law $r^* = \alpha_r C^{\beta_r}$. **(b)** Dependence of the fitted coefficient α_r on the fraction of activated experts $(1 - S)$, showing a clear power-law trend. **(c)** Dependence of the fitted exponent β_r on $(1 - S)$, also following a power-law. All axes are plotted on log–log scales.

fixed sparsity level, we extract r^* from the loss minima in Figure 1 and plot it against the total per-token compute C. As shown in Figure 2(a), r^* increases monotonically with compute across all sparsity regimes and follows an approximately linear trend on log–log scales.

This behavior is well captured by a power-law relationship,

$$r^* = \alpha_r C^{\beta_r},$$

which provides an accurate description of the observed data across model scales. These results establish that the optimal expert–attention allocation is itself a scale-dependent quantity rather than a fixed architectural constant.

3.3 Sparsity-Dependent Scaling Coefficients

We now analyze how sparsity modulates the scaling behavior of the optimal FLOPs ratio. Figures 2(b) and 2(c) show the dependence of the fitted coefficients α_r and β_r on the fraction of activated experts $(1 - S)$. Both parameters vary systematically with sparsity: α_r decreases while β_r increases as more experts are activated.

These trends indicate that sparsity affects not only the magnitude of the optimal ratio, but also its rate of growth with compute. Lower-sparsity models exhibit a steeper increase in r^* as scale increases, whereas under high sparsity the growth is more gradual. This confirms that sparsity reshapes the scaling regime of expert–attention allocation rather than acting as a simple constant offset.

Empirically, we find that the scaling coefficients follow simple power-law relationships with sparsity:

$$\alpha_r = 6.7 \times 10^{-5}(1 - S)^{-1.23},$$
$$\beta_r = 0.24(1 - S)^{0.21}.$$

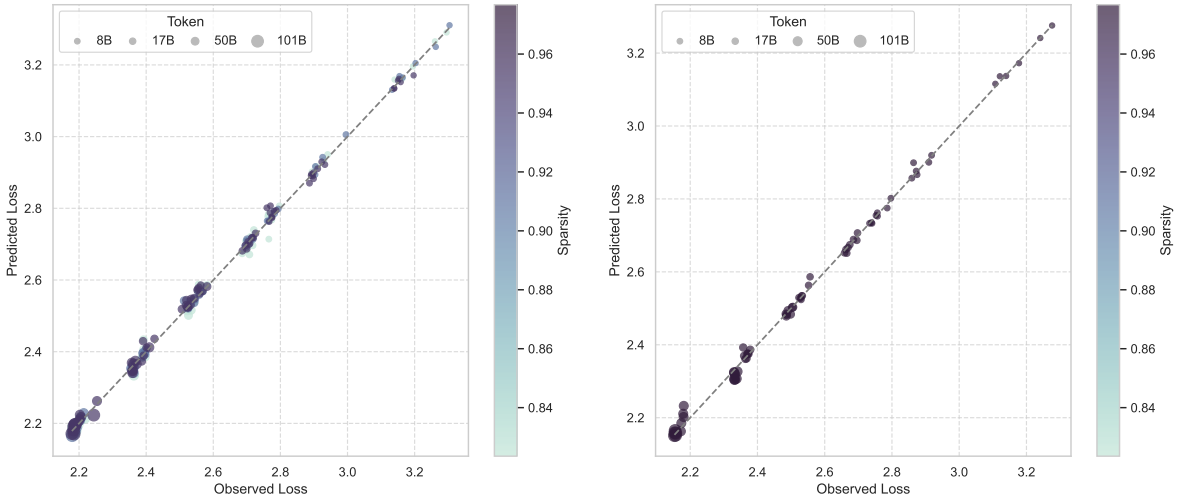
These expressions provide a closed-form parameterization of the optimal allocation across sparsity regimes.

3.4 Summary: An Empirical Law for Optimal Allocation

The results in this section establish that the optimal expert–attention FLOPs ratio is a well-defined and scale-dependent quantity. Across a wide range of model scales and sparsity regimes, the optimal ratio r^* follows a simple power-law relationship with total compute, with scaling coefficients that vary systematically with sparsity. Together, these findings yield an empirical allocation law of the form $r^*(C, S) = \alpha_r(S)\, C^{\beta_r(S)}$, which compactly captures how internal compute allocation should co-evolve with model scale and sparsity. In the next section, we incorporate this allocation law into a unified scaling framework to quantify its impact on training loss.

