INVESTIGATION OF SCALING LAWS FOR ENCODER-DECODER PROTEIN LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

We explore the optimal scaling of encoder-decoder protein language models, a critical yet underexplored architecture for deciphering biological systems. While dense transformer models have achieved remarkable success, guidance on efficiently scaling sparse and asymmetric architectures in the protein domain remains limited. Our investigation is grounded in a systematic study covering over 140 model variants, spanning varying scales (100M to 3.2B total parameters) and compute budgets (8B to 64B tokens). We derive unified scaling laws for both dense and sparse Mixture-of-Experts (MoE) models, providing the first comprehensive roadmap for this design space. First, we demonstrate that MoE models consistently exhibit superior scaling efficiency compared to dense counterparts, offering a more favorable compute-performance frontier. Second, we dissect the behavior of asymmetric encoder-decoder configurations (e.g., 5-of-6 ratios). While these architectures can accelerate convergence, we uncover a critical stability-efficiency trade-off, identifying specific regimes where training instability may offset efficiency gains. Finally, we validate our scaling laws across eight diverse downstream tasks, confirming that our compute-optimal findings translate directly to biological predictive power. Our work provides empirical evidence and practical guidelines for developing next-generation, compute-efficient protein language models.

1 INTRODUCTION

The application of large language models to biological sequences has emerged as a transformative paradigm in computational biology. Protein Language Models (PLMs), such as ESM Lin et al. (2023) and xTrimoPGLM Chen et al. (2024), have demonstrated that scaling model size and training data leads to emergent capabilities in structure prediction, function annotation, and de novo protein design. However, the prevailing approach to scaling has largely focused on increasing the size of dense, symmetric architectures. As the community pushes towards larger models and vast metagenomic datasets, the computational cost of training these dense models is becoming prohibitive. Consequently, the field faces a critical transition: shifting focus from simply "scaling up" to *optimizing the compute-efficient frontier*—identifying architectures that maximize biological understanding within fixed computational budgets.

While this transition is underway in natural language processing (NLP), the optimal scaling strategies for protein sequence modeling remain underexplored, particularly for Encoder-Decoder architectures. This architecture, exemplified by T5 Raffel et al. (2020), is uniquely versatile for both understanding tasks (via encoder embeddings) and generative tasks (via decoder generation). Yet, current protein T5 models rely on symmetric, dense configurations that may be computationally suboptimal. Two architectural innovations offer promising paths to break this efficiency bottleneck but lack systematic validation in the protein domain:

① **Sparse Mixture-of-Experts (MoE):** Biological functions are highly specialized. MoE architectures, which activate sparse subsets of parameters per token, theoretically align well with the modular nature of protein function. However, it remains unknown whether the efficiency gains observed in NLP transfer to the distinct vocabulary and density of biological data.

② **Asymmetric Encoder-Decoder Design:** Downstream biological applications typically rely heavily on encoder representations (e.g., for linear probing or structural prediction), often leaving the decoder underutilized. Redistributing parameter budgets to create "Encoder-Heavy" architectures

could theoretically yield higher returns on investment, yet the stability-efficiency trade-offs of such asymmetric designs have not been characterized for proteins.

In this work, we present the first systematic investigation of scaling laws for sparse and asymmetric encoder-decoder protein language models. We move beyond ad-hoc architectural choices, grounding our investigation in a comprehensive empirical study spanning over 140 model variants. Our experiments cover model scales from 100M to 3.2B total parameters and four distinct compute budgets (8B to 64B tokens), allowing us to map the precise relationship between compute, architecture, and performance. Our core contributions are as follows:

❶ **Unified Scaling Laws for Dense and Sparse Models:** We derive parametric scaling laws that unify dense and MoE architectures under a single framework. We demonstrate that MoE models consistently outperform their dense counterparts, establishing a superior Pareto frontier for protein language modeling. Specifically, MoE models achieve lower asymptotic loss limits, suggesting they are the more compute-efficient choice for future large-scale PLMs.

❷ **Empirical Guidelines for Asymmetric Architectures:** We dissect the behavior of asymmetric (e.g., 5-of-6 encoder-decoder ratio) configurations. While motivated by the encoder-centric nature of downstream biological tasks, we reveal a critical *stability-efficiency trade-off*. We find that while asymmetric models can accelerate convergence at large scales when properly stabilized (e.g., via QK LayerNorm), they introduce optimization instabilities at intermediate scales (e.g., 500M parameters) that can offset their theoretical benefits.

