Towards Greater Leverage: Scaling Laws for Efficient Mixture-of-Experts Language Models

Changxin Tian, Kunlong Chen, Jia Liu, Ziqi Liu, Zhiqiang Zhang*, Jun Zhou

Ling Team, Ant Group

*Corresponding author

Mixture-of-Experts (MoE) has become a dominant architecture for scaling Large Language Models (LLMs) efficiently by decoupling total parameters from computational cost. However, this decoupling creates a critical challenge: predicting the model capacity of a given MoE configurations (*e.g.*, expert activation ratio and granularity) remains an unresolved problem. To address this gap, we introduce *Efficiency Leverage (EL)*, a metric quantifying the computational advantage of an MoE model over a dense equivalent. We conduct a large-scale empirical study, training over 300 models up to 28B parameters, to systematically investigate the relationship between MoE architectural configurations and EL. Our findings reveal that EL is primarily driven by the expert activation ratio and the total compute budget, both following predictable power laws, while expert granularity acts as a non-linear modulator with a clear optimal range. We integrate these discoveries into a unified scaling law that accurately predicts the EL of an MoE architecture based on its configuration. To validate our derived scaling laws, we designed and trained `Ling-mini-beta`, a pilot model for Ling-2.0 series with only 0.85B active parameters, alongside a 6.1B dense model for comparison. When trained on an identical 1T high-quality token dataset, `Ling-mini-beta` matched the performance of the 6.1B dense model while consuming over 7x fewer computational resources, thereby confirming the accuracy of our scaling laws. This work provides a principled and empirically-grounded foundation for the scaling of efficient MoE models.

Date: July 20, 2025

Correspondence: tianchangxin.tcx@antgroup.com, lingyao.zzq@antgroup.com

1 Introduction

Recently, Mixture-of-Experts (MoE) models (Shazeer et al., 2017; Jiang et al., 2024; DeepSeek-AI, 2024) have emerged as a leading paradigm for constructing large language models (LLMs) (Zhao et al., 2023), primarily due to its remarkable computational efficiency (Clark et al., 2022). By leveraging sparse activation, MoE models can dramatically increase their total parameter count without proportionally increasing the computational cost (FLOPs). For instance, DeepSeekMoE (Deepseek-AI et al., 2024), with 16 billion total parameters, activates only 2.8 billion per token, yet achieves performance comparable to a 7-billion-parameter dense model, showcasing a parameter efficiency gain of approximately 2.5x. However, the decoupling of computational cost from the total parameter count in MoE architectures introduces a new challenge in assessing a model's capacity. While capacity in dense models is traditionally correlated with the parameter count, for MoE models, neither the total nor the activated parameter count alone serves as a reliable proxy for performance. Consequently, predicting the effective capacity of a specific MoE architecture and setting realistic

performance expectations before pre-training remains a critical and unresolved problem.

To understand the relationship between MoE architecture and performance, recent studies (Clark et al., 2022; Ludziejewski et al., 2024; Abnar et al., 2025; Ludziejewski et al., 2025) have begun to investigate their scaling laws. As a cornerstone to LLM research, scaling laws (Kaplan et al., 2020; Hestness et al., 2017) reveal a core principle: model performance improves predictably with increases in compute, model size, and data scale. This fundamental property allows us to infer the performance of vastly larger models by training a series of smaller ones, offering crucial insights into architectural scalability. However, existing research on MoE scaling laws has predominantly focused on isolated dimensions, such as parameter sparsity (Clark et al., 2022; Abnar et al., 2025) or expert granularity (Ludziejewski et al., 2024). Given that MoE performance is governed by the complex interplay of multiple interdependent factors, we are still unable to intuitively determine the "equivalent capacity" of a given MoE architecture.

To address this challenge, we introduce **Efficiency Leverage (EL)**, a metric designed to quantify the computational efficiency of a MoE architecture relative to a dense counterpart. Specifically, we define the EL of a target MoE architecture $\mathcal{X}_{\text{MoE}}$ with respect to a dense baseline $\mathcal{X}_{\text{Dense}}$ as the ratio of their computational costs C required to achieve the same performance level (*e.g.*, identical loss):

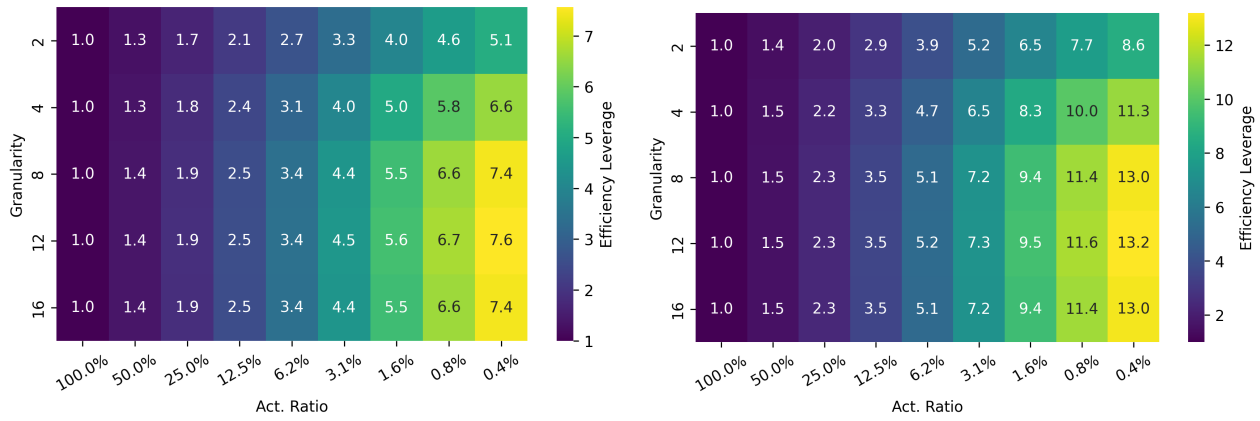
$$EL(\mathcal{X}_{\text{MoE}} \mid \mathcal{X}_{\text{Dense}}) = \frac{C_{\text{dense}}}{C_{\text{moe}}}, \tag{1}$$

This definition provides a standardized benchmark for architectural comparison: a higher EL value signifies greater efficiency. For instance, an EL of 2 indicates the MoE model requires only half the computational cost to reach the same performance as the dense baseline. Consequently, for a fixed compute budget, an MoE architecture with higher EL enables larger effective parameter scaling or more comprehensive training, thereby improving efficiency.

To investigate EL and its relationship with MoE architectures, our study employs a three-stage methodology. First, we establish scaling laws for optimal hyper-parameters and model-data allocation in dense/MoE models, ensuring all experimental models are evaluated in their "well-trained" conditions. Second, we deconstruct the MoE architecture into its core design dimensions, including the expert activation ratio, expert granularity, the shared expert ratio, and others configurations. We then systematically analyze the impact of each dimension on EL. Finally, we integrate our empirical findings into a unified scaling law to models the relationship between MoE configurations and the resulting EL, providing a predictive framework for designing efficient MoE models.

Our large-scale empirical study, encompassing over 300 trained models with up to 28B parameters, reveals several core principles governing the efficiency of MoE architectures. We distill our findings into the following key insights:

1. **Activation ratio as the primary driver of efficiency.** The expert activation ratio emerges as the primary determinant of EL. We observe a stable power-law relationship: EL increases as the activation ratio decreases (*i.e.*, as sparsity increases). This reveals that sparsely activated pathways yield consistent and predictable gains in computational efficiency.
2. **Expert granularity as a non-linear modulator.** Superimposed on this primary trend, expert granularity introduces a log-polynomial adjustment to EL. This effect is independent of the total compute budget and implies an optimal range for expert size. Our experiments, which utilize a standard load-balancing loss, identify this optimum to be between 8 and 12.
3. **Amplifying effect of the compute budget.** Crucially, the EL of a given MoE architecture is not static; it scales with the training compute budget, also following a power law. This finding



(a) FLOPs = $1e20$ **(b)** FLOPs = $1e22$

Figure 1 Estimated MoE efficiency leverage using Eq. (13) for FLOPs budgets of $1e20$ and $1e22$.

underscores the advantage of MoE models in large-scale pre-training scenario, where their efficiency gains become increasingly significant as computational resources expand.

4. **Secondary impact of other architectural factors.** Other design choices, such as the use of shared experts or the specific arrangement of MoE and dense layers, exert only a secondary influence on EL. These factors typically have broadly applicable, near-optimal settings that require minimal tuning.

Building upon these observations, we derive a unified scaling law for efficiency leverage of MoE. This formula integrates the combined effects of the compute budget, activation ratio, and expert granularity. It enables us to directly predict an MoE architecture's EL for a given configuration, providing principled guidance for efficient MoE architectural design. As a practical demonstration, Figure 1 illustrates the estimated EL under compute budgets of $1e20$ and $1e22$ FLOPs.

According to our derived scaling law for EL, we predict that an MoE model with a 3.1% activation ratio and an expert granularity of 12 will achieve over 7x computational efficiency under a $1e22$ FLOPs compute budget. To empirically validate this, we designed and trained "Ling-mini-beta," a pilot model for the Ling-2.0 series, with 0.85B activated parameters and 17.5B total parameters. The model was trained on a 1T-token high-quality dataset and benchmarked against its dense counterpart, a 6.1B parameter model. Experimental results show that when trained on the same 1T-token dataset, Ling-mini-beta achieves a lower final training loss and exhibits a slight performance advantage across a suite of downstream tasks. This outcome confirms our theoretical prediction, validating that this MoE architecture yields an efficiency gain of over 7x. These findings provide a solid theoretical and empirical foundation for the design of future large-scale, efficient MoE models.

2 Preliminary

2.1 Mixture-of-Expert Transformers.

MoE architecture modifies the standard Transformer by replacing each Feed-Forward Network (FFN) block with an MoE layer. This layer consists of multiple expert networks (or simply "experts") and a gating mechanism. For each input token, the gating mechanism dynamically routes it to a small subset of these experts. This selective activation of experts for each token significantly reduces the computational cost per forward pass compared to a dense model of equivalent parameter count.

Total and Active Parameters. In MoE models, we distinguish between two parameter counts. The *total parameters* (N) encompass all weights in the model, including those of every expert. In contrast, the *active parameters* (N_a) for a given input consist only of the non-expert components and the specific experts selected by the top-k gating mechanism.

Routable and Shared Experts. An MoE layer typically contains two types of experts. First, there are E *routable experts*, from which the gating network selects a subset of E_a (the number of activated experts) for each token. Additionally, many modern MoE architectures incorporate E_s *shared experts*, which are activated for every token to process and consolidate knowledge common to all inputs.

Activation Ratio and Sharing Ratio. We introduce two metrics to characterize the expert configuration. *Activation ratio*, A, is the ratio of activated experts to the total number of experts. *Sharing ratio*, S, is the ratio of shared experts to activated experts. Assuming all experts have identical dimensions, these rates are defined as $A = (E_a + E_s)/(E + E_s)$ and $S = E_s/(E_a + E_s)$. These metrics quantify the sparsity within the MoE layer, offering an intuitive measure of expert utilization.

Granularity of Experts. In conventional MoE architectures, the intermediate dimension of each expert, d_{expert}, is typically equals the feed-forward network (FFN) dimension, which is conventionally set to $4d_{\text{model}}$. However, recent works (DeepSeek-AI, 2024) have diverged from this practice by decoupling the expert dimension from the model's hidden size and the FFN's intermediate dimension. To systematically analyze this design choice, we define *expert granularity* as $G = 2d_{\text{model}}/d_{\text{expert}}$. A higher value of G corresponds to having a larger number of smaller experts for a fixed total parameter count within the MoE layers. It is important to note that, to align with recent leading MoE models (DeepSeek-AI, 2024; Moonshot-AI, 2025), we adopted a different definition of "granularity" from that of Ludziejewski et al. (2024). They define granularity as $4d_{\text{model}}/d_{\text{expert}}$, whereas our definition results in each expert being half the size for the same granularity value, which consequently leads to different observed phenomena.

Defining Model Scale via Computation. We quantify the computational cost using Floating Point Operations (FLOPs). Consistent with prior work (Bi et al., 2024), we define a model's scale in terms of computation, denoted as M, representing the number of non-embedding FLOPs per token in a single forward pass. For MoE models, this is particularly important as M accounts only for the sparsely activated components (*i.e.,* the selected experts). We exclude the embedding layer from this calculation because its contribution to both overall computation and model capacity is minimal. To ensure our analysis is grounded in accurate figures, we employ an exact calculation for M, avoiding error accumulation found in common approximations (details in Appendix C). The total training compute C is thus a function of M and the number of training tokens D:

$$C = M \cdot D \tag{2}$$

This formulation provides a consistent basis for comparing dense and MoE architectures.

2.2 Scaling Laws for MoE Optimal Hyper-parameters

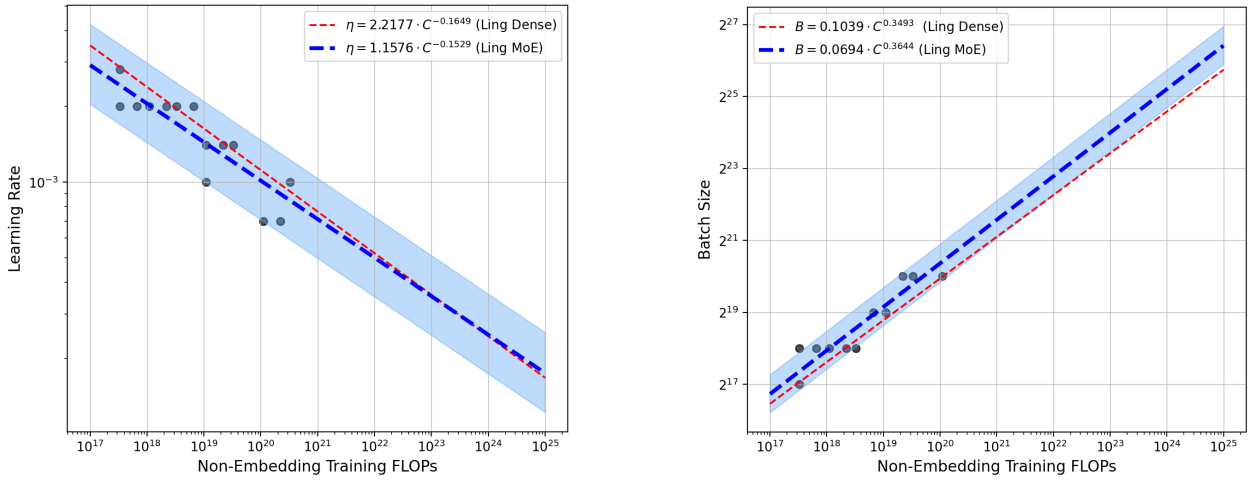
The performance of a MoE model is sensitive to its hyperparameters. To ensure that our subsequent architectural comparisons are reliable, it is crucial to evaluate each configuration under its optimal hyperparameter settings. Therefore, we first conduct a preliminary study to establish the scaling laws for optimal MoE hyperparameters. Previous research (Bi et al., 2024) has established that the

optimal hyperparameters are primarily a function of the total computational budget. Accordingly, we performed a hyperparameter search across a compute range of $3e17$ to $3e20$ FLOPs, using a Warmup-Stable-Decay (WSD) learning rate schedule (Hu et al., 2024). We trained multiple models, varying both learning rate and batch size, which were sampled from a log-base-2 grid. Specifically, the exponents for the learning rate ranged from -11 to -9.0, and for the batch size, from 18 to 21. To make this analysis tractable, we initially fixed the MoE configuration to one with 64 experts, of which 4 are activated per token, plus an additional shared expert (resulting in an activation ratio $A = 7.8\%$ and a granularity $G = 2$). Detailed settings of the experimental models are available in the Appendix B. We then verified that the conclusions from this configuration generalize across different activation ratios.