4 Scaling Laws with Expert–Attention Trade-offs

Section 3 establishes that the optimal expert–attention FLOPs ratio r^* follows a scale- and sparsity-dependent allocation law. We now incorporate this allocation law into a loss-level scaling framework to quantify the impact of internal compute misallocation. By explicitly modeling deviations from the optimal ratio, we extend conventional scaling laws to account for architectural allocation effects under fixed compute budgets.



(a) Fit on data used to estimate coefficients.

(b) Validating scaling law on held-out dataset.

Figure 3: Scaling law fit on data obtained from training compute-optimal models. **(a)** shows the fit on the data used to estimate the coefficients for Eq. 2, **(b)** validates these coefficients on a held-out dataset. All data points with S = 97.67% were excluded from the fitting process for out-of-sample validation.

4.1 From Allocation Law to Loss Scaling

The empirical results in Section 3 indicate that performance under a fixed compute budget depends not only on how much compute is used, but also on how it is allocated between attention and expert components. Conventional scaling laws implicitly assume a fixed internal allocation, treating models at the same compute budget as equivalent. Our results show that this assumption leads to systematic performance degradation when compute is misallocated.

To capture this effect, we extend existing scaling laws with two additional terms that penalize deviations from the optimal FLOPs ratio and excessive expert allocation. These terms do not introduce new resources; instead, they quantify the inefficiency arising from suboptimal use of a fixed compute budget.

To instantiate this formulation, we adopt the following extended scaling law:

$$\mathcal{L} = \frac{a}{N^{\alpha}} + \frac{b}{D^{\beta}} + c \cdot \frac{e^{R}(1-S)^{\gamma}}{N^{\lambda}} + d \cdot \frac{r}{r+1} + \tau, \tag{2}$$

where $a, b, c, d, \alpha, \beta, \lambda, \gamma, \tau$ are fitted parameters. In the next subsection, we evaluate the predictive accuracy of this formulation.

4.2 Empirical Validation of the Extended Scaling Law

We evaluate the predictive accuracy of the extended scaling law in Equation 2, which augments conventional loss scaling with explicit penalties for expert–attention misallocation. Figure 3(a) shows strong agreement between predicted and observed loss on the data used for coefficient estimation. More importantly, Figure 3(b) validates the same coefficients on a held-out sparsity level that was entirely excluded from fitting, demonstrating robust out-of-sample generalization.

To assess whether the model captures training dynamics beyond final loss values, Figure 4 compares predicted and observed loss trajectories for a representative MoE model. The close alignment indicates that the extended scaling law provides a consistent description of loss evolution under fixed compute allocation. Estimated coefficients are reported in Table 1.

4.3 Practical Implications under Fixed Compute Budgets

The extended scaling analysis implies that optimal MoE design requires internal compute allocation to co-scale with both total compute and sparsity. Maintaining a fixed expert–attention ratio across model scales leads to systematic inefficiencies, either underutilizing expert capacity at large scale or overspending compute on experts under high sparsity.

Coefficient	α	β	λ	γ	τ	a	b	c	d
Estimate	0.6288	0.0453	0.4228	0.0431	13.7354	15.12	18.62	39.55	0.0499

Table 1: Estimated values for coefficients in Equation 2.

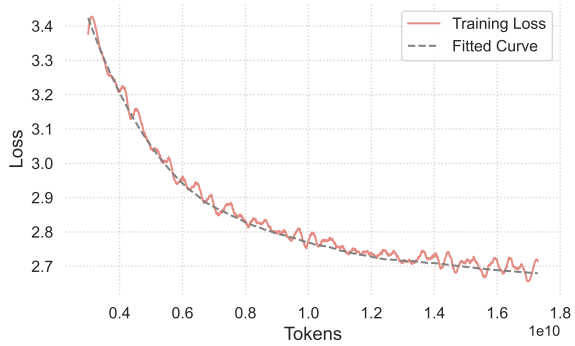


Figure 4: Comparison between the actual training curve of a model with 30M activation parameters and 550M total parameters (with 95.38% sparsity) and the fitted curve derived from Equation 2.

In practice, given a target compute budget C and sparsity level S, the empirical allocation law $r^*(C, S)$ provides a direct guideline for selecting the appropriate expert capacity. This enables architecture-level optimization under fixed resource constraints, ensuring that additional compute is translated into effective modeling capacity rather than wasted through suboptimal allocation.

5 Related Work

Neural Scaling Laws. Prior work has established that language model performance follows power-law relationships with model size, data, and compute (Kaplan et al., 2020; Hoffmann et al., 2022). These scaling laws provide principled guidance for allocating total training resources, but typically assume fixed internal architectural configurations.