❸ **Downstream Task Validation and Representation Analysis:** We go beyond pre-training metrics to validate our findings on eight diverse biological tasks, ranging from enzyme activity prediction to fold classification. We observe a strong correlation between our pre-training compute-optimality and downstream performance, confirming that our scaling laws serve as a reliable proxy for biological predictive power. Notably, we identify distinct performance profiles where MoE models match dense performance on binary classification while underperforming on regression and fine-grained classification tasks. To explain this gap, we analyze the geometry of encoder embeddings and find consistent signs of representation collapse in MoE models (e.g., increased anisotropy and reduced intrinsic dimensionality). Finally, building on prior MoE-embedding ideas, we show that augmenting encoder embeddings with simple routing-weight features (MoEE) can substantially mitigate this degradation on the harder downstream tasks without additional pre-training.

By establishing these scaling laws and architectural guidelines, we provide a rigorous roadmap for developing the next generation of compute-efficient protein language models.

2 Related work

Mixture-of-Experts Architectures. Sparse Mixture-of-Experts (MoE) architectures have emerged as an effective approach to scale language models efficiently. GShard Lepikhin et al. (2020) and Switch Transformer Fedus et al. (2022) pioneered activating a subset of model parameters per input, enabling significant capacity increases without proportional computational cost. Krajewski et al. (2024) analyzed scaling laws for fine-grained MoE models, showing that expert granularity plays a critical role in efficiency-accuracy tradeoffs. However, these studies focus exclusively on decoder-only architectures for natural language, leaving encoder-decoder MoE models and biological domains unexplored.

Protein Language Models. Protein language models such as ESM-2 Lin et al. (2023), ProtT5 Elnaggar et al. (2021), and xTrimoPGLM Chen et al. (2024) have demonstrated the utility of self-supervised learning at scale, achieving success in structure prediction, contact mapping, and function annotation. However, their dense nature poses scalability challenges as biological sequence databases continue to grow.

Asymmetric Encoder-Decoder Architectures. Recent work has demonstrated that asymmetric encoder-decoder configurations can offer significant advantages. T5Gemma Zhang et al. (2025a) showed that pairing a large encoder (9B) with a small decoder (2B) achieves over 3% accuracy improvement compared to balanced configurations while maintaining similar generation latency to smaller models. This approach is particularly effective for tasks where deep input understanding is more critical than output complexity. For protein language modeling, where downstream applications

typically rely on encoder representations via linear probing, concentrating parameters in the encoder may yield higher returns.

Scaling Laws. Kaplan et al. (2020) established scaling laws for dense models, while Clark et al. (2022) provided unified scaling laws for routed networks. Our work addresses a gap by providing the first systematic study of scaling laws for both dense and MoE encoder-decoder models in the protein domain.

3 PRETRAINING PIPELINE

Model Architecture. We adopt the encoder-decoder Transformer based on the T5 architecture Raffel et al. (2020) as our base model. To systematically study the effect of scale, we trained four model sizes with approximately 100M, 250M, 500M, and 1B active parameters, respectively. For MoE variants, the feed-forward network (FFN) layers in both the encoder and decoder are replaced with sparsely activated Mixture-of-Experts (MoE) layers. For each model size, we further trained variants with 8, 16, 32, and 64 experts, activating one-quarter of the experts (denoted as k) per token.

The MoE module replaces the standard FFN. For a given input token representation x, the output is a weighted sum of the outputs of the top-k selected experts, as shown in Equation (1):

$$\texttt{MoE}(x) = \sum_{i \in \texttt{TopK}(r(x))} \texttt{softmax}(r(x))_i \cdot E_i(x) \tag{1}$$

where r, called the *router*, is a learned linear layer that computes logits for all N_E experts. The softmax function converts these logits into routing probabilities. Each selected expert E_i is a standard FFN that processes the input x. The outputs of the selected Top-k experts are then weighted by their respective routing probabilities and summed.