Figure 2 illustrates the fitting process. To ensure robustness, we identify "near-optimal" configurations as those achieving a loss within 0.25% of the minimum for a given compute budget. After removing outliers, we fitted the optimal batch size, B^{opt}, and learning rate, η^{opt}, against the compute budget C. The resulting scaling laws reveal clear trends: B^{opt} increases and η^{opt} decreases with larger C. The final formulas obtained from the fitting process are as follows:

$$\begin{aligned} \eta^{\text{opt}} &= 1.1576 \cdot C^{-0.1529} \\ B^{\text{opt}} &= 0.0694 \cdot C^{0.3644} \end{aligned} \tag{3}$$

A key finding emerges when comparing these laws to those of dense models. As shown in Figure 2, MoE models favor a significantly larger batch size and a slightly lower learning rate at large compute scales. This phenomenon is attributable to MoE's sparsity: during backpropagation, each expert's parameters are updated using only a subset of the tokens in a batch, whereas dense parameters receive gradients from the entire batch (Sun et al., 2024).

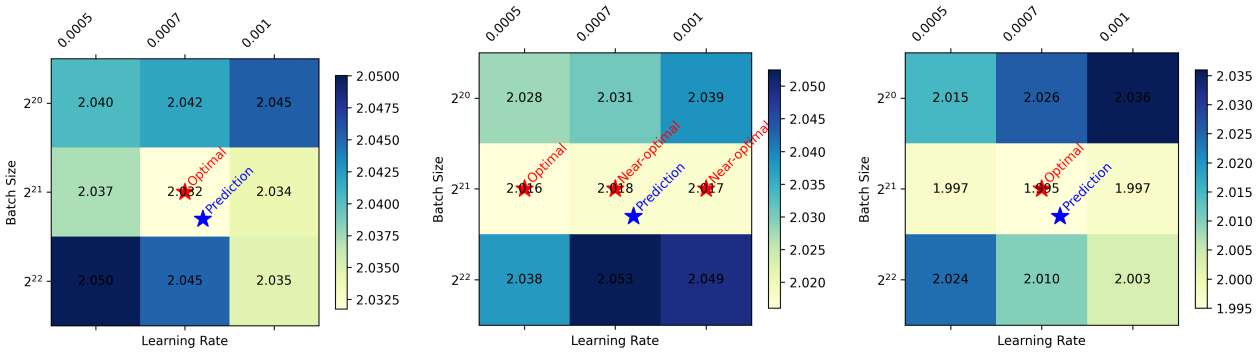


(a) Learning rate scaling curve

(b) Batch size scaling curve

Figure 2 Scaling laws for optimal hyperparameters. Blue and red lines represent the fitted laws for MoE and dense models, respectively, derived on the same training dataset. Gray circles are the experimental data points used for fitting.

To validate the generalizability of these laws, we conduct experiments on MoE models with varying activation ratios. We used the derived laws to predict optimal hyperparameters at a compute budget of $3e20$ FLOPs, after fitting them on data up to $1e20$ FLOPs. As shown in Figure 3, the predicted optimal regions effectively capture the best-performing hyperparameters for activation ratios from 4.7% to 10.9%, demonstrating that the laws can be applied to MoE models within this



(a) $A = 10.9\%$ **(b)** $A = 7.8\%$ **(c)** $A = 4.7\%$

Figure 3 Validation of MoE hyperparameters scaling laws across different activation ratios (A). *"Near-optimal"* refers to hyperparameters achieving a loss within 0.25% of the optimal ones.

range of activation rates. This confirms that our hyperparameter scaling laws provide a reliable foundation for exploring diverse MoE architectures under fair and near-optimal training conditions.

2.3 Scaling laws for MoE Optimal Model-Data Allocation

To determine optimal allocation between model size and data size, we analyze loss trajectories across FLOPs budgets from hyperparameter scaling experiments. By identifying the (M, D) combination that yields the minimum loss for a fixed FLOP budget, we derive optimal allocation strategies for specific MoE configurations activating 4 of 64 experts and an additional shared expert ($A = 7.8\%$, $G = 2$). Crucially, MoE capacity exhibits strong dependence on activation ratio. Thus, this analysis aims to deepen our understanding of MoE architectures and to provide general guidance for model selection in subsequent experiments. The problem can be formally defined as:

$$(M^{\text{opt}}, D^{\text{opt}}) = \arg\min_{M,D} \mathcal{L}(M, D; C, A, G, S) \quad \text{s.t.} \quad C = M \cdot D \tag{4}$$

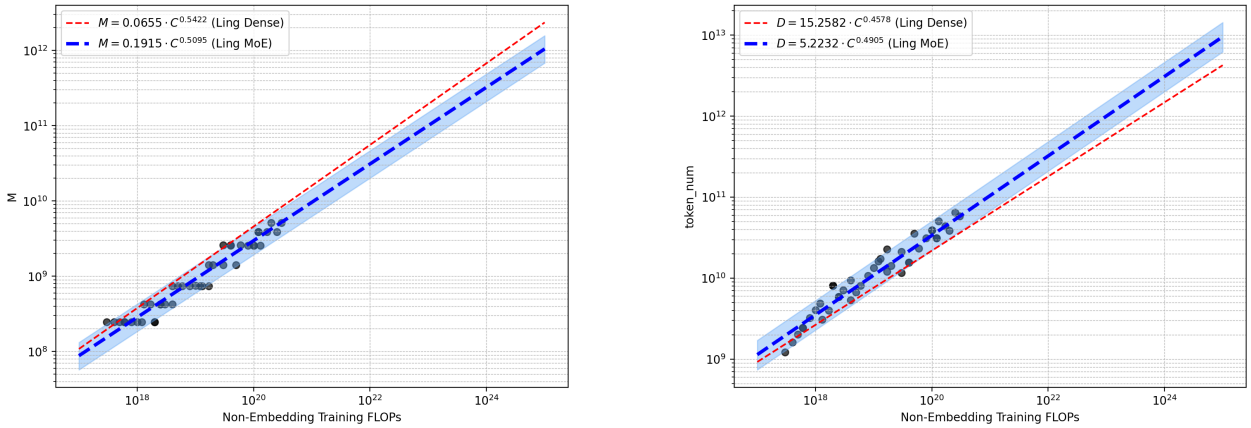
The resulting scaling laws for the optimal model size (M^{opt}) and data size (D^{opt}) are presented in Figure 4 and summarized in Table 1. For comparison, we derive the same laws for dense models. Our analysis yields two key insights:

1. The optimal allocation coefficients for different architectures are similar and close to 0.5. This aligns with findings from previous studies (Bi et al., 2024; Hoffmann et al., 2022), indicating that for compute-optimal training, the budget should be split roughly equally between increasing model size and data volume.
2. Crucially, at any given compute budget, the optimal MoE model is computationally smaller (lower M^{opt}) but trained on more data (larger D^{opt}) than its optimal dense counterpart. This suggests that MoEs possess greater capacity, enabling them to support larger training datasets with smaller model sizes. In real-world scenarios where data is abundant but computational resources are limited, this is significant for improving efficiency.

While practical training strategies may deviate from this compute-optimal allocation, these scaling laws provide a crucial reference. They offer a principled basis for determining the necessary amount of training data for a given model to approach convergence, designing informative ablation studies, and ultimately, developing more efficient MoE architectures.

Table 1 Scaling law parameters for compute-optimal allocation of model scale (M^{opt}) and data size (D^{opt}) for MoE and dense models on identical datasets.

	Optimal Model Scale (M^{opt})	Optimal Data Size (D^{opt})
Dense	$M^{\text{opt}} = 0.0655 \cdot C^{0.5422}$	$D^{\text{opt}} = 15.2582 \cdot C^{0.4578}$
MoE	$M^{\text{opt}} = 0.1915 \cdot C^{0.5095}$	$D^{\text{opt}} = 5.2232 \cdot C^{0.4905}$



(a) Optimal Model Scale (M^{opt}) Scaling

(b) Optimal Data Size (D^{opt}) Scaling

Figure 4 Scaling laws for optimal model scale (M^{opt}) and data size (D^{opt}) on identical datasets. For a given budget, MoE models (blue) optimally allocate more resources to data and fewer to model size compared to dense models (red).

3 Efficiency Leverage: Metric for Quantifying MoE Compute-Efficiency

Next, we define the efficiency leverage and use it to outline our objectives and roadmap.

Definition of Efficiency Leverage. To quantify the computational efficiency gain of MoE compared to dense models, we introduce core metric of **Efficiency Leverage (EL)**. Let $\mathcal{X}_{\text{Dense}}$ denote a standard dense architecture and $\mathcal{X}_{\text{MoE}}$ represent a MoE architectures. Within $\mathcal{X}_{\text{MoE}}$, models share identical core configurations (attention mechanisms, expert count, granularity, shared experts), scaled solely through hidden dimensions (d_{model}, d_{ffn}, d_{expert}) and layer count n_{layer}. Formally, we define the EL of $\mathcal{X}_{\text{MoE}}$ as the ratio of compute budgets required for the dense and MoE models to achieve the same performance level. While model performance can be quantified through loss values, benchmark scores, or task-specific metrics, this study adopts loss as the primary metric.

Definition 3.1 (Efficiency Leverage)**.** For $\mathcal{X}_{\text{MoE}}$ achieving minimal loss $\mathcal{L}(C_{\text{moe}}; \mathcal{X}_{\text{MoE}})$ at compute budget C_{moe}, assuming there exists a compute budget C_{dense} such that $\mathcal{X}_{\text{Dense}}$ attains comparable minimal loss $\mathcal{L}(C_{\text{dense}}; \mathcal{X}_{\text{Dense}})$, we define the efficiency leverage as:

$$\begin{aligned} EL(\mathcal{X}_{\text{MoE}} \mid \mathcal{X}_{\text{Dense}} \,;\, C_{\text{target}}) &= \frac{C_{\text{dense}}}{C_{\text{moe}}}, \\ \text{s.t.} \quad |\mathcal{L}(C_{\text{moe}}; \mathcal{X}_{\text{MoE}}) - \mathcal{L}(C_{\text{dense}}; \mathcal{X}_{\text{Dense}})| &\leq \epsilon \quad (\epsilon \to 0) \end{aligned} \tag{5}$$

Here, the minimal loss achievable by an architecture under specific computational constraints represents its performance ceiling at that scale. An EL greater than 1 signifies that the MoE architecture is more computationally efficient than the dense model, achieving the same performance with less compute. Conversely, an EL less than 1 indicates inferior efficiency.

Following established practice (Kaplan et al., 2020), we model the relationship between compute (C) and loss ($\mathcal{L}$) with a power law: $\mathcal{L}(C; \mathcal{X}) = \alpha_{\mathcal{X}} C^{-\beta_{\mathcal{X}}}$. This allows us to simplify the EL definition in the compute-optimal training regime (Hoffmann et al., 2022) or similar over-training regime (Gadre et al., 2024). Given the computational cost $C = M \cdot D$, the efficiency leverage simplifies under fixed data size D to the ratio of model scales: $EL(\mathcal{X}_{\text{MoE}} \mid \mathcal{X}_{\text{Dense}}) \approx M_{\text{Dense}}/M_{\text{MoE}}$. This formulation demonstrates that EL quantifies the relative model scale of $\mathcal{X}_{\text{MoE}}$ compared to $\mathcal{X}_{\text{Dense}}$ in achieving equivalent performance. *In other words, given the model scale of an MoE and its corresponding efficiency leverage, we can directly determine the equivalent dense model scale required to achieve the same performance.*

Objective and Roadmap. Existing studies (Ludziejewski et al., 2024; Abnar et al., 2025; Clark et al., 2022) indicate that the model capacity of MoE is significantly influenced by architectural configurations. The primary objective of this work is to understand and quantify how MoE architectural choices influence Efficiency Leverage. Our central research question is:

How do the architectural configurations of an MoE model affect its Efficiency Leverage, and how does this relationship scale with the computational budget?

Specifically, our investigation focuses on three critical architectural dimensions[1]: the *Activation Ratio* (A), *Expert Granularity* (G), and *Shared Expert Ratio* (S). They jointly determine the effective capacity of MoE models, and can be used to derive other MoE configurations (*e.g.*, the number of experts, the number of actived experts) based on the definitions in Section 2.1. Our goal is to find the configuration $(A^{\text{opt}}, G^{\text{opt}}, S^{\text{opt}})$ that maximizes EL for a given compute budget C:

$$(A^{\text{opt}}, G^{\text{opt}}, S^{\text{opt}}) = \arg \max_{(A,G,S) \in \mathcal{X}_{\text{MoE}}} EL(\mathcal{X}_{\text{MoE}} \mid \mathcal{X}_{\text{Dense}} \,;\, C) \tag{6}$$

To make the analysis tractable, we assume the effects of these dimensions are largely independent and conduct systematic ablation studies. We start with a baseline MoE architecture (2 of 64 experts activated, plus one shared expert) and vary one dimension at a time across a range of compute budgets (from $3e18$ to $3e20$ FLOPs). To ensure a fair and robust comparison, we leverage the findings from our preliminary studies (Sections 2.2 and 2.3). For each architecture and compute budget, we determine the reasonable model size (M) and data size (D) using our derived allocation laws and configure training with optimal hyperparameters from our hyperparameter scaling laws. This rigorous protocol ensures that each architecture is evaluated at or near its peak potential for a given budget, yielding reliable and cost-effective conclusions. Further details on the experimental setup are provided in Appendix B. Next, we first empirically analyze the impact of each dimension on EL, and then integrate our empirical findings into a unified scaling law to models the relationship between MoE configurations and the resulting EL.