Scaling Laws for MoE Models. Mixture-of-Experts (MoE) architectures (Fedus et al., 2021; Lepikhin et al., 2021; Dai et al., 2024) decouple parameter count from per-token compute through sparse expert activation. Recent studies analyze how performance scales with total parameters, expert count, granularity, and sparsity (Clark et al., 2022; Ludziejewski et al., 2024; Wang et al., 2024; Abnar et al., 2025). However, these works largely treat internal compute allocation as fixed and do not explicitly model how compute should be distributed between attention and expert components.

Our Contribution. In contrast, we treat internal compute allocation as a first-order scaling variable. Rather than optimizing only total capacity or sparsity, we model how expert–attention FLOPs allocation should co-scale with compute and sparsity. This perspective extends existing scaling frameworks by explicitly accounting for architectural compute misallocation and its impact on training loss.

6 Discussion

6.1 Optimal Compute Allocation and Scaling Laws

Our results demonstrate that the optimal division of computational resources between attention and expert submodules in MoE architectures is a function of both the total training budget C and the model's sparsity S, rather than a fixed hyperparameter. Whereas traditional scaling laws treat model size N, data volume D as orthogonal axes, we introduce *internal compute allocation*—the FLOPs ratio r^*—as a fourth, scale-dependent dimension. Concretely, we observe a clear power-law scaling

$$r^*(C, S) = \alpha(S)\, C^{\beta(S)},$$

and find that even modest deviations from r^* can lead to substantial performance degradation. For example, at $C = 2 \times 10^{20}$ FLOPs, a 20% misallocation relative to r^* incurs a loss increase of 0.15, corresponding to roughly a 15% decline in model quality. This underscores that co-scaling architectural ratios alongside compute budgets is essential for fully realizing MoE efficiency gains.

The role of sparsity S emerges as a readily tunable lever for architectural efficiency. Lower values of S (i.e., more active experts per token) shift the optimum toward expert-heavy regimes: when S decreases from 97.67% to 82.35%, r^* grows by over 40%, reflecting that highly sparse models derive greater benefit from dedicating additional FLOPs to expert layers. Conversely, sparser activation patterns favor increased attention capacity. This behavior provides clear guidance for system designers: under tight compute constraints, low–sparsity

configurations should tilt more compute toward experts, while high–sparsity settings should allocate proportionally more to attention.

By incorporating r^* into the broader Chinchilla-style scaling framework (Section 4), practitioners can now navigate the joint trade-offs among model size, data, and internal architecture. For instance, with a fixed compute budget and $S = 95.38\%$, diverting 10% of attention FLOPs to expert layers yields a loss reduction of 0.08; however, the same reallocation at $S = 82.35\%$ increases loss by 0.12. Such quantitative insights enable precise budget allocation decisions, particularly for organizations operating under limited computational resources.

6.2 Limitations and Future Work

Our study, however, has several caveats that suggest avenues for future work. First, all experiments were conducted on autoregressive language modeling tasks, and the optimal r^* may shift in multimodal, classification, or reinforcement-learning domains. Second, we assume a static sparsity S during pre-training; allowing S itself to adapt dynamically—through techniques like adaptive routing—could further enhance performance. Finally, our analysis neglects real-world communication costs—such as inter-device bandwidth and routing overhead—and future work should integrate these system-level constraints into the compute-allocation formulation.

7 Conclusion

In this paper, we establish that the compute allocation between attention and expert layers—quantified by the FLOPs ratio r—is a critical yet understudied dimension of MoE scaling. We present three key contributions:

- Dynamic Allocation Principle: We demonstrate that the optimal r^* follows a power-law relationship with total compute C, modulated by sparsity S, such that $r^* \propto C^{\beta_r(S)}$, with $\beta_r(S) \propto (1 - S)^{0.24}$.
- We integrate the architectural allocation of compute with the Chinchilla scaling framework, revealing that deviations from the optimal r^* incur an exponential penalty on model loss.
- Design Guidelines: We identify key design principles, where high-sparsity regimes ($S > 95\%$) favor attention-centric configurations ($r < 1$), while low-sparsity models ($S < 90\%$) benefit from expert-heavy configurations ($r > 1$).