Key decisions in designing an MoE model include determining the number of active and total parameters, the design of the experts (e.g., granularity, sharing), and the choice of the routing algorithm and its associated losses. Experiments related to these design choices are detailed in Section 6.1; Table 1 shows our final configuration for the MoE models in this study.

Table 1: Key MoE design choices and our final setup. The symbol column aligns with our loss formulation in Equation (2).

Sym.	Design Choice	Description	Final Decision
N_{act}	Active params	# active params per token	100M, 250M, 500M, 1B
N_{total}	Total params	Total # of params	140M, 600M, 1.6B, 3.2B
G	Granularity	# experts per layer	8, 16, 32, 64
-	Sharing	Shared expert	Enabled
-	Routing	Token assignment	Dropless token choice
$\mathcal{L}_{LB}$	Load balance	Aux loss for equal assign.	Not used
b_e	Expert bias	Learnable router bias	0.001 with sign
$\mathcal{L}_{RZ}$	Router z-loss	Aux loss for large logits	$\beta = 0.001$

In summary, our MoE architecture utilizes fine-grained experts, with a configuration of 32 experts per layer and 4 activated experts ($k = 4$) yielding the best performance-to-cost trade-off. We use a dropless token choice router and train all models from scratch. For load balancing, instead of a traditional auxiliary loss, we employ a learnable expert bias (Wang et al., 2024) added to the router logits, which is periodically updated to encourage balanced expert utilization. To maintain training stability, we incorporate a router z-loss (Zoph et al., 2022). Our final training loss $\mathcal{L}$ is a linear combination of the cross-entropy loss ($\mathcal{L}_{CE}$) and the router z-loss ($\mathcal{L}_{RZ}$):

$$\mathcal{L} = \mathcal{L}_{CE} + \beta \mathcal{L}_{RZ} \tag{2}$$

where β is the loss weight for the router z-loss, as defined in Table 1.

Pretraining Data. We utilize the Uniref dataset (Suzek et al., 2007), which originally comprises 412M protein sequences. The dataset underwent deduplication and filtering to exclude sequences longer than $1,000$ amino acids. We implemented an efficient pre-processing and offline packing

algorithm that uses a sliding window approach to maximize space utilization while adhering to the $1,024$-length constraint. This optimization resulted in a consolidated dataset of 137M rows, with sequences separated by <EOS> tokens, achieving over 98% packing efficiency and a total of 137B sequence tokens. To address potential information leakage between sequences within the same row, we utilized a variable-length attention mechanism in our modeling approach (Lu et al., 2024).

4 Unified Scaling Law for Dense and Sparse Models

In this section, we address the core question: how do the scaling laws differ between MoE variants and dense versions of an encoder-decoder Transformer model for proteins, and how does expert granularity influence these properties? To this end, we aim to derive a parametric scaling law for predicting the final loss value $\mathcal{L}$ based on the expert granularity G, the number of non-embedding active parameters N_{act}, and the number of training tokens D.

The concept of granularity, inspired by Krajewski et al. (2024), allows for fine-grained control over the size and number of experts. We use the term *active parameters*, denoted as N_{act}, to refer to the non-embedding parameters used to produce the output for a single token, excluding the router itself. Granularity G is then defined as the ratio of the standard feed-forward hidden dimension d_{ff} to the hidden dimension of a single expert, d_{expert}:

$$G = \frac{d_{\text{ff}}}{d_{\text{expert}}} \tag{3}$$

In other words, granularity denotes the multiplier for the change in the number of experts from a standard model where $G = 1$. A higher granularity ($G > 1$) results in a larger number of smaller experts.

To derive the scaling laws, we conducted over 80 experiments on our encoder-decoder Transformer architecture, where each feed-forward component was replaced by a MoE layer. These experiments covered four model sizes based on their active parameters, $N_{\text{act}} \in \{100\text{M}, 250\text{M}, 500\text{M}, 1\text{B}\}$, and five architectural variants: a dense baseline and four MoE configurations with varying granularities, $G \in \{2, 4, 8, 16\}$. Each of these variants was trained on four different data scales, $D \in \{8\text{B}, 16\text{B}, 32\text{B}, 64\text{B}\}$ tokens. For each token count, the learning rate decay schedule was adjusted accordingly to ensure compute-optimal training, following the principles of Hoffmann et al. (2022). The full details of all architectures, the training procedure, and hyperparameter choices are described in Appendix 8.