4 Scaling Laws for Efficient MoE Architecture

To achieve greater leverage, we first conduct an extensive empirical study on the architectural configurations of MoE and derive scaling laws for efficient MoE architectures.

4.1 Empirical Study on the Interplay between Efficiency Leverage and MoE Architecture

To identify the MoE architecture that maximizes Efficiency Leverage (EL) for a given compute budget, we systematically investigate the impact of several key design choices. These include

[1]Other architectural configurations, such as the arrangement of MoE and dense layers, have been verified to have a secondary impact on the efficiency leverage of MoE. See Appendix B and Appendix D for details.

the activation ratio, expert granularity, shared expert ratio, and other configurations. For each architectural dimension, we vary it systematically while holding other factors and the model scale M constant. To ensure a fair comparison, all models are trained following the configurations derived from our scaling laws (Section 2), which specify the ideal model size (M), data volume (D), and hyperparameters for any given total compute budget. Guided by the scaling laws for optimal model-data allocation (defined in Section 2.3, we train each model on over three times its optimal number of tokens. This was done to simulate the overtrained state commonly observed in real-world scenarios. All of trained models can be found in Appendix F. Based on the observed training dynamics, we plot the resulting loss curves and EL trends to isolate and quantify the influence of each design choice. To ensure robust analysis, we presuppose a standard power-law relationship between FLOPs cost and training loss, and observe the loss of experimental models after sufficient training using the theoretically optimal allocation as a reference.

4.1.1 Optimal Expert Activation Ratio

We begin by investigating the activation ratio (A), a critical factor governing MoE efficiency. Our experimental design isolates the effect of A by holding the computational cost per token (M) constant. This is achieved by fixing the number of activated experts and their granularity, while varying the total number of experts in the pool from 2 to 256. This setup allows us to explore a wide range of activation ratios (from 0.8% to 100%, where 100% represents a dense model) without altering the forward pass FLOPs. The optimization problem for a given compute budget C is thus:

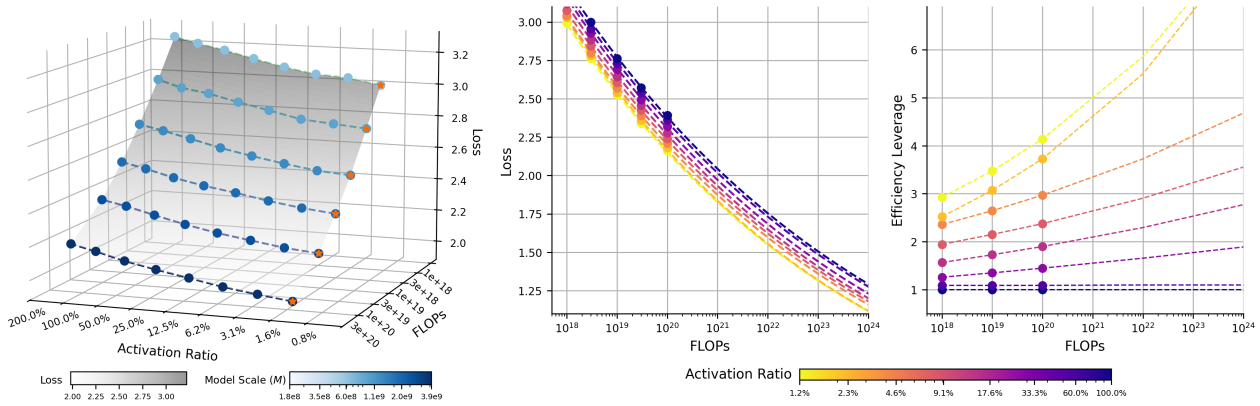
$$A^{\text{opt}} = \arg\min_{A} \mathcal{L}(A; C, M, G, S) \tag{7}$$

The IsoFLOPs curves, presented in Figure 5a, reveal a clear and consistent trend. Across all tested FLOPs budgets (from $1e18$ to $3e20$), loss monotonically decreases with activation ratio, following a power-law pattern. For all configurations, the lowest tested ratio of 0.8% consistently yields the minimum loss. This finding suggests a core principle: for a fixed computational cost, greater model sparsity (*i.e.,* lower activation ratio) leads to higher parameter efficiency.

To quantify this efficiency improvement, we fit a series of loss scaling curves at different activation ratios. Based on these curves, we compute the efficiency leverage for different activation ratios and FLOPs budgets, as illustrated in Figure 5b. The results reveal two key trends. First, for a fixed FLOPs budget, the EL consistently increases as the activation ratio decreases, indicating that sparse activation can always enhance computational efficiency. Second, for a fixed activation ratio, the EL grows with the computational budget, demonstrating that the MoE advantage is amplified at larger scales. These findings confirm that reducing the activation ratio yields substantial efficiency gains, and these benefits are magnified in large-scale, high-computation regimes.

Key Takeaway 1

- **Monotonic Relationship Between Efficiency and Activation Ratio.** For a fixed computational cost, model performance consistently improves as the activation ratio decreases. This indicates a direct, monotonic relationship between sparsity and efficiency.
- **Efficiency Gains Amplify with Scale.** The efficiency advantage of MoE models (their EL) grows with the total training budget. This highlights their suitability for large-scale training, where their benefits become even more significant.



(a) IsoFLOPs curves for varying A

(b) Loss and EL scaling curve over varying A.

Figure 5 Impact of the Activation Ratio A on Loss and Efficiency. (a) At any fixed compute budget (each colored line), lower activation ratios yield lower loss. The orange stars mark the optimal (lowest) loss point. (b) Loss and EL scaling curves illustrate that EL increases with both higher compute budgets and lower activation ratios, showing that MoE advantages are magnified at scale.

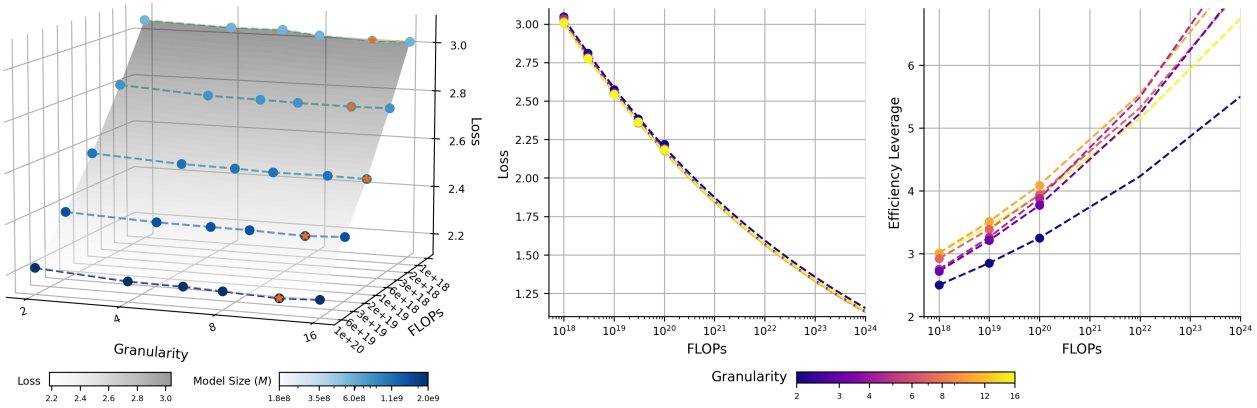
4.1.2 Optimal Granularity of Experts

The granularity of experts is a critical factor in the efficiency of MoE. While prior works (Ludziejewski et al., 2024; Deepseek-AI et al., 2024) suggests that finer-grained experts improve performance, the optimal balance remains an open question. To investigate the influence of expert granularity on MoE efficiency, for a fixed model size M and activation ratio A, we vary the expert granularity from 2 to 16 by increasing the total number of experts from 64 to 512 while proportionally decreasing the size of each expert to keep computational cost (FLOPs) per token constant. This creates a spectrum of models from coarse-grained (fewer, larger experts) to fine-grained (more, smaller experts). By training these models and comparing their final training losses, we can identify the granularity that yields the best performance for a given FLOPs budget. This problem is formalized as:

$$G^{\text{opt}} = \arg\min_{G} \mathcal{L}(G; C, M, A, S) \tag{8}$$

where G^{opt} is the optimal granularity that minimizes the training loss $\mathcal{L}$ under a fixed FLOPs budget C, model size M, activation ratio A, and shared expert ratio S. As shown in Figure 6a, our experiments across a range of FLOPs budgets (10^{18} to 10^{20}) reveal a distinct trend. For any given budget, as we increase expert granularity, the training loss first decreases and then, after reaching a minimum, begins to increase. This demonstrates the existence of an optimal expert granularity that maximizes computational efficiency of MoE. To further analyze this relationship, we fit loss scaling curves for different granularities (Figure 6b), quantifying their impact on EL.

Our study yields two primary insights: First, for a fixed FLOPSs budget, the training loss follows a U-shaped (polynomial) relationship with respect to expert granularity, which confirms an optimal point for maximizing model performance per FLOP. This finding contrasts with the conclusions of Ludziejewski et al. (2024), and we detail the reasons for this discrepancy in Section 6.1. Second, across different FLOPSs budget, the optimal granularity remains within a stable range (around 12 in our experiments), offering a reliable heuristic for model design. Furthermore, we find that routing balance significantly impacts the choice of optimal granularity. Poor routing balance shifts the optimal point towards coarser granularities and degrades overall model performance (see Appendix D for details). This suggests that improving routing mechanisms could unlock the potential of even more fine-grained MoEs, marking a promising direction for future work.



(a) IsoFLOPs curves over varying G. **(b)** Loss and efficiency leverage scaling curve over varying G.

Figure 6 Impact of the Experts Granularity G on Loss and Efficiency. (a) IsoFLOPs curves reveal a U-shaped (polynomial) relationship between expert granularity and training loss. Orange stars mark the optimal granularity for each FLOPs budget. (b) Loss and EL scaling curves show that MoE efficiency improves as FLOPs increase and expert granularity approaches the optimal range.

Key Takeaway 2

- **Existence of Optimal Expert Granularity.** For a fixed FLOPs budget and model scale, training loss exhibits a U-shaped (polynomial) relationship with expert granularity, indicating an optimum that maximizes efficiency.
- **Stable Range of Optimal Expert Granularity.** The optimal granularity (*e.g.*, around 12 in our experiments) is stable across a wide range of FLOPs budgets. However, poor routing balance shifts this optimum toward coarser granularity.

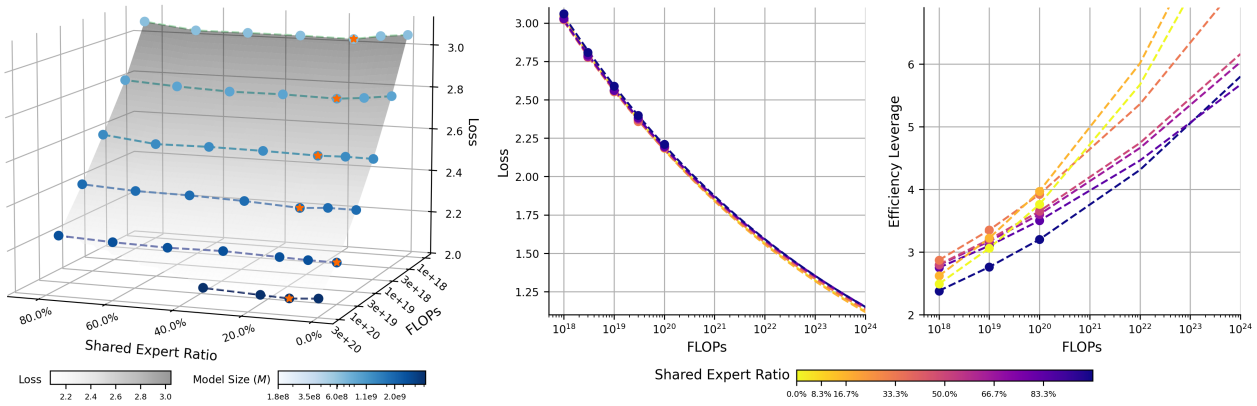
4.1.3 Optimal Shared Expert Ratio

Shared experts are always active to capture common knowledge (Deepseek-AI et al., 2024). To determine the optimal proportion of shared experts, we designed a series of experiment to isolate the impact of the shared expert ratio S. We fix the total model size M, the activation ratio A, and the total number of active experts ($E_s + E_a$). We then systematically vary S by substituting routed experts (E_a) with shared experts (E_s), exploring configurations from fully specialized ($S = 0\%$) to highly shared ($S = 83.3\%$). This allows us to identify the optimal ratio that minimizes training loss for a given computational budget. The problem is formalized as:

$$S^{\text{opt}} = \arg\min_{S} \mathcal{L}(S; C, M, A, G) \tag{9}$$

where S^{opt} is the optimal shared expert that minimizes the training loss $\mathcal{L}$ under a fixed FLOPs budget C, model size M, activation ratio A, and granularity G. Our experiments, as depicted in Figure 7a, reveal a U-shaped relationship between the shared expert ratio and training loss. The minimum loss is generally achieved at a relatively low shared expert ratio, while having no shared experts ($S = 0\%$) usually results in suboptimal performance. Furthermore, we observe a subtle trend where the optimal sharing ratio appears to scale with the compute budget. This is supported by our empirical scaling law (EL) analysis in Figure 7b, which shows that lower FLOPs budgets ($\leq 10^{20}$) benefit from a slightly higher sharing ratio ($S = 16.7\%$), whereas larger budgets ($> 10^{20}$) achieve greater efficiency with a lower ratio ($S = 8.3\%$).

Since large-scale pre-training runs typically exceed 10^{20} FLOPs, this suggests a practical heuristic: the optimal design choice is to use the lowest possible non-zero sharing ratio. Assuming the dimensions of shared and regular experts are equal, this can be heuristically implemented by setting the number of shared experts to one.



(a) IsoFLOPs curves over varying G.

(b) Loss and efficiency leverage scaling curve over varying G.

Figure 7 Impact of the Shared Ratio S on Loss and Efficiency. (a) Loss curves demonstrate that a low, non-zero sharing ratio minimizes training loss, outperforming both no shared experts ($S = 0\%$) and highly shared configurations.. (b) EL analysis reveal that the optimal sharing ratio is higher ($S = 16.7\%$) for smaller FLOPs ($< 10^{20}$) and decreases to $S = 8.3\%$ for larger FLOPs ($> 10^{20}$).

Key Takeaway 3

- **Optimal Sharing Ratio Exhibits a Subtle Scaling Trend.** We identify a subtle scaling trend between the optimal shared expert ratio and the compute budget: the ideal ratio decreases as the compute budget increases.
- **"One Shared Expert" Rule for Large-Scale Training.** For large-scale pre-training with uniformly sized experts, the optimal design heuristic is to employ a single shared expert. This configuration establishes the minimal non-zero sharing ratio.