These results shift the paradigm of MoE scaling from merely "adding more experts" to jointly optimizing architecture and scale. By providing both theoretical grounding and empirical validation, our work equips practitioners with tools to navigate the expanding design space of sparse models. Future research could extend this framework to dynamic sparsity, cross-modal tasks, and hardware-aware optimization.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No.62506318); Guangdong Provincial Department of Education Project (Grant No.2024KQNCX028); CAAI-Ant Group Research Fund; Scientific Research Projects for the Higher-educational Institutions (Grant No.2024312096), Education Bureau of Guangzhou Municipality; Guangzhou-HKUST(GZ) Joint Funding Program (Grant No.2025A03J3957), Education Bureau of Guangzhou Municipality.

References

Samira Abnar, Harshay Shah, Dan Busbridge, Alaaeldin Mohamed Elnouby Ali, Josh Susskind, and Vimal Thilak. 2025. Parameters vs flops: Scaling laws for optimal sparsity for mixture-of-experts language models. *arXiv preprint arXiv:2501.12370*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Aidan Clark, Diego de Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, and 1 others. 2022. Unified scaling laws for routed language models. In *International conference on machine learning*, pages 4057–4086. PMLR.

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. 2024. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, Bangkok, Thailand. Association for Computational Linguistics.

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, and 181 others. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint*.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, and 1 others. 2020. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, and 1 others. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, and 7 others. 2024. Mixtral of experts. *CoRR*, abs/2401.04088.

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

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*.

Jan Ludziejewski, Jakub Krajewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, and 1 others. 2024. Scaling laws for fine-grained mixture of experts. In *Forty-first International Conference on Machine Learning*.

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, and 5 others. 2024. Olmoe: Open mixture-of-experts language models. *Preprint*, arXiv:2409.02060.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, and 1 others. 2021. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 1–15.

Team Qwen. 2024. Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters".

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

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Siqi Wang, Zhengyu Chen, Bei Li, Keqing He, Min Zhang, and Jingang Wang. 2024. Scaling laws across model architectures: A comparative analysis of dense and MoE models in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5583–5595, Miami, Florida, USA. Association for Computational Linguistics.

A Complete Derivation

A.1 Base Framework: Optimal Allocation

We begin by deriving the intrinsic relationship between the optimal FLOPs ratio r^* and the total compute C, assuming fixed architectural parameters. The model's loss $\mathcal{L}$ is governed by two competing factors:

Attention Capacity ($\mathcal{A}$): This represents the ability to model token interactions and scales sublinearly with attention compute C_A. The reason for this sublinear scaling is that as the model's attention mechanism captures more long-range dependencies in the input sequence, the additional improvement from each extra unit of compute becomes progressively smaller. Beyond a certain point, the model reaches a level where further compute contributes less to the overall understanding of token interactions. This diminishing return is commonly observed in large-scale attention models, which can be captured by the relation $\mathcal{A}(C_A) = C_A^{\mu_A}$, where $\mu_A \in (0, 1)$.

Expert Capacity ($\mathcal{E}$): This refers to the model's ability to process specialized features and scales sublinearly with expert compute C_E. As more compute is allocated to experts, the model's performance improves at first, but the gains decrease

due to the overhead of managing additional experts and the limited interactions between them. Furthermore, in MoE models, only a subset of experts are activated for each token, so the additional compute does not always result in proportionally more active experts, leading to diminishing returns. This behavior is reflected in the scaling law $\mathcal{E}(C_E) = C_E^{\mu_E}$, where $\mu_E \in (0, 1)$.

The total loss $\mathcal{L}$ is modeled as a weighted sum of these two components:

$$\mathcal{L} = \alpha_A \cdot \mathcal{A}(C_A)^{-\gamma_A} + \alpha_E \cdot \mathcal{E}(C_E)^{-\gamma_E}, \quad (3)$$

where γ_A and γ_E represent diminishing returns for attention and expert capacities, respectively, and α_A and α_E are architecture-dependent coefficients.

A.2 Optimality Condition

The goal is to find the optimal compute allocation between attention and expert capacity, defined by the ratio $r^* = C_E/C_A$, which minimizes the model's loss. The optimal ratio is found by setting the derivatives of the loss with respect to C_A and C_E equal to each other:

$$\frac{\partial \mathcal{L}}{\partial C_A} = \frac{\partial \mathcal{L}}{\partial C_E} \implies \frac{\alpha_A \gamma_A \mu_A}{C_A^{\gamma_A \mu_A + 1}} = \frac{\alpha_E \gamma_E \mu_E}{C_E^{\gamma_E \mu_E + 1}}.$$