4.1 Parametric Form of the Scaling Law

For both dense and MoE models, we adopt the functional form of the standard Chinchilla scaling law Hoffmann et al. (2022), which models the final loss $\mathcal{L}$ as a function of the number of non-embedding active parameters N_{act} and the number of training tokens D:

$$\mathcal{L}(N_{\text{act}}, D) = c + \frac{a}{N_{\text{act}}^{\alpha}} + \frac{b}{D^{\beta}} \tag{4}$$

While prior work such as Krajewski et al. (2024) has proposed a more complex form incorporating expert granularity, our preliminary experiments revealed weak dependence on granularity beyond the `Top-4 of 16` configuration (see Figure 8 in Appendix 13.0.1). Consequently, we omit G from the final parametric fit and instead model MoE and Dense as separate model families, each following the standard Chinchilla form.

Following Hoffmann et al. (2022) and Krajewski et al. (2024), we fit the parameters of these laws by minimizing the Huber loss with $\delta = 0.1$. The optimization is performed using the BFGS algorithm with a weight decay of 5×10^{-4}. The resulting fitted coefficients for both MoE and dense models are presented in Table 2. Figure 1a presents the scaling law curves for all four architectural variants: symmetric Dense, symmetric MoE, asymmetric (5-of-6) Dense, and asymmetric MoE. Per-variant detailed plots are provided in Appendix 9.

A comparison of the fitted coefficients reveals key differences in the scaling behavior of MoE and dense models. The MoE models exhibit a slightly larger exponent for both model size ($\alpha = 0.300$ vs.

Table 2: Values of the fitted coefficients for the scaling laws of MoE and Dense models.

Model	a	α	b	β	c
MoE	187.0	0.300	11273	0.414	0.445
Dense	173.5	0.295	10155	0.410	0.534

0.295) and data size ($\beta = 0.414$ vs. 0.410), suggesting that they benefit more from scaling up both parameters and data. Furthermore, the irreducible loss term c is substantially lower for MoE models (0.445 vs. 0.534), indicating a lower asymptotic loss limit. This implies that with sufficient scale, MoE models have the potential to achieve better ultimate performance than their dense counterparts in the protein domain.

5 Asymmetric Encoder-Decoder Architectures

In this section, we present our empirical study of asymmetric encoder-decoder architectures for protein language models. We examine how allocating different proportions of parameters to the encoder and decoder affects model performance and training efficiency.

5.1 Motivation for Asymmetric Architectures

The conventional encoder-decoder architecture, as exemplified by the original Transformer and T5 models, typically employs a roughly symmetric design where the encoder and decoder have similar capacities. However, recent work has demonstrated that asymmetric configurations can offer significant advantages for certain applications.

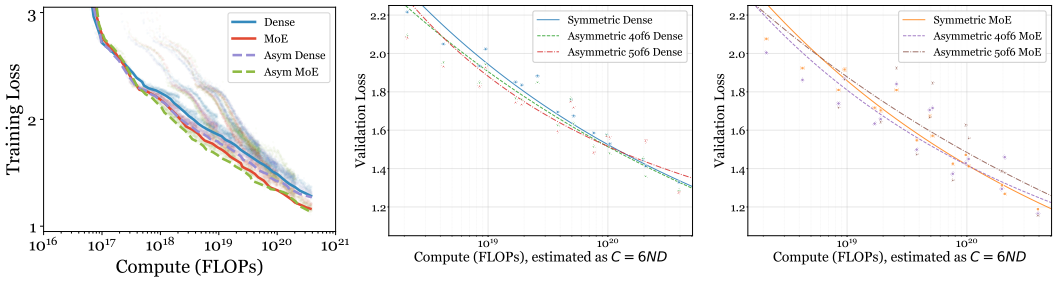
Evidence from Natural Language Processing. Recent work on T5Gemma Zhang et al. (2025a) has demonstrated the effectiveness of highly asymmetric encoder-decoder configurations. By adapting pre-trained decoder-only Gemma 2 models, they construct encoder-decoder models with different parameter budgets for the encoder and decoder. Their 9B-2B model (9 billion parameter encoder paired with 2 billion parameter decoder, totaling 10.4B parameters) achieves over 3% accuracy improvement compared to the balanced 2B-2B configuration while maintaining similar generation latency to the smaller Gemma 2 2B model. This suggests that for tasks where deep understanding of the input is more critical than the complexity of the generated output, concentrating parameters in the encoder can be highly beneficial.