4.1.4 Other Configurations of MoE Architecture

To further optimize the efficiency of MoEs , we also explore two design dimensions: arrangement of MoE and dense layers and compute resource allocation between attention and FFN. The detailed experimental results can be found in Appendix D.

First, we analyze replacing the initial MoE layers with dense layers while keeping total FLOPs constant (e.g., 60-layer models with the first 1–3 layers set dense). The experimental results show that replacing the first few layers with dense layers has a minor impact on performance, with efficiency leverage close to 1 within a FLOPs budget of up to $3e20$ FLOPs. This adjustment reduces the total number of parameters and mitigates routing imbalances, making it a valuable design optimization. As FLOPs budgets increase, the optimal dense proportion also grows; for example, at $1e18$ FLOPs, the optimal dense proportion is zero. As the compute budget increases to $3e20$ FLOPs, the optimal dense layer proportion shifts to approximately $2/60$ or $3/60$.

Second, we explore the impact of computational allocation between the attention mechanism and FFN on MoE efficiency. By constructing models with varying attention-FFN compute allocation and observing performance changes, we find that: 1) An attention FLOPs ratio of 30%–40%

ensures stable performance, corresponding to default settings in MoE models. 2) Broad ratio adjustments (20%–50%) minimally impact performance due to attention's computational density, which increases knowledge density but may raise downstream inference costs

Key Takeaway 4

- **Introducing Dense Layers is a Valuable Design Optimization.** Incorporating dense layers in the early stages of MoE has minor impact on efficiency but helps mitigate routing imbalances and reduces overall parameters. The optimal proportion of dense layers increases with higher FLOPs budgets, though it offers limited efficiency gains.
- **Robustness of Compute Budget Allocation between Attention and FFN** Allocating 30%-40% of FLOPs to the attention mechanism achieves optimal or near-optimal performance, with minor impact outside this range. Increasing attention FLOPs proportion enhances knowledge density but reduces downstream inference efficiency.

4.2 Scaling Laws for MoE Efficiency Leverage

From the observations, both the dense layer and attention ratio have minimal impact on MoE's efficiency leverage. While sharing experts is broadly beneficial, employing a single shared expert typically emerges as the optimal choice in practice. Thus, we aim to derive a parametric scaling law for predicting the efficiency leverage based on activation ratio A, granularity G, and FLOPs C.

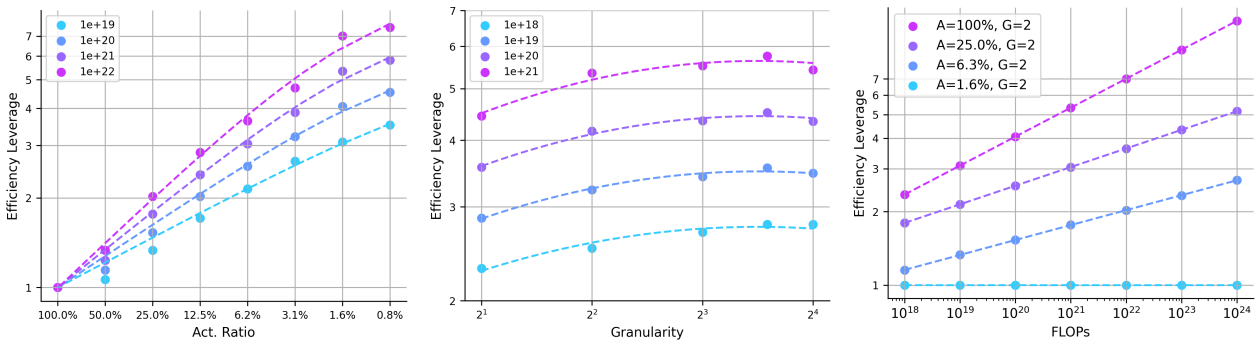
4.2.1 Separable Scaling Laws for Efficiency Leverage

Based on a large amount of previous empirical study in Section 4.1, we collect the MoE efficiency leverages under different settings and summarized them to be presented in Figure 8.

Interaction of Efficiency Leverage and Activation Ratio. Based on the preceding observations, activation ratio is identified as the primary factor influencing the computational efficiency of MoEs. As illustrated in Figure 8a, reducing the activation ratio (*i.e.*, increasing sparsity) consistently yields substantial efficiency gains, following a similar power-law relationship across different FLOPs budgets. Consequently, we propose the following hypothesis: for a given FLOPs budget and granularity, there exists a power-law dependence between efficiency leverage and activation ratio.

$$\begin{aligned} \log EL_{C,G}(\hat{A}) &= a_A \log \hat{A}, \quad \text{i.e. } EL_{C,G}(\hat{A}) = \hat{A}^{a_A}, \\ \text{where } \frac{1}{\hat{A}} &= \frac{1}{A + (1/A_{start} - 1/A_{max})^{-1}} + \frac{1}{A_{\max}}, \end{aligned} \tag{10}$$

where $\hat{A}$ is a saturating transformation of A, as defined in Clark et al. (2022), and we set the lower bound of meaningful activation ratio as 0. Clearly, when $A = 1$, we have $EL = 1$, indicating that the EL of the dense model is 1, which satisfies the dense equivalence. We fit Eq. (10) to each FLOPs budget and plot the predictions for varying activation ratios as dotted lines in the Figure 8a. The predictions align well with the observed data. Notably, we observe that a_A increases as A decreases and C increases. This trend corresponds to a diminishing benefit from increased sparsity, consistent with findings from prior research (Clark et al., 2022). Additionally, a_A also increases with C, suggesting a greater benefit from the bigger compute budget (Ludziejewski et al., 2024). We will analyze the relationship between FLOPs and EL in the following paragraph.



(a) Scaling with Activation Ratio (A) **(b)** Scaling with Granularity (G) **(c)** Scaling with Compute Budget (C)

Figure 8 Scaling Behavior of Efficiency Leverage (EL). (a) With fixed granularity ($G = 2$), EL follows a power law with respect to A across all tested compute budgets (C). (b) With a fixed activation ratio ($A = 3.1\%$), EL's scaling with G conforms to a log-polynomial law across all compute budgets. (c) With both activation ratio (A) and granularity (G) held constant, EL scales with compute according to a standard power law.

Interaction of Efficiency Leverage and Granularity. As previously mentioned, the relationship between expert granularity and EL does not exhibit an ideal power-law pattern. Instead, there exists an optimal granularity that maximizes the EL. Based on this observation, we hypothesize that under fixed FLOPs budget C and activation ratio A, the relationship between EL and granularity G follows a log-polynomial pattern:

$$\log EL_{C,A}(G) = a_G + b_G \left(\log G \left(\log G + c_G\right)\right), \tag{11}$$

where a_G is the granularity-independent base EL. It indicates the theoretical EL value of the model when the expert granularity is 1. b_G controls the strength of the curvature in the relationship between EL and granularity, reflecting the sensitivity of the model architecture to changes in expert granularity. c_G directly determines the location of the optimal granularity that maximizes EL. We fit Eq. (11) to each FLOPs budget and plot the predictions for varying granularity as dotted lines in the Figure 8b. As shown in the figure, the curves under different FLOPs are identical (i.e., with similar values of b_G and c_G), indicating that the impact of expert granularity on MoE efficiency remains consistent across various computational budgets.

Interaction of Efficiency Leverage and Compute Budget. Based on the analysis presented in Section 4.1, we observe that the efficiency advantage of MoE increases as the computational budget grows. To formalize the relationship between the FLOPs budget and Efficiency Leverage, we assume a standard power-law pattern as follows:

$$\log EL_{A,G}(C) = a_C \log C + c_C, \quad \text{i.e. } EL_{A,G}(C) = \exp(c_C) \cdot C^{a_C}, \tag{12}$$

where a_C reflects the scaling capability of MoE efficiency with respect to the computational budget under given configurations A and G. We collect the values of the EL corresponding to different model architectures under the granularity setting of 2, and fit Eq. (11) to each architectures. The predictions for varying granularity are plotted as dotted lines in the Figure 8c. The results indicate that all tested MoE architectures show a trend of higher EL as the FLOPs budget increases, demonstrating the potential of MoE in large-scale pre-training.

4.2.2 Joint Scaling Law for Efficiency Leverage

Based on the preceding observations, we arrive at the following key insights:

- The activation ratio (or sparsity) is the core determinant of MoE efficiency, establishing its foundational power-law scaling.
- Building upon this power law, expert granularity adds a non-linear adjustment that operates independently of the compute budget.
- Furthermore, the efficiency advantage of MoE over dense models is amplified by the compute budget C through the power-law term.

To unify these interconnected effects, we derive a joint scaling law for EL, formulated as follows:

$$EL(A, G, C) = \hat{A}^{\alpha+\gamma(\log G)^2+\beta \log G}, \tag{13}$$

where $\alpha = a + d \log C$ is the compute-dependent exponent that captures the primary power-law relationship between EL and activation ratio. The term a represents the base scaling exponent at a reference compute budget, while d is a positive constant that quantifies how the EL is amplified by a larger compute budget C. The parameters β and γ model the non-linear impact of granularity G. This quadratic form in $\log G$ directly reflects the log-polynomial pattern observed in our initial analysis, capturing the existence of an optimal granularity.

To validate the proposed scaling law for EL, we fit Eq. (13) using Huber loss and the BFGS optimization algorithm (Hoffmann et al., 2022). We use data points with an EL factor below 6 for training, while those are reserved as a validation set. We depict the results in Figure 9. The values are presented in Table 2. The alignment between the scaling law and both the training data and validation set provides strong empirical support for the proposed relationship. More importantly, the scaling law exhibits remarkable extrapolation capabilities, as it accurately models performance trends for high-leverage validation points outside the training range. These results confirm that Eq. (13) effectively captures the underlying interaction between MoE architecture and EL.

Furthermore, we select two compute budget (1e20 and 1e22 FLOPs), and apply our fitted scaling laws to predict efficiency leverage across various MoE configurations. As shown in Figure 1, our analysis predicts that an efficiency leverage (EL) exceeding 7x can be achieved at a budget of 1e22 FLOPs with an activation ratio of 3.1% and a granularity of 12. The subsequent section provides experimental validation for this specific claim.

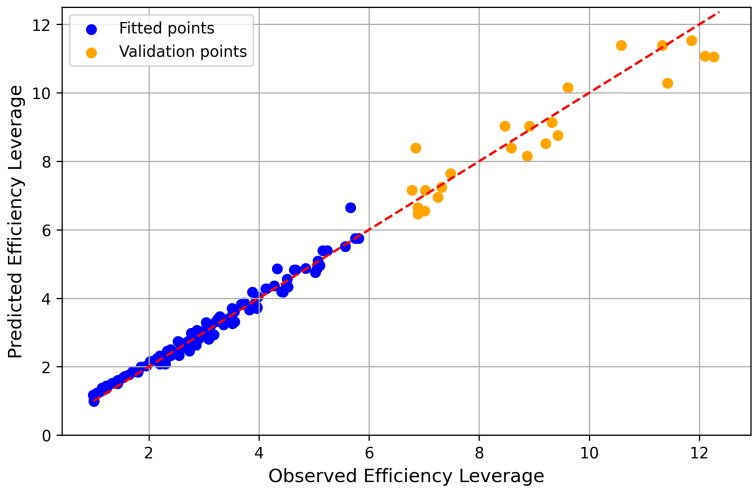


Figure 9 Validation of the Scaling Laws for MoE Efficiency Leverage. We fit Eq. (13) to the data points with an efficiency leverage of less than 6, using the remaining points as the validation set.

Table 2 Values of the Fitted Coefficients.

a	d	γ	β	A_{start}	A_{max}
1.23	-7.61e-2	1.67e-2	-1.17e-1	1.63e-2	5.28e+16

5 `Ling-mini-beta`: More Efficient MoE Language Model

Based on the findings in Section 4, we identify the efficient architectural configuration within the current MoE framework. To validate the effectiveness of this configuration, we design an MoE model with 0.855 billion active parameters out of a total of 17.5 billion (referred to as "`Ling-mini-beta`," a pilot model for Ling-2.0 series) and test it using 1 trillion tokens of training data. `Ling-mini-beta` is configured with a granularity G of 12 and an activation ratio A of just 3.4%. Referring to Figure 1, at the $1e22$ FLOPs compute budget, we hypothesize that `Ling-mini-beta` achieves ***more than 7× in compute-efficiency leverage*** over a comparable dense model. Concurrently, we train a traditional dense model with 6.1 billion parameters (named "`Dense-6.1B`") for comparison. This section presents a detailed analysis of the performance differences between `Ling-mini-beta` and the conventional dense model `Dense-6.1B`, highlighting that the active parameter count, training costs, and downstream inference costs of `Dense-6.1B` are more than seven times those of `Ling-mini-beta`.

5.1 Model and Training Details

The architectures of `Ling-mini-beta` and `Dense-6.1B` are given in Table 3. Other settings include:

- **Model Setting.** `Ling-mini-beta` adopts the same GQA (Ainslie et al., 2023) attention architecture as `Dense-6.1B`, with the only difference being the extension of the original FFN layers to MoE layers. Additionally, both `Ling-mini-beta` and `Dense-6.1B` employ Rotary Position Embedding (RoPE) (Su et al., 2024) and supports a sequence length of 8K.
- **Training Data.** The training data is sourced from a large-scale multilingual corpus created by the Ling Team, primarily covering English and Chinese, while also including various other languages. This corpus encompasses web text, mathematical materials, programming scripts, published literature, and diverse textual content. To validate model performance, we extracted a 1T-token subset from this corpus for training.
- **Training Setting.** Both `Ling-mini-beta` and `Dense-6.1B` were trained using the AdamW optimizer (Loshchilov and Hutter, 2017) with hyperparameters set as follows: $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight decay of 0.1. Gradient clipping norm is set to 1.0. The learning rate schedule employs a WSD (warmup-stable-decay) strategy (Hu et al., 2024). According to the hyperparameter scaling laws for dense and MoE models, the maximum learning rates were set to $3.78e{-}4$ for `Ling-mini-beta` and $2.93e{-}4$ for `Dense-6.1B`. The batch sizes were configured as 1792 and 2048, respectively.

More details about model training setting can be found in the Appendix B.