To proceed, we express C_A and C_E in terms of the total compute C and the FLOPs ratio r. Assuming the following relationships:

$$C_A = \frac{C}{1+r}, \quad C_E = \frac{Cr}{1+r},$$

we substitute these into the optimality condition and solve for r^*. This gives the optimal FLOPs ratio r^* as:

$$r^* = \alpha_r \cdot C^{\beta_r}, \quad (4)$$

where α_r and β_r are defined as:

$$\alpha_r = \left(\frac{\alpha_E \gamma_E \mu_E}{\alpha_A \gamma_A \mu_A} \right)^{\frac{1}{\gamma_A \mu_A + \gamma_E \mu_E + 1}},$$

$$\beta_r = \frac{\gamma_E \mu_E - \gamma_A \mu_A}{\gamma_A \mu_A + \gamma_E \mu_E + 1}.$$

This relationship establishes a power-law scaling between the optimal ratio r^* and total compute C, with the constants α and β determined by the elasticity parameters $\mu_A, \mu_E, \gamma_A, \gamma_E$.

B Experimeal Setup

In this section, we detail the core components and procedures underpinning our empirical evaluation. We begin by describing the model architecture, including the design of sparse MoE layers and the configuration used to control per-token compute. Next, we outline the composition and sourcing of our training data, highlighting its multilingual and multimodal nature. Finally, we present our strategy for selecting the optimal FLOPs ratio r^*, including criteria for handling unexpected fluctuations in the loss landscape. Together, these subsections provide a comprehensive overview of how we ensure fairness, reproducibility, and practical relevance in our experiments.

B.1 Model Architecture and Training Configuration

We implement a GPT-style decoder-only Transformer model, with sparse Mixture-of-Experts (MoE) layers replacing the conventional feed-forward networks (FFNs). Each MoE layer is designed to activate a subset of experts, specifically **3 experts per token**, consisting of 1 shared expert and 2 routed experts. This design yields a sparsity factor $S = \frac{E-3}{E}$, where E represents the total number of experts per layer. We evaluate models with varying expert counts, specifically $E \in \{17, 33, 65, 129\}$, which correspond to sparsity levels of $S \in \{82.35\%, 90.91\%, 95.38\%, 97.67\%\}$, covering a broad spectrum of sparsity commonly found in current MoE models. This range ensures that our findings are broadly applicable and relevant to contemporary MoE model designs.

To ensure fairness and consistency in our experiments, we maintain a constant total per-token compute budget across all models. The per-token compute cost, C_{token}, is defined as:

$$C_{\text{token}} = \frac{C_A + C_E}{n_{\text{token}}},$$

where C_A and C_E represent the compute costs for the attention and expert layers, respectively, and n_{token} is the number of tokens processed. We vary the compute allocation ratio, $r = \frac{C_E}{C_A}$, in the range of $r \in [0.2, 1.5]$, adjusting the dimensions of the attention and expert layers while keeping the total compute budget fixed.

We conduct experiments across six benchmark models of different sizes—spanning a total parameter count N from 100 M to 5 B, with the largest

training FLOPs reaching 1×10^{21}—for each sparsity level. For each model, we fix the per-token compute budget C_{token} and vary the FLOPs ratio r by independently scaling the dimensions of the attention and expert sublayers. All variants share the same depth (number of layers) and number of attention heads. Each model is assigned the activation-parameter label N_a based on its corresponding benchmark, ensuring that models with identical C_{token} but different r share the same label. Prior to experiments, we performed a hyperparameter search over learning rates and batch sizes for each label, finding that all models within a label shared a single optimal combination of learning rate and batch size. These hyperparameter settings are summarized in Table 2. For consistency across all experiments, we fix the context length to $n_{\text{ctx}} = 4096$ tokens and the vocabulary size to $n_{\text{vocab}} = 128\text{k}$ tokens. All experiments were conducted on an A100 GPU cluster.

B.2 Datasets

The training dataset is carefully curated to provide a comprehensive and balanced representation of diverse domains, ensuring the model's generalization across various languages and content types. The dataset is composed primarily of Chinese (15%), English (60%), and code (25%), reflecting the model's focus on multilingual and multimodal capabilities. The content is sourced from a wide variety of domains, with the largest proportion coming from web-based data (50%), followed by code (25%), books (5%), academic sources (5%), and math-related content (5%). Other domains, including news, social media, and exam-related content, contribute smaller yet significant portions.

This diverse selection of training data not only enables the model to learn from a wide range of linguistic and contextual patterns but also facilitates its ability to generate high-quality content across different modalities. By incorporating a mixture of textual and code-based data, the model is equipped to handle a variety of real-world tasks, from natural language understanding and generation to code completion and debugging. The inclusion of data from multiple languages further enhances its robustness, allowing for effective processing and generation in both English and non-English contexts.