Relevance to Protein Language Modeling. A key observation motivating our investigation is that many downstream applications in protein language modeling rely primarily on encoder representations. Specifically, our evaluation methodology employs linear probing on encoder embeddings for sequence-level classification and regression tasks. In this setting, the decoder parameters are not directly utilized for downstream predictions, suggesting that allocating more capacity to the encoder could yield higher returns for these applications.

Based on these motivations, we conducted extensive experiments with asymmetric architectures. While T5Gemma achieves asymmetry by pairing pre-trained models of different parameter sizes, we explore a complementary approach: allocating different proportions of transformer layers to the encoder and decoder within a fixed total layer budget. Specifically, we examine **4-of-6** configurations (where the encoder comprises 4 out of 6 total transformer layers, i.e., 4 encoder layers and 2 decoder layers) and **5-of-6** configurations (where the encoder comprises 5 out of 6 total layers, i.e., 5 encoder layers and 1 decoder layer). In contrast, symmetric models allocate layers equally between encoder and decoder (3 encoder layers and 3 decoder layers for a 6-layer model).

5.2 Training Stability Challenges

Our experiments revealed that asymmetric architectures introduce significant training stability challenges. Without QK LayerNorm (as discussed in Section 13.1), we observed convergence failures across multiple configurations: ❶ **5-of-6 models:** Both 100M and 250M scale models exhibited convergence issues, with training loss diverging or failing to decrease at expected rates. ❷ **4-of-6**



(a) All variants (b) Dense by ratio (c) MoE by ratio

Figure 1: Scaling law comparison. (a) All architectural variants: MoE models (blue, purple) achieve lower loss than Dense (red, green) at equivalent compute. (b-c) Dense and MoE models by encoder-decoder ratio. Asymmetric variants show variable performance with no single configuration consistently dominating.

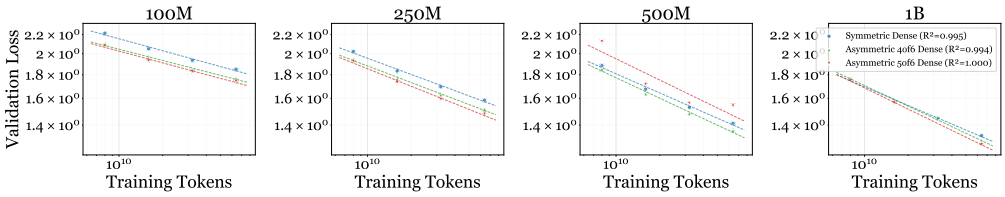


Figure 2: Dense model performance grouped by model size (100M, 250M, 500M, 1B). The 500M scale shows notably weaker performance for asymmetric variants, which contributes to their less favorable overall scaling law fit.

models: The 100M scale model showed similar convergence problems, though larger scales were more stable.

These stability issues appear to stem from the imbalanced gradient flow between the larger encoder and smaller decoder. When the encoder is significantly larger, the decoder may receive insufficient gradient signal during training, leading to optimization difficulties.

After enabling QK LayerNorm to stabilize training, we observed a different pattern in our experiments at the 1B scale with 64B tokens: the convergence speed followed the ordering **5-of-6** $>$ **4-of-6** $>$ **symmetric** for both dense and MoE architectures. This suggests that when training is properly stabilized, asymmetric architectures can indeed accelerate convergence.

5.3 Scaling Law Analysis for Asymmetric Architectures

To understand the broader implications of asymmetric architectures, we fitted scaling laws for all six model configurations: dense and MoE variants of symmetric, 4-of-6, and 5-of-6 architectures. Figure 1 shows the scaling law curves for both dense and MoE models.

A critical observation from these scaling law curves is that **asymmetric variants do not consistently outperform symmetric configurations**. While asymmetric models may show advantages in specific compute regimes or for particular model sizes, the symmetric baseline remains competitive across the full range of our experiments.