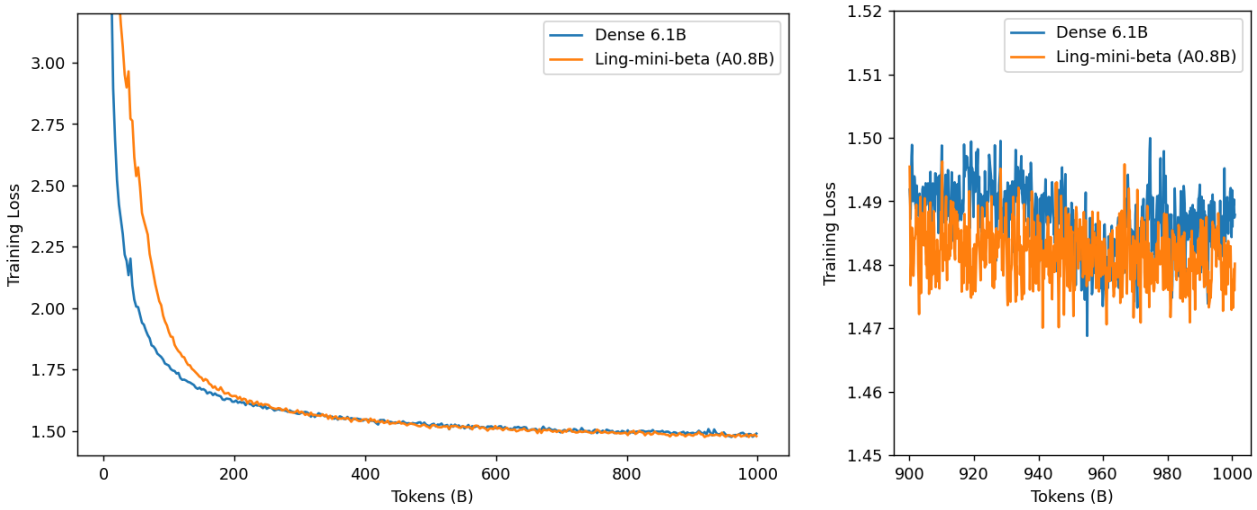
Table 3 Detailed Architectures of `Ling-mini-beta` and Dense Model for Comparison. We determined the architecture of the `Ling-mini-beta` based on the findings of Section 4.

Model	n_{layers}	d_{model}	d_{ffn}	d_{expert}	n_{heads}	n_{kv_head}	E	E_a	E_s	N	N_a
Dense 6.1B	28	4096	14336	-	32	8	-	-	-	6.11B	6.11B
`Ling-mini-beta` (A0.8B)	20	2048	5120	384	16	4	384	12	1	17.5B	0.85B

5.2 Training Dynamics

The Dynamic of Training Loss The training loss curves for `Ling-mini-beta` and `Dense-6.1B`, shown in Figure 10, illustrate a clear difference in their convergence behavior. The dense model exhibits faster convergence during the early training phases, indicating an aptitude for rapid initial learning. In contrast, `Ling-mini-beta`'s loss decreases more gradually at the start. However, over the full course of training, `Ling-mini-beta` steadily improves and ultimately achieves a performance level comparable to that of the dense model, highlighting its ability to reach high performance with sufficient training. Focusing on the final 100 billion tokens of training provides further insight. In this concluding stage, the performance gap between `Ling-mini-beta` and `Dense-6.1B` narrows to a negligible difference of about 0.01 in loss value. This confirms that `Ling-mini-beta` can nearly match the dense model's effectiveness while operating with significantly fewer computational resources. Crucially, this near-equal performance underscores `Ling-mini-beta`'s ability to deliver over 7x gains in training efficiency, making it a highly cost-effective and powerful alternative for large-scale pre-training.

The Dynamic of Benchmarks The training process for both `Ling-mini-beta` and `Dense-6.1B` was monitored by comparing their performance on standard benchmarks. The data reveals a clear and consistent trend: the two models improved almost synchronously. At no point during training did one model show a decisive or lasting advantage over the other. This lockstep progression continued until the end of the training cycle, where they posted nearly identical final scores on the evaluation leaderboard. This synchronous dynamic and convergent outcome suggest a fundamental parity in their learning efficiency and final performance ceiling under our experimental conditions.



(a) Overview of the loss throughout the training process.

(b) Zoom in the loss at the training end.

Figure 10 Dynamic of Training Loss. (a) Comparing the training processes of `Ling-mini-beta` and the dense model shows that the dense model converges faster in the early stages. However, while `Ling-mini-beta` starts slower, its training loss becomes nearly equivalent to the dense model's after sufficient training. (b) Zooming in on the training loss for the final 100B tokens, the training loss difference between `Ling-mini-beta` and `Dense-6.1B` is less than 0.01, demonstrating over 7x efficiency gains for `Ling-mini-beta` with comparable performance to the dense model.

5.3 Evaluation

Evaluation Benchmarks To evaluate performance, we consider a diverse suite of downstream tasks designed to provide a holistic assessment of model capabilities. These tasks are grouped

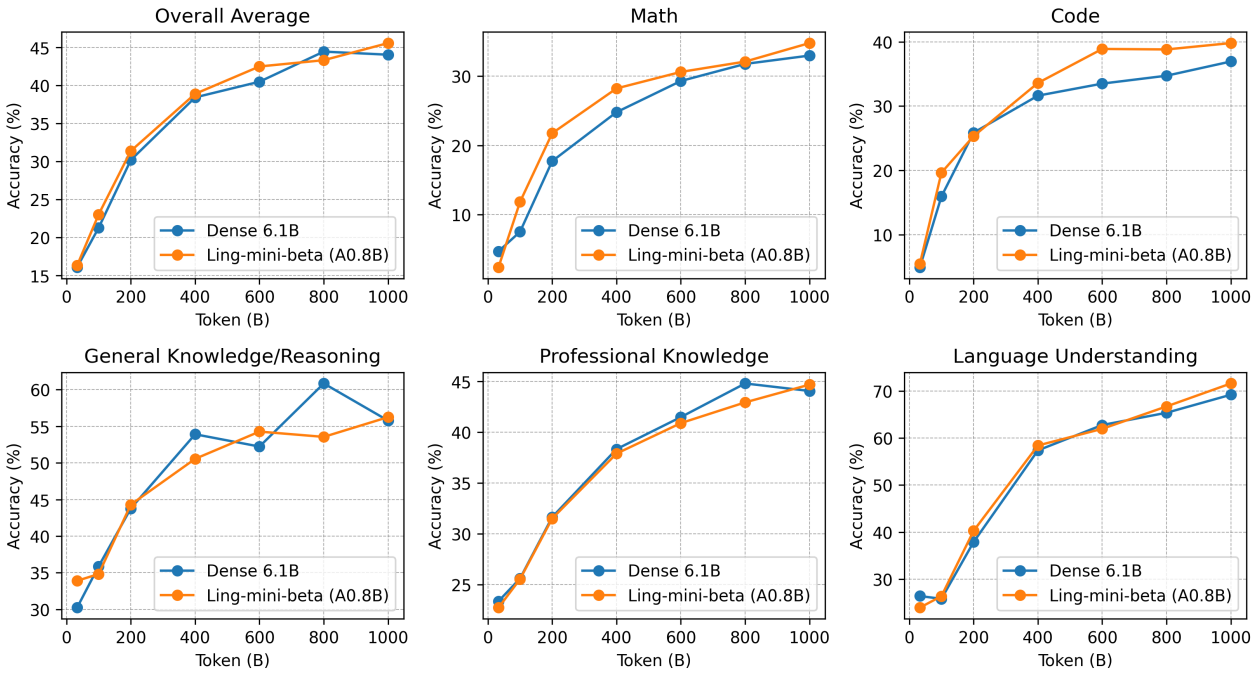


Figure 11 Dynamic of Benchmarks. The comparison of the benchmarks changes between `Ling-mini-beta` and the `Dense-6.1B` during training shows that their performances improved almost synchronously throughout the process, ultimately achieving similar final leaderboard results.

into several categories, such as: (a) General Knowledge/Reasoning (*e.g.,* ARC (Bhakthavatsalam et al., 2021), AGIEval (Zhong et al., 2024), OpenBookQA (Mihaylov et al., 2018), BBH (Suzgun et al., 2023), ProntoQA (Saparov and He, 2023), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), Multi-LogiEval (Patel et al., 2024)) (b) Language Understanding (*e.g.,* RACE (Lai et al., 2017)) (c) Professional Knowledge (*e.g.,* MMLU (Hendrycks et al., 2021a), CMMLU (Li et al., 2024), MMLU-Pro (Wang et al., 2024b), GPQA (Rein et al., 2023), C-Eval (Huang et al., 2023), CommonsenseQA (Talmor et al., 2018)) (d) Math (*e.g.,* GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b), GAOKAO (Zhang et al., 2023), Gaokao2023-Math-En, MGSM (Shi et al., 2023), CMATH (Wei et al., 2023), MathBench (Liu et al., 2024), Minerva-Math (Lewkowycz et al., 2022), CN-Middle School 24) (e) Code (*e.g.,* Humaneval (Chen et al., 2021), HumanEval-cn (Peng et al., 2024), HumanEval-plus (Liu et al., 2023), HumanEval-FIM (Bavarian et al., 2022), LiveCodeBench (Jain et al., 2025), MBPP (Tao et al., 2024), MBPP-Plus (Liu et al., 2023), CruxEval (Gu et al., 2024)).

Evaluation Results The comparative evaluation in Table 4 reveals that `Ling-mini-beta` achieves an average score of 45.5, surpassing `Dense-6.1B`'s 44.0. This result compellingly demonstrates that `Ling-mini-beta` accomplishes a "small yet powerful" feat with significantly lower inference costs, its activated parameters amount to only about *13%* of its competitor's, striking an exceptional balance between performance and efficiency.

Upon closer examination of performance across specific dimensions, `Ling-mini-beta`'s advantages are both comprehensive and focused. In general knowledge and reasoning tasks, it exhibits notable advantages in open-ended question answering tasks such as OpenBookQA and complex logical reasoning benchmarks like Multi-LogiEval. This trend continues in specialized knowledge domains, where `Ling-mini-beta` delivers better results on comprehensive academic benchmarks like MMLU and MMLU-Pro. Its superiority is particularly evident in language understanding

Table 4 Detailed performance comparison of `Ling-mini-beta` (17B-A0.8B) and `Dense-6.1B`.

	Metric	Dense-6.1B	Ling-mini-beta (A0.8B)
General Knowledge /Reasoning	ARC-challenge	59.7	57.0
	ARC-easy	78.0	**78.7**
	AGIEval	33.4	**34.9**
	OpenBookQA	68.6	**75.2**
	BBH	**48.0**	35.7
	ProntoQA	16.5	**19.5**
	Multi-LogiEval	55.6	**61.3**
	HellaSwag	65.6	**66.6**
	PIQA	76.6	**77.2**
	Average	55.8	**56.2**
Professional Knowledge	MMLU	51.1	**53.1**
	MMLU-Pro	21.7	**24.0**
	CMMLU	50.7	**51.9**
	C-Eval	**52.5**	51.1
	CommonsenseQA	**63.6**	60.6
	GPQA	24.8	**27.3**
	Average	44.0	**44.7**
Language Understanding	RACE-middle	73.4	**75.6**
	RACE-high	65.0	**67.6**
	Average	69.2	**71.6**
Code	HumanEval	31.7	**35.4**
	HumanEval-cn	**34.2**	32.3
	HumanEval-Plus	35.4	**51.8**
	HumanEval-FIM	**62.8**	61.3
	MBPP	41.0	**44.6**
	MBPP-Plus	50.0	**51.6**
	LiveCodeBench	**7.5**	7.4
	CruxEval	32.9	**34.1**
	Average	36.9	**39.8**
Math	GSM8K	**59.2**	58.0
	MATH	23.7	**29.8**
	CMATH	60.5	**62.9**
	MGSM-zh	35.6	**36.8**
	CN-Middle School 24	41.6	**42.6**
	Minerva-Math	**3.3**	2.9
	MathBench	27.5	**28.6**
	Gaokao2023-Math-En	33.1	**33.5**
	GAOKAO-Math24	12.1	**17.6**
	Average	32.9	**34.7**
Overall Average		44.0	**45.5**

tasks, as it consistently outperforms its competitor in the RACE series of reading comprehension tests, showcasing stronger contextual understanding capabilities. In tasks requiring high coding proficiency, `Ling-mini-beta` stands out significantly, especially in the HumanEval-Plus benchmark, which measures code robustness, achieving an impressive lead of over *16 points*. Similarly, in mathematical reasoning, while slightly lagging in basic arithmetic tasks like GSM8K, it excels in challenging benchmarks such as MATH and GAOKAO-Math24, demonstrating strong potential in solving complex problems. Collectively, `Ling-mini-beta` achieves a 1.5-point overall advantage,

validating its parameter-efficient MoE design. It not only drastically reduces inference costs through sparse activation but, more critically, its "expert networks" seem to enable higher performance ceilings in key areas such as language understanding, code generation, and advanced reasoning.

Conclusion on `Ling-mini-beta` (17B-A0.8B)

Based on the scaling laws for efficiency leverage in Section 4, we design the `Ling-mini-beta`, a pilot model for the Ling-2.0 series, which has 17.5 B total parameters but only active 0.8 B parameters. Experimental results demonstrate that `Ling-mini-beta` achieves over a 7× efficiency leverage while maintaining comparable performance to dense models with 6.1B, more than 7× the number of active parameters.

6 Discussion and Limitations

6.1 Comparison with Previous Works.

Comparison with Clark et al. (2022). In their study, Clark et al. (2022) used a fixed dataset and concluded that the efficiency of MoE models over dense models diminishes beyond a certain scale. Contrary to the findings of them, our results in Figure 5 demonstrate that MoE models are consistently more compute-efficient than their dense counterparts across all evaluated model sizes. The apparent contradiction can be reconciled by examining the experimental design. Our preliminary studies reveal that for a fixed compute budget, the optimal resource allocation for MoE and dense models differs significantly: MoE models favor fewer parameters and more training tokens, as shown in Section 2.3. Consequently, evaluating all models on a fixed dataset can lead to an unfair comparison, where MoE models are likely under-trained relative to dense models, yielding potentially misleading conclusions. This hypothesis is further corroborated by the convergence dynamics in Figure 10a, which show that MoE models, despite a slower start, eventually outperform dense models as training progresses, which also be verified by Ludziejewski et al. (2024). Therefore, diverging from prior work, our experiments is guided by the scaling laws for optimal compute-allocation. We dynamically scale the number of training tokens with compute budget, ensuring that experimental models achieve a comparable and sufficient degree of training. This approach ensures the fairness and reliability of our comparison.