B.3 Selection Strategy for the Optimal Point r^*

In our experimental setup, for a fixed set of parameters—specifically, the total FLOPs C and sparsity S—the optimal value of r is typically chosen based on the lowest loss observed during the experiments. This value of r is selected as the optimal point r^*.

However, in cases where the actual optimal point does not align with expectations, we consider the possibility of selecting a suboptimal point. This occurs when, despite an increase in the total FLOPs, the value of the optimal ratio r^* decreases. This observation contradicts the common industry expectation that, as training volume increases, the proportion of attention should decrease. In such instances, we interpret this behavior as a fluctuation, rather than a clear deviation. Consequently, we select the suboptimal point as the theoretical optimal point, provided that the loss difference between the suboptimal point and the actual optimal point is less than 0.001.

C Estimating MoE FLOPs

In this section, we describe the computation of floating-point operations (FLOPs) for training a Mixture of Experts (MoE) model, following Narayanan et al. (2021). Specifically, we consider the forward and backward passes, with a focus on matrix multiplications that dominate the computational cost. This section breaks down the FLOPs into several key components: **attention mechanisms**, **sparse MLP layers**, and **output layers**. The FLOPs for the backward pass are assumed to be approximately twice that of the forward pass unless otherwise specified.

C.1 Attention Mechanism FLOPs

The attention mechanism in a Transformer model consists of several stages, including query, key, and value projections, followed by the computation of the attention weights and their application to the values.

Query Projection The query projection is computed as a matrix multiplication between the input sequence and the query projection matrix. The FLOPs for the query projection are given by:

$$C^{\text{Q}}_{\text{proj}} = 2 n_{\text{ctx}} d^2_{\text{hidden}}.$$

Key-Value Projection For MoE models, the number of key and value heads is typically dif-

Table 2: Key Hyperparameters

Label (Active Params)	Layers (n_{layer})	Heads (n_{head})	Batch Size	Learning Rate
20M	8	8	96	0.0015
30M	8	8	160	0.0013
55M	10	10	224	0.0011
100M	14	12	320	0.0009
200M	16	16	512	0.0008

ferent from the number of attention heads. If a generic query architecture (GQA) is used, the key and value projections require separate computations for the keys and values. The corresponding FLOPs are given by:

$$C_{\text{proj}}^{\text{KV}} = 4n_{\text{ctx}}d_{\text{hidden}}^2 \div \text{kv head ratio}.$$

Otherwise, without GQA, the key-value projection FLOPs are twice that of the query projection:

$$C_{\text{proj}}^{\text{KV}} = 2C_{\text{proj}}^{\text{Q}}.$$

Attention Weights and Value Application The computation of the attention weights involves matrix multiplications between the queries and keys, and their application to the values:

$$C_{\text{weight}}^{\text{Attn}} = 2n_{\text{ctx}}^2 d_{\text{hidden}}.$$

$$C^{\text{Value}} = 2n_{\text{ctx}}d_{\text{hidden}}^2.$$

Final Output Projection After applying the attention weights, a final output projection is computed:

$$C_{\text{proj}}^{\text{Output}} = 2n_{\text{ctx}}d_{\text{hidden}}^2.$$

The total FLOPs for the attention mechanism (including both projections and attention computations) are then given by:

$$\text{Attn Forward FLOPs} = \text{Attn Projection FLOPs} + \text{Attn Mechanism FLOPs}.$$

C.2 Sparse MLP Forward FLOPs

In MoE models, the sparse expert mechanism introduces additional computational overhead due to the dynamic selection of experts for each token. The sparse MLP forward pass involves a combination of linear transformations and matrix multiplications across the selected experts.

The FLOPs for the sparse MLP forward pass are calculated as follows:

$$C_{\text{expert}} = n_{\text{ctx}} \cdot \Big(2d_{\text{hidden}}E + 3 \cdot 2 \cdot d_{\text{hidden}}d_{\text{expert}} \cdot \big(\text{top-}k + n_{\text{shared_experts}}\big)\Big).$$

C.3 Output Layer FLOPs

Finally, the output logits layer computes the probability distribution over the vocabulary for each token. The FLOPs for the output layer are given by:

$$C_{\text{logits}} = 2n_{\text{ctx}}d_{\text{hidden}}n_{\text{vocab}}.$$

C.4 Decoder Layer FLOPs

Each transformer decoder layer combines attention and sparse MLP computations. The total FLOPs for a single decoder layer forward pass are therefore the sum of the attention and sparse MLP FLOPs:

$$C_{\text{forward}}^{\text{layer}} = \text{Attention Forward FLOPs} + \text{Sparse MLP Forward FLOPs}.$$