5.4 Analysis by Model Size

To better understand the source of this variability, we analyzed the scaling behavior separately for each model size. Figure 2 shows the loss versus training tokens for dense models grouped by model size.

This analysis reveals that the inconsistent performance of asymmetric architectures is primarily driven by the **500M model scale**, where asymmetric variants show notably weaker performance compared

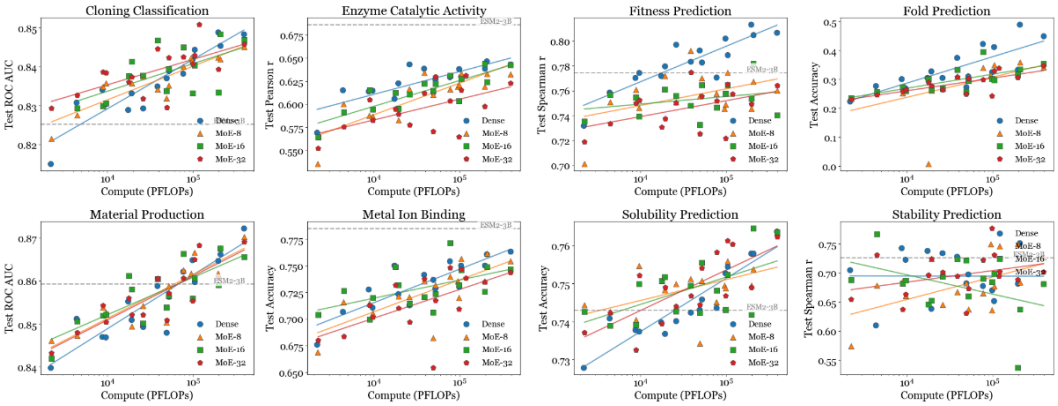


Figure 3: Downstream task performance as a function of pre-training compute across eight diverse tasks. Each point represents a model checkpoint, with different colors and markers distinguishing Dense and MoE variants with symmetric (3-of-6), symmetric (4-of-6 and 5-of-6) encoder-decoder configurations. Seven out of eight tasks exhibit clear scaling behavior, with performance improving as pre-training compute increases. Notably, dense models outperform MoE models on the three non-binary tasks (Enzyme Catalytic Activity, Fitness Prediction, and Fold Prediction), while MoE models achieve comparable or slightly superior performance on the four binary classification tasks.

to the symmetric baseline. At other scales (100M, 250M, and 1B), the asymmetric configurations perform more comparably to or even slightly better than symmetric models. The poor performance at the 500M scale significantly impacts the overall scaling law fit, leading to the observed variability in the aggregate curves.

We hypothesize that the 500M scale may represent a challenging intermediate regime where the asymmetric parameter allocation creates optimization difficulties that are not present at smaller scales (where overall model capacity is limited) or larger scales (where both encoder and decoder have sufficient capacity despite the imbalanced allocation).

6 Downstream Task Evaluation

Downstream Tasks. We evaluate all model variants on eight diverse downstream tasks spanning protein structure, function, and development: Enzyme Catalytic Activity, GB1 Fitness Prediction, Fold Classification, Metal Ion Binding, Solubility Prediction, Stability Prediction, Cloning Classification, and Material Production. These tasks include both regression and classification challenges. Detailed task descriptions are provided in Appendix 10.1.

Finetuning Methodology. We evaluate all models by probing on encoder embeddings. For most tasks, we use a 2-layer MLP with hidden dimension 128; for Fold Prediction, we use single-layer linear probing due to the large number of classes (1,195). We follow the official data splits and evaluation metrics from prior work Chen et al. (2024). Details are provided in Appendix 10.2.

Results on the downstream applications. Figure 3 presents the relationship between pre-training compute (measured in PFLOPs) and downstream task performance across all eight evaluation tasks. Our analysis reveals two key findings regarding scalability and architectural trade-offs.

Scaling behavior. Seven out of eight tasks demonstrate clear positive scaling trends, where downstream performance improves consistently with increased pre-training compute. This strong correlation between compute investment and task performance validates our scaling approach and confirms that the representational improvements achieved through larger-scale pre-training effectively transfer to biologically relevant applications. The sole exception is Stability Prediction, which exhibits high variance and no discernible scaling pattern, suggesting that this task may require specialized architectural modifications or training strategies beyond simple scale increases.