Comparison with Ludziejewski et al. (2024). Our findings regarding the impact of expert granularity differ from those of Ludziejewski et al. (2024) in two main aspects. First, we observe a log-polynomial relationship between performance and granularity, indicating an optimal granularity range, whereas Ludziejewski et al. (2024) reported a monotonic trend where finer granularity consistently reduces loss. Second, our experiments show that the EL of the MoE is usually within a 10x factor at the tested compute scales—significantly lower than their >10× "Relative FLOPs to train equivalent Transformer" (Figure 1(b) in their paper). We attribute these differences to three key variations in experimental setups: (1) Granularity definition: While Ludziejewski et al. (2024) uses $G = d_{\text{ffn}}/d_{\text{expert}} = 4d_{\text{model}}/d_{\text{expert}}$, our experiments, aligned with leading models (DeepSeek-AI, 2024; Moonshot-AI, 2025), adopts stricter $G = 2d_{\text{model}}/d_{\text{expert}}$. As a result, our experts are effectively half-sized at equivalent nominal granularity (*e.g.*, $G = 16$), enabling exploration of finer actual granularity. (2) Hyperparameter strategies: Unlike their use of uniform hyperparameters across experiments, we optimize these settings for each compute budget, as our preliminary studies in Section 2.2 confirm that optimal configurations vary significantly with compute scale. This avoids inequitable comparisons that may arise from a fixed hyperparameter set. (3) Base MoE ar-

chitectures: Our experiments utilize a base MoE architecture with a 1/32 routable expert activation ratio, whereas their architecture employs a sparser 1/64 ratio. This sparser activation inherently provides higher baseline efficiency, potentially amplifying its measured advantage. In summary, our distinct conclusions arise from investigating a finer granularity spectrum under a different definition and ensuring appropriate training conditions for all models.

Comparison with Abnar et al. (2025). Our findings on the optimal activation ratio align with those of Abnar et al. (2025), confirming that under a fixed compute budget, larger and sparser models yield better performance. However, our research extends beyond this conclusion in both methodology and scope. However, our research substantially extends this direction. First, we determine training hyperparameters through extensive preliminary experiments. Second, we systematically investigate how architectural factors—particularly expert granularity and shared expert ratios—affect model performance. This reveals that beyond the primary activation ratio trend, expert granularity introduces log-polynomial adjustments to performance. Ultimately, our primary contribution is the direct derivation of scaling laws for the efficiency leverage of MoE models relative to their dense counterparts, rather than conventional scaling laws for loss. The key advantage is its independence from specific training datasets. It directly establishes a quantitative relationship between MoE architectural configurations and their relative performance efficiency, offering more generalizable and actionable principles for model design.

Comparison with Ludziejewski et al. (2025). Our research and the work of Ludziejewski et al. (2025) are complementary, with each study addressing a distinct facet of the scaling laws for MoE models. Our work addresses the question: given a fixed compute budget and a specific model scale (*i.e.,* FLOPs per token), how should one configure the architectural parameters (*i.e.,* expert granularity, activation ratio) to maximize performance? In contrast, their study concentrates on a different optimization problem: under the dual constraints of a compute budget and memory limitations, what is the optimal allocation of resources between model scale and data size? While our preliminary experiments did touch upon the model-data allocation for MoE models, this exploration was intentionally limited. It was conducted under a single compute budget constraint and for a specific MoE architecture. Its primary purpose was not to derive a comprehensive allocation strategy, but rather to establish the fundamental differences in optimal resource allocation between MoE and dense models. This foundational understanding was crucial for our main experiments, as it enabled us to provision a sufficient training budget to ensure all models were compared fairly under conditions of adequate, near-optimal training.

6.2 Limitations

Consistent with standard practice in scaling law research (Kaplan et al., 2020; Hoffmann et al., 2022; Clark et al., 2022; Ludziejewski et al., 2024; Abnar et al., 2025), our analysis quantifies computational cost exclusively in terms of theoretical FLOPs. While FLOPs provide a valuable, hardware-agnostic metric for comparing model architectures, we acknowledge that this abstraction does not capture the full spectrum of real-world costs. Factors such as hardware specifications, system infrastructure, and implementation details can introduce discrepancies between theoretical FLOPs and actual wall-clock time. Furthermore, due to significant resource constraints, our methodology relies on the simplifying assumption that the effects of different MoE architectural factors are largely independent. Based on this premise, we conducted a series of individual ablation studies to quantify the impact of each factor in isolation, subsequently synthesizing these effects into a unified scaling law. We acknowledge that a primary limitation of this approach is its potential to overlook

interaction effects between architectural components. Nevertheless,it remains the most pragmatic and feasible pathway within the scope of our available resources. Despite these limitations, our findings underscore the immense potential of MoE models. By enabling a massive increase in model capacity with a minimal increase in per-token computation, they offer a clear path toward improving both model performance and efficiency.

7 Related Work

7.1 Scaling Laws for Language Models

Scaling laws provide a framework for understanding and predicting the performance of language models under varying conditions. Kaplan et al. (2020) laid the foundation by demonstrating that model performance adheres to predictable power-law relationships involving model size, dataset size, and compute budget. Building on this, Hoffmann et al. (2022) introduced the Chinchilla scaling laws, highlighting the importance of balancing model size and training data volume for compute-optimal training. They showed that scaling model size without a corresponding increase in data leads to diminishing performance gains. Sardana et al. (2023) advanced this understanding by incorporating inference costs into compute-optimal frameworks, proposing strategies for optimizing performance under fixed inference constraints. Additionally, Bi et al. (2024) emphasized the critical role of data quality, demonstrating that higher-quality datasets enable more efficient scaling, particularly with larger models. Recent advancements have applied these scaling laws to various specialized areas. For example, hyperparameter optimization has been explored in the context of scaling laws (Bi et al., 2024; Li et al., 2025), while Gadre et al. (2024) investigated the phenomena of over-training and its implications on model performance. Furthermore, scaling laws have been analyzed for their impact on downstream task performance across a range of applications (Chen et al., 2024; Ruan et al., 2024; Isik et al., 2025; Hu et al., 2023; Grattafiori et al., 2024; Li et al., 2025), underscoring their adaptability and relevance in addressing both theoretical and practical challenges in language modeling.

7.2 Scaling Laws for Mixture-of-Experts (MoE)

Mixture-of-Experts (MoE) models (Shazeer et al., 2017; Lepikhin et al., 2020) have emerged as a powerful architecture for language modeling, primarily due to their ability to decouple computational cost from parameter count. Recent research has further explored optimizations within the MoE paradigm. For instance, DeepSeekMoE (Deepseek-AI et al., 2024) investigated the impact of fine-grained expert settings on model performance, proposing a novel design that incorporates shared experts and a hybrid structure combining dense layers with MoE layers. Complementing this, Zoph et al. (2022) highlighted that the performance gains from increased sparsity diminish significantly once the number of experts exceeds 256, suggesting a practical limit for highly sparse models. With the widespread adoption of the MoE architecture, the scaling laws governing MoE models have been extensively studied. Early work by Clark et al. (2022) examined scaling by varying model size and the number of experts on a fixed dataset, concluding that routed models offer efficiency advantages only up to a certain scale. This analysis was subsequently extended by Ludziejewski et al. (2024), who incorporated variable dataset sizes and explored the effects of expert granularity. Additionally, Wang et al. (2024a) investigated the transferability and discrepancies of scaling laws between dense models and MoE models. Abnar et al. (2025) advanced this line of inquiry by deriving scaling laws for optimal sparsity, explicitly considering the interplay between training FLOPs and model size. They also analyzed the relationship between pretraining loss

and downstream task performance, noting distinct behaviors between MoE and dense models on certain tasks. More recently, Ludziejewski et al. (2025) derived joint scaling laws applicable to both dense Transformers and MoE models, demonstrating that MoE architectures can outperform dense counterparts even under constraints of memory usage or total parameter count.

8 Conclusion

In this work, we introduced Efficiency Leverage (EL), a metric that measures the computational advantage of an MoE model relative to a dense counterpart, to quantify the scaling behavior of MoE performance with architectural factors. Our large-scale empirical study, based on over 300 trained models, systematically deconstructed the relationship between MoE design choices and EL. We found that the efficiency of an MoE architecture is governed by a set of predictable principles. Specifically, EL scales as a power law with both the activation ratio and the total compute budget, while expert granularity acts as a non-linear modulator with a stable optimal range. Other factors, such as shared experts, were found to have only a secondary impact. We distilled these insights into a unified scaling law that accurately predicts the EL of any MoE configuration. The predictive power of our framework was empirically validated through the successful design and training of a 17.5B parameter MoE model, which, as predicted, achieved an efficiency leverage of over 7x compared to its dense equivalent.

For future work, our framework can be extended in several key directions: (1) Incorporating memory constraints and communication overhead into the EL framework, particularly for distributed training scenarios where these factors dominate practical efficiency. (2) Developing a unified metric that balances training compute leverage with inference latency requirements, enabling end-to-end efficient architecture co-design. We hope this work inspires continued innovation in MoE architectures, ultimately propelling the field toward greater leverage.

References

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

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle. *CoRR*, abs/2207.14255, 2022. doi: 10.48550/ARXIV.2207.14255. https://doi.org/10.48550/arXiv.2207.14255.

Sumithra Bhakthavatsalam, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, and Peter Clark. Think you have solved direct-answer question answering? try arc-da, the direct-answer AI2 reasoning challenge. *CoRR*, abs/2102.03315, 2021. https://arxiv.org/abs/2102.03315.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6239. https://doi.org/10.1609/aaai.v34i05.6239.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf,

Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. https://arxiv.org/abs/2107.03374.

Yangyi Chen, Binxuan Huang, Yifan Gao, Zhengyang Wang, Jingfeng Yang, and Heng Ji. Scaling laws for predicting downstream performance in llms. *arXiv preprint arXiv:2410.08527*, 2024.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. https://arxiv.org/abs/2110.14168.

DeepSeek-AI. Deepseek-v3 technical report, 2024. https://arxiv.org/abs/2412.19437.

Deepseek-AI, Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

Samir Yitzhak Gadre, Georgios Smyrnis, Vaishaal Shankar, Suchin Gururangan, Mitchell Wortsman, Rulin Shao, Jean Mercat, Alex Fang, Jeffrey Li, Sedrick Keh, et al. Language models scale reliably with over-training and on downstream tasks. *arXiv preprint arXiv:2403.08540*, 2024.

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

Alex Gu, Baptiste Rozière, Hugh James Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida Wang. Cruxeval: A benchmark for code reasoning, understanding and execution. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. https://openreview.net/forum?id=Ffpg52swvg.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021a. https://openreview.net/forum?id=d7KBjmI3GmQ.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021b. https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

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. *arXiv preprint arXiv:2203.15556*, 2022.

Shengding Hu, Xin Liu, Xu Han, Xinrong Zhang, Chaoqun He, Weilin Zhao, Yankai Lin, Ning Ding, Zebin Ou, Guoyang Zeng, et al. Predicting emergent abilities with infinite resolution evaluation. *arXiv preprint arXiv:2310.03262*, 2023.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing*

Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, 2023. http://papers.nips.cc/paper_files/paper/2023/hash/c6ec1844bec96d6d32ae95ae694e23d8-Abstract-Datasets_and_Benchmarks.html.

Berivan Isik, Natalia Ponomareva, Hussein Hazimeh, Dimitris Paparas, Sergei Vassilvitskii, and Sanmi Koyejo. Scaling laws for downstream task performance in machine translation. In *The Thirteenth International Conference on Learning Representations*, 2025.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. https://openreview.net/forum?id=chfJJYC3iL.

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

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

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. RACE: large-scale reading comprehension dataset from examinations. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 785–794. Association for Computational Linguistics, 2017. doi: 10.18653/V1/D17-1082. https://doi.org/10.18653/v1/d17-1082.

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. *arXiv preprint arXiv:2006.16668*, 2020.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. CMMLU: measuring massive multitask language understanding in chinese. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 11260–11285. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.671. https://doi.org/10.18653/v1/2024.findings-acl.671.

Houyi Li, Wenzhen Zheng, Jingcheng Hu, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Shuigeng Zhou, Xiangyu Zhang, et al. Predictable scale: Part i–optimal hyperparameter scaling law in large language model pretraining. *arXiv preprint arXiv:2503.04715*, 2025.

Ling-Team, Binwei Zeng, Chao Huang, Chao Zhang, Changxin Tian, Cong Chen, Dingnan Jin, Feng Yu, Feng Zhu, Feng Yuan, et al. Every flop counts: Scaling a 300b mixture-of-experts ling llm without premium gpus. *arXiv preprint arXiv:2503.05139*, 2025.

Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. Mathbench: Evaluating the theory and application proficiency of llms with a hierarchical mathematics benchmark. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 6884–6915. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.411. https://doi.org/10.18653/v1/2024.findings-acl.411.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. https://openreview.net/forum?id=1qvx610Cu7.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

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

Jan Ludziejewski, Maciej Pióro, Jakub Krajewski, Maciej Stefaniak, Michał Krutul, Jan Małaśnicki, Marek Cygan, Piotr

Sankowski, Kamil Adamczewski, Piotr Miłoś, et al. Joint moe scaling laws: Mixture of experts can be memory efficient. *arXiv preprint arXiv:2502.05172*, 2025.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2381–2391. Association for Computational Linguistics, 2018. doi: 10.18653/V1/D18-1260. https://doi.org/10.18653/v1/d18-1260.

Moonshot-AI. Kimi k2: Open agentic intelligence, 2025. https://moonshotai.github.io/Kimi-K2/.

Nisarg Patel, Mohith Kulkarni, Mihir Parmar, Aashna Budhiraja, Mutsumi Nakamura, Neeraj Varshney, and Chitta Baral. Multi-logieval: Towards evaluating multi-step logical reasoning ability of large language models. *arXiv preprint arXiv:2406.17169*, 2024.

Qiwei Peng, Yekun Chai, and Xuhong Li. Humaneval-xl: A multilingual code generation benchmark for cross-lingual natural language generalization. In Nicoletta Calzolari, Min-Yen Kan, Véronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/COLING 2024, 20-25 May, 2024, Torino, Italy*, pages 8383–8394. ELRA and ICCL, 2024. https://aclanthology.org/2024.lrec-main.735.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022, 2023. doi: 10.48550/ARXIV.2311.12022. https://doi.org/10.48550/arXiv.2311.12022.

Yangjun Ruan, Chris J Maddison, and Tatsunori Hashimoto. Observational scaling laws and the predictability of language model performance. *arXiv preprint arXiv:2405.10938*, 2024.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*, 2023. https://openreview.net/forum?id=qFVVBzXxR2V.