Total Forward Pass FLOPs The total forward pass FLOPs for the entire model, excluding gradient checkpointing, is given by:

$$C_{\text{forward}} = n_{\text{layer}}C_{\text{forward}}^{\text{layer}} + C_{\text{logits}}.$$

Backward Pass FLOPs The backward pass typically incurs twice the computational cost of the forward pass, as it requires the computation of gradients for all parameters. If PEFT (Parameter-Efficient Fine-Tuning) is used, we assume that the backward pass will require a factor of 2. Otherwise, we use a factor of 3:

$$C_{\text{backward}} = C_{\text{forward}} \cdot \text{factor}.$$

Gradient Checkpointing If gradient checkpointing is enabled, the forward pass computations are adjusted to store intermediate results during the forward pass to reduce memory usage, but at the cost of additional computational overhead. The total FLOPs in this case are given by:

$$C_{\text{forward}} = n_{\text{layer}} C_{\text{forward}}^{\text{layer}} \cdot (\text{factor}+1) + C_{\text{logits}} \cdot \text{factor}.$$

Given these calculations, the total hardware FLOPs and model FLOPs can be computed for both the forward and backward passes. The specific computational cost depends on factors such as the use of gradient checkpointing, the PEFT setting, and the number of active experts in the MoE model.

D Detailed Analysis and Visualizations of the Scaling Law Fitting Process

D.1 Initialization Parameters for L-BFGS Optimization

Table 3 shows the parameters used to initialize L-BFGS, which was employed to fit the proposed parametric scaling law presented in Equation 2.

D.2 Justification for the Efficiency Loss Term

To justify the use of the $\frac{r}{r+1}$ form for the efficiency loss term, we analyze its mathematical properties in relation to diminishing returns. This function exhibits three key characteristics that align with empirical observations of expert compute allocation:

- **Bounded Growth**: As $r \to \infty$, $\frac{r}{r+1} \to 1$, preventing unbounded loss growth while maintaining sensitivity to allocation changes in practical r ranges (typically $r \in [0.2, 1.5]$ in our experiments). This reflects the physical reality that over-allocating compute to experts cannot infinitely degrade performance.

- **Sublinear Scaling**: The derivative $\frac{d}{dr}\left(\frac{r}{r+1}\right) = \frac{1}{(r+1)^2}$ decreases quadratically with r, explicitly modeling diminishing marginal returns from increasing expert allocation. This matches our empirical finding that performance gains from expert compute saturate faster than those from attention compute.

- **Zero-Cost Baseline**: At $r = 0$ (no expert allocation), the term vanishes ($\frac{0}{0+1} = 0$), ensuring no artificial penalty when experts are disabled. The linear regime $\frac{r}{r+1} \approx r$ when $r \ll 1$ matches observed near-linear improvements at small r values.

Alternative formulations like $\log(1 + r)$ or $1 - e^{-r}$ were tested but showed poorer empirical fit, particularly in matching the saturation behavior observed at high r. The chosen form provides the best trade-off between empirical accuracy and parameter efficiency, requiring only a single scaling coefficient d to capture the full efficiency loss trajectory.

D.3 Visualizations of Alternative Formula Fits

In this subsection, we compare the observed validation losses with the predictions made by two previously proposed scaling formulas, using scatter plots to assess their fitting quality.

Wang et al. (2024) extend the traditional dense-scaling law by introducing the total number of experts E as an additional factor:

$$\mathcal{L} = \frac{a}{N^{\alpha} E^{\gamma}} + \frac{b}{D^{\beta}} + \tau. \tag{5}$$

However, this formulation ignores the fraction of activated experts and is constrained to $E < 100$, which limits its applicability to modern high-sparsity, fine-grained MoE models (DeepSeek-AI et al., 2024). Figure 5(a) shows that, across all model scales, Equation 5 fails to capture the true loss trends.