Nikhil Sardana, Jacob Portes, Sasha Doubov, and Jonathan Frankle. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws. *arXiv preprint arXiv:2401.00448*, 2023.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

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. *arXiv preprint arXiv:1701.06538*, 2017.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. https://openreview.net/forum?id=fR3wGCk-IXp.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Xingwu Sun, Yanfeng Chen, Yiqing Huang, Ruobing Xie, Jiaqi Zhu, Kai Zhang, Shuaipeng Li, Zhen Yang, Jonny Han, Xiaobo Shu, et al. Hunyuan-large: An open-source moe model with 52 billion activated parameters by tencent. *arXiv preprint arXiv:2411.02265*, 2024.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13003–13051. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-ACL.824. https://doi.org/10.18653/v1/2023.findings-acl.824.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.

Ning Tao, Anthony Ventresque, Vivek Nallur, and Takfarinas Saber. Enhancing program synthesis with large language models using many-objective grammar-guided genetic programming. *Algorithms*, 17(7):287, 2024. doi: 10.3390/A17070287. https://doi.org/10.3390/a17070287.

Siqi Wang, Zhengyu Chen, Bei Li, Keqing He, Min Zhang, and Jingang Wang. Scaling laws across model architectures: A comparative analysis of dense and moe models in large language models. *arXiv preprint arXiv:2410.05661*, 2024a.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024b. http://papers.nips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract-Datasets_and_Benchmarks_Track.html.

Tianwen Wei, Jian Luan, Wei Liu, Shuang Dong, and Bin Wang. CMATH: can your language model pass chinese elementary school math test? *CoRR*, abs/2306.16636, 2023. doi: 10.48550/ARXIV.2306.16636. https://doi.org/10.48550/arXiv.2306.16636.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1472. https://doi.org/10.18653/v1/p19-1472.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. Evaluating the performance of large language models on GAOKAO benchmark. *CoRR*, abs/2305.12474, 2023. doi: 10.48550/ARXIV.2305.12474. https://doi.org/10.48550/arXiv.2305.12474.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2), 2023.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 2299–2314. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-NAACL.149. https://doi.org/10.18653/v1/2024.findings-naacl.149.

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. *arXiv preprint arXiv:2202.08906*, 2022.

A Notation

To aid readability, we provide a list of key symbols used throughout this paper.

Table 5 Notation.

Symbol	Description
E	Number of routable experts.
E_a	Number of activated experts.
E_s	Number of shared experts.
N	Number of non-vocabulary parameters.
N_a	Number of activated parameters.
d_{model}	Model hidden dimension.
d_{expert}	Expert hidden dimension.
C	Total training compute in FLOPs
M	Compute (w/o embedding) per token in FLOPs.
D	Dataset size in tokens.
A	Activation ratio, *i.e.*, $(E_a + E_s)/(E + E_s)$.
G	Granularity of experts, *i.e.*, $2d_{mode;}/d_{expert}$
S	Shared expert ratio, *i.e.*, $E_s/(E_a + E_s)$

B Experimental Setup

Our experiments primarily follow the architecture and training configurations of the Ling series models (Ling-Team et al., 2025).

Architecture and Tokenizer We adopt a Grouped Query Attention (GQA) (Ainslie et al., 2023) architecture based on the standard decoder-only Transformer, consisting of an embedding layer, multiple alternating layers of attention mechanisms and feed-forward networks, and a final de-embedding layer. Additionally, we use the BPE (Byte-Pair Encoding) algorithm (Sennrich et al., 2015) and RoPE (Rotary Positional Embedding) (Su et al., 2024) to handle positional information. The vocabulary size is 126,464, and the sequence length is 4,096.

Expert Routing Strategy In our MoE layers, a routing network assigns each token's hidden state h_t to the top-N_a experts. This is achieved by generating gating scores $g_t = \text{Softmax}(W_g \cdot h_t)$, where W_g is a learnable matrix. The final output is a weighted sum of the selected experts' outputs: $o_t = \sum_{i \in \text{TopK}(g_t)} g_{t,i} \cdot E_i(h_t)$, where E_i is the i-th expert in total N experts. To ensure balanced expert utilization and stable training, we incorporate two standard auxiliary losses: a load balancing loss (Lepikhin et al., 2020) (coefficient of 0.01) to encourage uniform token distribution, and a router z-loss (Zoph et al., 2022) (coefficient of 0.001) to regularize the magnitude of the gating logits.

Optimizer and Scheduler The parameters of experimental models are initialized from a distribution with a standard deviation of 0.006 and optimized using the AdamW optimizer (Loshchilov and Hutter, 2017). The optimizer's hyperparameters are set to $\beta_1 = 0.9$ and $\beta_2 = 0.95$, with 0.1 weight decay applied. The learning rate schedule employs a WSD (warmup-stable-decay) strategy (Hu et al., 2024): the first 1% of training steps use linear warm-up, followed by exponential decay that reduces the learning rate to 10% of its peak value.

Pre-training Data The training data is sourced from a large-scale multilingual corpus created by the Ling Team, primarily covering English and Chinese, while also including various other languages. This corpus encompasses web text, mathematical materials, programming scripts, published literature, and diverse textual content. To validate model performance, we extracted a 2T-token subset from this corpus for training. In Table 6, we present the composition of the training datasets for all experiments. Unless otherwise specified, this configuration is used throughout.

Table 6 Pre-training data composition.

Type	Web	Books	Wiki	Academic	Code	News	Social	Domain	SFT	Math	Exam
Ratio	46.0%	5.0%	4.0%	6.0%	25.0%	0.1%	1.9%	1.0%	4.0%	6.0%	1.0%

C Estimating FLOPs

To analyze the efficiency of our models, we quantify the computational cost in terms of total training Floating Point Operations (FLOPs). Following standard practice (Kaplan et al., 2020), we estimate the total training FLOPs as approximately three times the cost of a single forward pass ($C_{\text{train}} \approx 3 \cdot C_{\text{fwd}}$). The forward pass FLOPs are the sum of computations from the attention and feed-forward network (FFN) layers, plus a final logit projection.

For a model with hidden size d_{model}, batch size B, and sequence length s, the cost of the attention block per layer, C_{attn}, which includes Grouped-Query Attention (GQA) (Ainslie et al., 2023) and all projections, is approximately:

$$C_{\text{attn}} \approx 2Bsd_{\text{model}}^2 \left(1 + \frac{2}{n_h/n_{kv}}\right) + 4Bs^2 d_{\text{model}} \tag{14}$$

where n_h and n_{kv} are the number of attention and key-value heads, respectively. The FFN cost varies by layer type. A dense layer with intermediate size d_{ffn} requires $C_{\text{dense\_ffn}} = 6Bsd_{\text{model}}d_{\text{ffn}}$ FLOPs. A MoE layer activating E_a experts, each with size d_{expert}, requires:

$$C_{\text{moe\_ffn}} \approx 6Bsd_{\text{model}}(E_a \cdot d_{\text{expert}}) \tag{15}$$

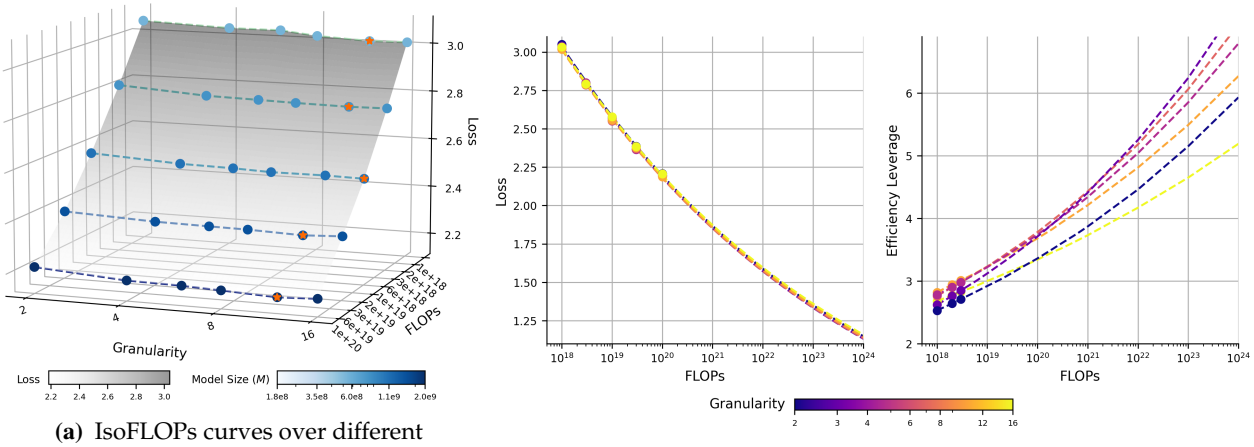
If a shared expert of size d_{shared} is used, its cost, $4Bsd_{\text{model}}d_{\text{shared}}$, is added. For a model with L layers (of which the first L_{dense} are dense) and a vocabulary of size V, the total forward FLOPs are:

$$C_{\text{fwd}} = \sum_{i=1}^{L}(C_{\text{attn}} + C_{\text{ffn},i}) + 2Bsd_{\text{model}}V \tag{16}$$

where $C_{\text{ffn},i}$ is the FFN cost for the i-th layer, which can be either $C_{\text{dense\_ffn}}$ or $C_{\text{moe\_ffn}}$.

D Additional Experiments

The Impact of Routing Balance on the Optimal Expert Granularity. To investigate how routing quality influences the optimal expert granularity, we induce a state of routing imbalance. This is achieved by setting the coefficient of load balancing loss to 0.001, a setup known to cause load imbalance. In this setting, we train MoE models with a varying expert granularity while maintaining a constant total parameter count. As shown in Figure 12, our results reveal that a coarser expert granularity becomes optimal under such imbalanced routing. Specifically, compared with the results in Section 4.1.2, the IsoFLOPs curves (Figure 12a) demonstrate that models with coarser granularity ($G = 6, 8$) achieve lower loss for a given computational budget. This trend is consistently observed in the loss scaling curves (Figure 12b). This phenomenon indicates that when the routing mechanism becomes a performance bottleneck, a fine-grained architecture with numerous specialized experts is counterproductive. The weakened router cannot distribute tokens effectively, nullifying the benefits of specialization. Consequently, the model benefits more from a coarser-grained design with fewer, more generalized experts, as this simplifies the routing task and mitigates the detrimental effects of the load imbalance.



(a) IsoFLOPs curves over different granularity.

(b) Loss scaling curves under over different granularity.

Figure 12 Impact of Expert Granularity on Loss Under Weakened Routing Balance.

Arrangement of MoE and Dense Layers To ensure balanced routing in the early layers, mainstream MoE models typically replace all FFNs except for the first few layers with MoE layers.

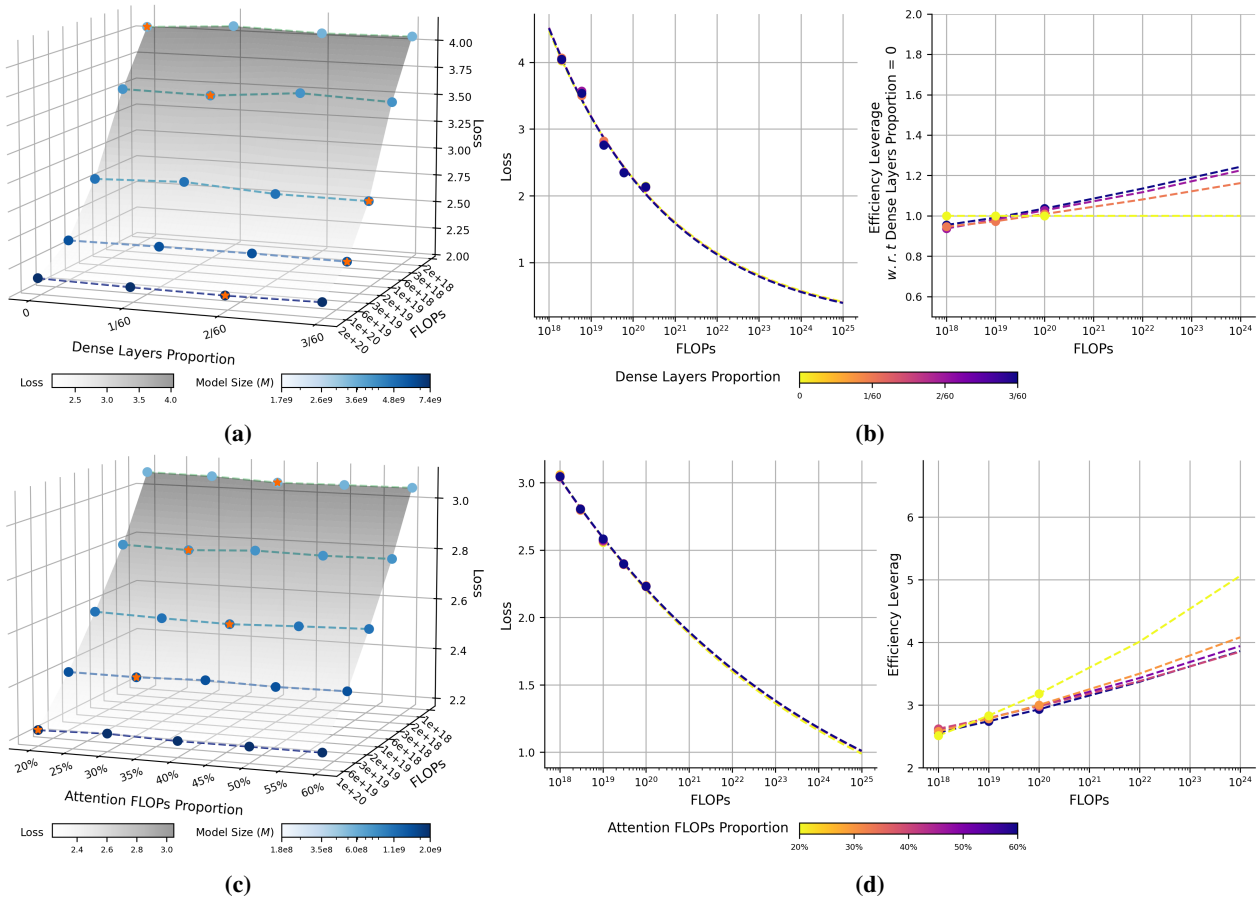


Figure 13 Impact of Dense Layers Proportion and Compute Budget Allocation between Attention and FFN. (a,b) Replacing the first few layers with dense layers shows minor impact on model performance. As computational budgets increase, the optimal proportion of dense layers also gradually rises. (c,d) Modifying the attention FLOPs ratio within a broad range (20%-50%) has a negligible influence on model performance, demonstrating the robustness of this configuration.