Abnar et al. (2025) incorporate sparsity S into their fit:

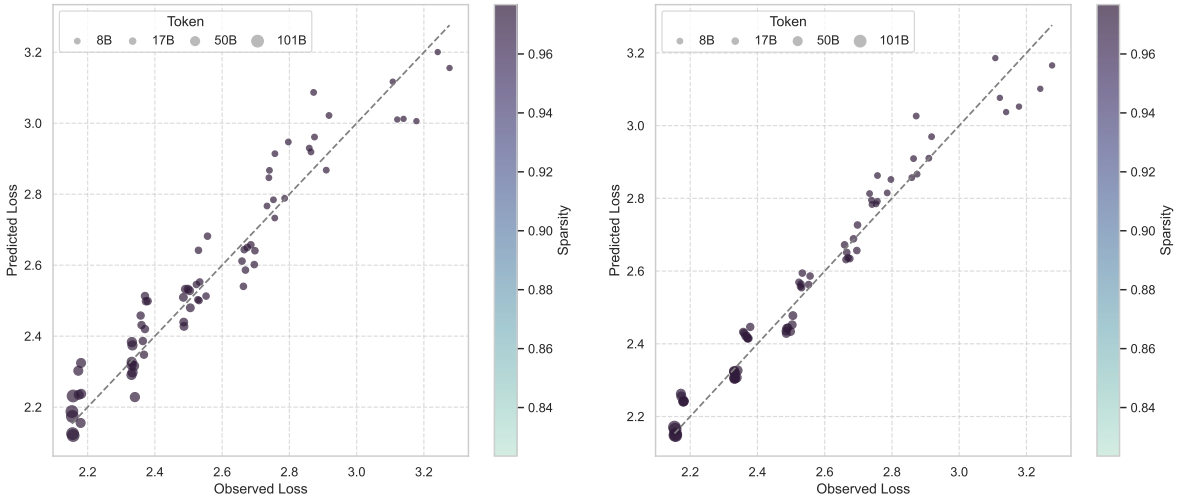
$$\mathcal{L} = \frac{a}{N^{\alpha}} + \frac{b}{N^{\beta}} + \frac{c}{(1-S)^{\gamma}} + \frac{d}{(1-S)^{\delta} N^{\gamma}} + \tau. \tag{6}$$

As plotted in Figure 5(b), this formula performs reasonably well on large models but still exhibits substantial errors for smaller ones.

E Additional Visualizations

Coefficients	Initial Values
$\log(a), \log(b), \log(c), \log(d)$	[0, 10, 20]
$\alpha, \beta, \lambda, \gamma$	[0, 0.25, 0.5, 0.75, 1, 1.25]
$\log(\tau)$	1.5

Table 3: Initial values used to estimate coefficients in Equation 2.



(a) Wang et al. (2024)

(b) Abnar et al. (2025)

Figure 5: Comparison of observed versus predicted validation losses for two alternative scaling formulas. (a) Predictions using the Wang et al. (2024) formulation (Equation 5), which fails to generalize across modern high-sparsity settings. (b) Predictions using the Wang et al. (2024) formulation, which fits large models well but underperforms on smaller ones. The solid diagonal line indicates perfect agreement.

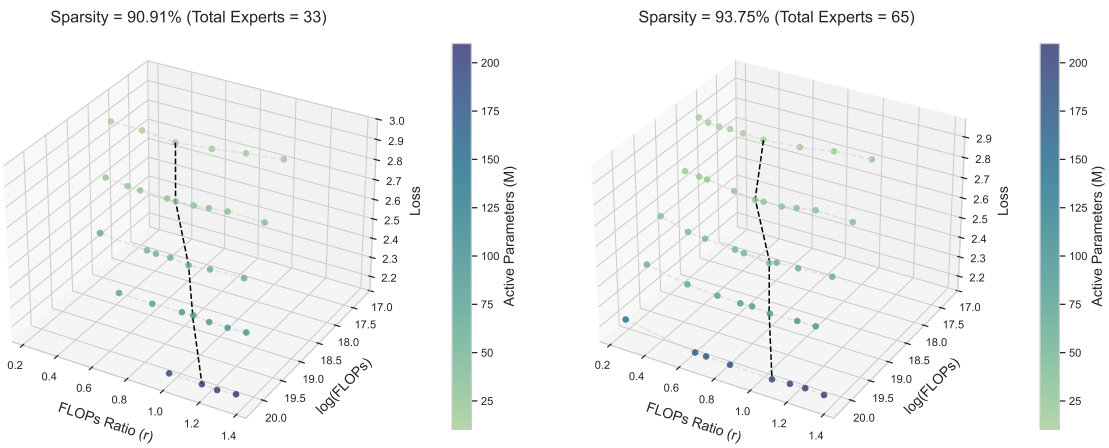


Figure 6: Loss as a function of FLOPs ratio and total compute. Black dashed lines trace optimal r^*. Color indicates active parameters. Low-sparsity models (left) favor higher r^* at scale.