We investigate the impact of this design decision on the efficiency of MoE models. To ensure a meaningful exploration space, we extend all models in our experiments to 60 layers and set the first 1, 2, or 3 layers as dense layers sequentially. The dimension of these dense layers is set to match the total dimension of the activated experts in the corresponding MoE layers, ensuring the overall computational cost (FLOPs/token) remains constant. This design allows us to isolate and study the effect of the proportion of dense layers on MoE efficiency. The experimental results, presented in Figure 13a and 13b, reveal the following key findings: 1) From a model performance perspective, replacing the first few layers with dense layers has a minor impact. Using a dense proportion of zero as the baseline, we estimated the efficiency leverage for each configuration. Within a FLOPs budget of up to 1×10^{24} FLOPs, the efficiency leverage remains close to 1. This indicates that configuring the initial layers as dense offers negligible efficiency improvement. However, this adjustment effectively reduces the total number of parameters in the model and mitigates routing imbalances in the early layers. Thus, despite its limited efficiency gains, this remains a valuable design optimization. 2) Further investigation into the optimal proportion of dense layers under varying computational budgets reveals a trend: as FLOPs budgets increase, the optimal dense proportion also grows. For example, in our experiments, when the compute budget is 1×10^{18} FLOPs, the optimal dense proportion is zero. As the compute budget increases to 3×10^{20} FLOPs, the optimal dense layer proportion shifts to approximately $2/60$ or $3/60$.

Compute Resource Allocation between Attention and FFN As two core components of the Transformer model, the attention mechanism (Attention) and FFN account for the majority of the model's computational load. To this end, we explore the impact of computational allocation between the attention mechanism and the FFN on the efficiency of the MoE model. Specifically, we construct a series of models with fixed model scale M but varying compute budgets by increasing the hidden layer size of the attention module while reducing the hidden layer size of each expert in the MoE. We then observe the performance changes of these models under different computational allocations and evaluate their scaling trends. The experimental results are illustrated in Figure 13c and 13d, revealing the following key findings: 1) When the attention FLOPs ratio is between 30% and 40%, it represents a relatively stable and reliable configuration. Models tend to achieve optimal or near-optimal performance within this range. This configuration is consistent with the default settings of mainstream open-source MoE models. 2) Adjusting the attention FLOPs ratio within a broader range (20%-50%) has minor impact on model performance. As shown in Figure 13d, the loss scaling curves and efficiency leverage of these models are nearly identical. Since the attention mechanism generally has a higher computational density (*i.e.*, FLOPs-per-parameter) compared to the FFN, increasing the attention FLOPs ratio while keeping the overall model size constant reduces the total number of model parameters, resulting in higher knowledge density. However, this also implies potentially higher downstream inference costs.

E Additional Evaluation Results of Ling-mini-beta

We present a detailed evaluation of `Ling-mini-beta`'s training process. Figure 14 provides a comprehensive comparison across datasets and categories, as outlined in the main experiments in Section 5.3. The results show that `Ling-mini-beta` achieves comparable performance to `Dense-6.1B` on the majority of datasets.

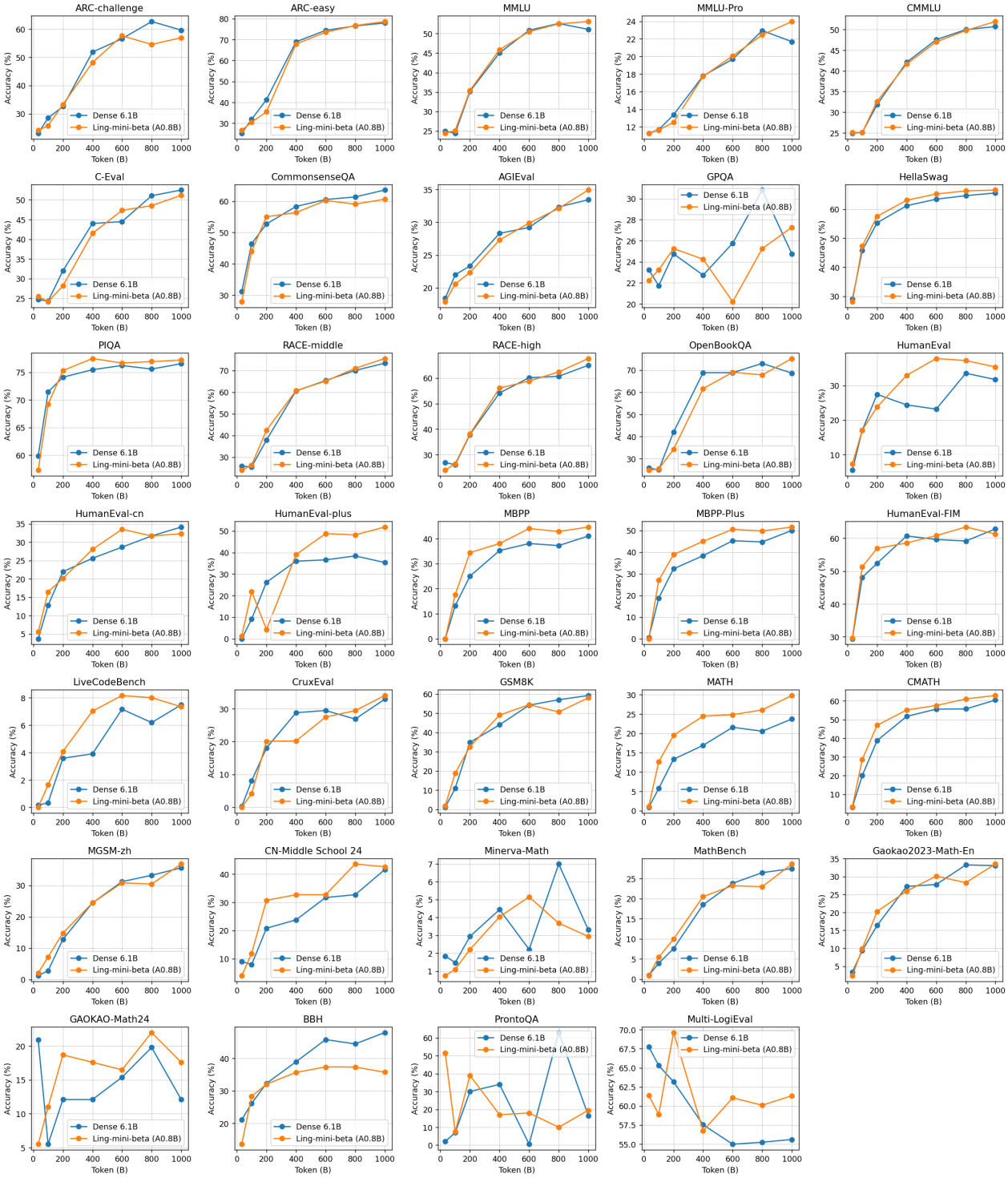


Figure 14 Overall and category-wise performance comparison between `Ling-mini-beta` (17B-A0.8B) and `Dense-6.1B`.

F List of Experimental Models

The detailed configurations for all experiments conducted in this study are presented in Tables 7 (activation ratio), Tables 8 (expert granularity), Tables 9 (shared experts), Tables 10 (layer arrangement), and Tables 11 (compute allocation between attention and FFNs).

Table 7 Experimental configurations for the expert activation ratio analysis. Within each group, the number of activated experts ($E_a = 2$) is fixed, while the total number of experts (E) is varied to study the effect of the activation ratio.

n_{layers}	d_{model}	d_{expert}	n_{heads}	n_{kv_head}	E	E_a	E_s	η	B	Max training FLOPs
8	384	320	8	2	[2,4,8,16,32,64,128,256]	2	1	1.52e-3	98	2e18
8	512	512	8	2	[2,4,8,16,32,64,128,256]	2	1	1.31e-3	147	6e18
10	640	640	10	2	[2,4,8,16,32,64,128,256]	2	1	1.11e-3	228	2e19
14	768	768	12	4	[2,4,8,16,32,64,128,256]	2	1	9.5e-4	342	6e19
16	1024	1024	16	4	[2,4,8,16,32,64,128,256]	2	1	8.1e-4	531	2e20
22	1280	1280	20	4	[2,4,8,16,32,64,128,256]	2	1	7.0e-4	795	6e20

Table 8 Experimental configurations for the expert granularity analysis. Within each group, the base model architecture is fixed while the MoE configuration (total experts E, activated experts E_a, shared experts E_s, and expert dimension d_{expert}) is varied to study the effect of granularity.

n_{layers}	d_{model}	n_{heads}	E	E_a	E_s	d_{expert}	B	η	Max training FLOPs
8	384	8	64	2	1	384	98	1.52e-3	2e18
			128	4	2	192			
			192	6	3	128			
			256	8	4	96			
			384	12	6	64			
			512	16	8	48			
8	512	8	64	2	1	512	147	1.31e-3	6e18
			128	4	2	256			
			192	6	3	170			
			256	8	4	128			
			384	12	6	85			
			512	16	8	64			
10	640	10	64	2	1	640	228	1.11e-3	2e19
			128	4	2	320			
			192	6	3	213			
			256	8	4	160			
			384	12	6	106			
			512	16	8	80			
14	768	12	64	2	1	768	342	9.5e-4	6e19
			128	4	2	384			
			192	6	3	256			
			256	8	4	192			
			384	12	6	128			
			512	16	8	96			
16	1024	16	64	2	1	1024	531	8.1e-4	2e20
			128	4	2	512			
			192	6	3	341			
			256	8	4	256			
			384	12	6	170			
			512	16	8	128			
22	1280	20	64	2	1	1280	795	7.0e-4	6e20
			128	4	2	640			
			192	6	3	426			
			256	8	4	320			
			384	12	6	213			
			512	16	8	160			

Table 9 Experimental configurations for the shared expert ratio analysis. Within each group, we fix the total number of experts ($E = 256$) and the total number of activated pathways ($E_a + E_s = 12$), while varying the ratio between specialized experts (E_a) and shared experts (E_s) to study its impact on performance.

$nlayers$	d_{model}	n_{heads}	E	E_a	E_s	d_{expert}	B	η	Max training FLOPs
8	384	8	256	2	10	96	98	1.52e-3	2e18
			256	4	8	96			
			256	6	6	96			
			256	8	4	96			
			256	11	1	96			
			256	12	0	96			
8	512	8	256	2	10	128	147	1.31e-3	6e18
			256	4	8	128			
			256	6	6	128			
			256	8	4	128			
			256	11	1	128			
			256	12	0	128			
10	640	10	256	2	10	160	228	1.11e-3	2e19
			256	4	8	160			
			256	6	6	160			
			256	8	4	160			
			256	11	1	160			
			256	12	0	160			
14	768	12	256	2	10	192	342	9.5e-4	6e19
			256	4	8	192			
			256	6	6	192			
			256	8	4	192			
			256	11	2	192			
			256	12	0	192			
16	1024	16	256	2	10	256	531	8.1e-4	2e20
			256	4	8	256			
			256	6	6	256			
			256	8	4	256			
			256	11	1	256			
			256	12	0	256			
22	1280	20	256	2	10	320	795	7.0e-4	6e20
			256	4	8	320			
			256	6	6	320			
			256	8	4	320			
			256	11	1	320			
			256	12	0	320			

Table 10 Experimental configurations for the arrangement of MoE and dense layers analysis. Within each group, the total number of layers is fixed at 60, while the mix of dense layers ($n_{\text{dense\_layers}}$) and MoE layers ($n_{\text{moe\_layers}}$) is varied to study the impact of their ratio and placement on performance.

n_{layers}	n_{dense_layers}	n_{moe_layers}	d_{model}	d_{ffn}	n_{heads}	E	E_a	E_s	d_{expert}	B	η	Max training FLOPs
60	0 1 2 3	60 59 58 57	384	1280	8	64	2	1	384	98	1.52e-3	2e18
60	0 1 2 3	60 59 58 57	512	2048	8	64	2	1	512	147	1.31e-3	6e18
60	0 1 2 3	60 59 58 57	640	2560	10	64	2	1	640	228	1.11e-3	2e19
60	0 1 2 3	60 59 58 57	768	3072	12	64	2	1	768	342	9.5e-4	6e19
60	0 1 2 3	60 59 58 57	1024	4096	16	64	2	1	1024	531	8.1e-4	2e20
60	0 1 2 3	60 59 58 57	1280	5120	20	64	2	1	1280	795	7.0e-4	6e20

Table 11 Experimental configurations for analyzing the compute allocation between attention and FFNs. Within each group, the core MoE structure is held constant, while we systematically vary the model's hidden dimension (d_{model}) and the expert dimension (d_{expert}) to explore the optimal trade-off in compute allocation between the attention mechanism and the FFN experts.

layers	d_{model}	d_{expert}	n_{heads}	n_{kv_head}	E	E_s	E_a	η	B	Max training FLOPs
8	352	450	8	2	64	1	2	1.52e-3	96	2e18
8	368	380	8	2	64	1	2	1.52e-3	96	2e18
8	384	320	8	2	64	1	2	1.52e-3	96	2e18
8	400	260	8	2	64	1	2	1.52e-3	96	2e18
8	416	208	8	2	64	1	2	1.52e-3	96	2e18
8	480	626	8	2	64	1	2	1.31e-3	160	6e18
8	512	512	8	2	64	1	2	1.31e-3	160	6e18
8	544	410	8	2	64	1	2	1.31e-3	160	6e18
8	560	364	8	2	64	1	2	1.31e-3	160	6e18
8	576	320	8	2	64	1	2	1.31e-3	160	6e18
10	600	766	10	2	64	1	2	1.11e-3	224	2e19
10	640	640	10	2	64	1	2	1.11e-3	224	2e19
10	680	528	10	2	64	1	2	1.11e-3	224	2e19
10	700	476	10	2	64	1	2	1.11e-3	224	2e19
10	740	380	10	2	64	1	2	1.11e-3	224	2e19
14	696	988	12	4	64	1	2	9.5e-3	320	6e19
14	768	768	12	4	64	1	2	9.5e-3	320	6e19
14	816	642	12	4	64	1	2	9.5e-3	320	6e19
14	840	584	12	4	64	1	2	9.5e-3	320	6e19
14	888	474	12	4	64	1	2	9.5e-3	320	6e19
16	896	1378	16	4	64	1	2	8.1e-3	512	2e20
16	1024	1024	16	4	64	1	2	8.1e-3	512	2e20
16	1088	876	16	4	64	1	2	8.1e-3	512	2e20
16	1152	742	16	4	64	1	2	8.1e-3	512	2e20
16	1184	680	16	4	64	1	2	8.1e-3	512	2e20
22	1120	1686	20	4	64	1	2	7.0e-3	768	6e20
22	1280	1280	20	4	64	1	2	7.0e-3	768	6e20
22	1360	1110	20	4	64	1	2	7.0e-3	768	6e20
22	1440	956	20	4	64	1	2	7.0e-3	768	6e20
22	1520	816	20	4	64	1	2	7.0e-3	768	6e20