Dense vs. MoE performance trade-offs. A nuanced pattern emerges when comparing dense and MoE architectures across different task types. For the three non-binary tasks (Enzyme Catalytic

Activity, Fitness Prediction, and Fold Prediction), dense models consistently outperform their MoE counterparts at equivalent compute budgets. This suggests that dense architectures may be more effective at capturing the fine-grained representations required for predicting continuous values or distinguishing among many classes. In contrast, for the four binary classification tasks (Cloning Classification, Material Production, Metal Ion Binding, and Solubility Prediction), MoE and dense models achieve comparable performance, with MoE models occasionally demonstrating slight advantages. This pattern indicates that the computational efficiency gains of MoE architectures can be realized without sacrificing performance on binary discrimination tasks, while more complex prediction scenarios may benefit from the denser parameter utilization of traditional architectures.

We further investigate why MoE models underperform on regression tasks through representation collapse analysis and propose MoEE (routing weight augmentation) to mitigate this issue. These analyses are detailed in Appendix 12.1.

6.1 Expert Granularity

Expert granularity refers to the trade-off between the number and size of experts in an MoE layer. Instead of using a few large experts, one can employ a larger number of smaller, "fine-grained" experts while keeping the total active parameters and computational cost constant. Recent studies Dai et al. (2024); Muennighoff et al. (2024); Krajewski et al. (2024) have highlighted the benefits of fine-grained experts, demonstrating that increased combinatorial flexibility enhances knowledge acquisition.

In our experiments, we observed that increasing granularity from a `Top-2 of 8` to a `Top-4 of 16` configuration led to a noticeable improvement in convergence speed. However, further refining the experts to a `Top-8 of 32` setup did not yield significant additional gains, while higher granularity resulted in decreased token throughput due to communication overhead Zhang et al. (2025b); Luo et al. (2025). Therefore, we identified the `Top-4 of 16` configuration as the optimal sweet spot. Detailed training curves are provided in Appendix 13.0.1.

Beyond expert granularity, we conducted extensive ablation studies on additional MoE design choices and general pretraining settings, including the use of shared experts, load balancing strategies (auxiliary loss vs. expert bias), router z-loss, QK LayerNorm, and embedding gradient shrinking. These experiments informed our final architecture decisions summarized in Table 1. Due to space constraints, detailed descriptions and experimental results for these ablations are provided in Appendix 13. A detailed comparison of MoE vs. Dense training efficiency is also provided in Appendix 11.1.

7 Conclusion

We present a systematic investigation of scaling laws for encoder-decoder protein language models, examining both Mixture-of-Experts (MoE) and asymmetric encoder-decoder configurations. Through extensive experiments spanning over 140 model variants, we derive unified scaling laws and provide detailed analyses of the trade-offs involved in each architectural choice.

Our investigation yields several key observations. First, MoE models consistently demonstrate superior scaling efficiency compared to dense counterparts, achieving faster convergence and lower asymptotic loss limits. This confirms that the benefits of sparse architectures observed in natural language processing transfer effectively to the protein domain.

Second, our analysis of asymmetric encoder-decoder architectures reveals a more nuanced picture. While asymmetric configurations can accelerate convergence when properly stabilized (with techniques such as QK LayerNorm), they introduce training instabilities that limit their practical benefits. Notably, asymmetric variants do not consistently outperform symmetric baselines across all model sizes and compute budgets, with particularly challenging behavior observed at the 500M scale.

These observations provide empirical insights for practitioners: MoE architectures demonstrate clear scaling advantages over dense models for protein language modeling, while asymmetric configurations require careful consideration of the stability-performance trade-offs. Future work should investigate methods to stabilize asymmetric training more reliably and explore whether the potential advantages of encoder-heavy architectures can be more consistently realized in practice.

REFERENCES

Bo Chen, Xingyi Cheng, Pan Li, Yangli-ao Geng, Jing Gong, Shen Li, Zhilei Bei, Xu Tan, Boyan Wang, Xin Zeng, et al. xtrimopglm: unified 100b-scale pre-trained transformer for deciphering the language of protein. *arXiv preprint arXiv:2401.06199*, 2024.

Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, Xian-Ling Mao, et al. On the representation collapse of sparse mixture of experts. *Advances in Neural Information Processing Systems*, 35:34600–34613, 2022.

Aidan Clark, Diego de las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, George van den Driessche, Eliza Rutherford, Tom Hennigan, Matthew Johnson, Katie Millican, Albin Cassirer, Chris Jones, Elena Buchatskaya, David Budden, Laurent Sifre, Simon Osindero, Oriol Vinyals, Jack Rae, Erich Elsen, Koray Kavukcuoglu, and Karen Simonyan. Unified scaling laws for routed language models, 2022. URL https://arxiv.org/abs/2202.01169.

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.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International conference on machine learning*, pp. 7480–7512. PMLR, 2023.

Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, et al. Prottrans: Towards cracking the language of life's code through self-supervised deep learning and high performance computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*, 2019.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. In *International Conference on Learning Representations*, 2022.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Raghavi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. Olmo: Accelerating the science of language models, 2024.

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.

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.

Sameer Khurana, Reda Rawi, Khalid Kunji, Gwo-Yu Chuang, Halima Bensmail, and Raghvendra Mall. Deepsol: a deep learning framework for sequence-based protein solubility prediction. *Bioinformatics*, 34(15):2605–2613, 2018.

Jakub Krajewski, Jan Ludziejewski, 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. *arXiv preprint arXiv:2402.07871*, 2024.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Huang, Yanping Chen, Orhan Firat, Zhifeng Chen, Maxim Krikun, Yanqi Cao, Yicheng Wang, et al. Gshard: Scaling giant models with conditional computation and automatic sharding. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Ziyue Li and Tianyi Zhou. Your mixture-of-experts llm is secretly an embedding model for free. *arXiv preprint arXiv:2410.10814*, 2024.

Zeming Lin, Hanieh Akin, Roshan Rao, Brian Hie, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 2023.

Jiasen Lu, Christopher Clark, Sangho Lee, Zichen Zhang, Savya Khosla, Ryan Marten, Derek Hoiem, and Aniruddha Kembhavi. Unified-io 2: Scaling autoregressive multimodal models with vision language audio and action. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26439–26455, 2024.

Shuqing Luo, Pingzhi Li, Jie Peng, Hanrui Wang, Yu Cheng, Tianlong Chen, et al. Occult: Optimizing collaborative communication across experts for accelerated parallel moe training and inference. *arXiv preprint arXiv:2505.13345*, 2025.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Peter Chen, John Canny, Pieter Abbeel, and Yun Song. Evaluating protein transfer learning with tape. *Advances in neural information processing systems*, 32, 2019.

Gabriel J Rocklin, Tamuka M Chidyausiku, Inna Goreshnik, Alex Ford, Scott Houliston, Alexander Lemak, Lauren Carter, Rashmi Ravichandran, Vikram K Mulligan, Aaron Chevalier, et al. Global analysis of protein folding using massively parallel design, synthesis, and testing. *Science*, 357 (6347):168–175, 2017.

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

Baris E Suzek, Hongzhan Huang, Peter McGarvey, Raja Mazumder, and Cathy H Wu. Uniref: comprehensive and non-redundant uniprot reference clusters. *Bioinformatics*, 23(10):1282–1288, 2007.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*, 2022.

Biao Zhang, Fedor Moiseev, Joshua Ainslie, Paul Suganthan, Min Ma, Surya Bhupatiraju, Fede Lebron, Orhan Firat, Armand Joulin, and Zhe Dong. Encoder-decoder gemma: Improving the quality-efficiency trade-off via adaptation. *arXiv preprint arXiv:2504.06225*, 2025a.

Mohan Zhang, Pingzhi Li, Jie Peng, Mufan Qiu, and Tianlong Chen. Advancing moe efficiency: A collaboration-constrained routing (c2r) strategy for better expert parallelism design. *arXiv preprint arXiv:2504.01337*, 2025b.

Zuobai Zhang, Chuanrui Wang, Minghao Xu, Vijil Chenthamarakshan, Aurélie Lozano, Payel Das, and Jian Tang. A systematic study of joint representation learning on protein sequences and structures. *arXiv preprint arXiv:2303.06275*, 2023.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models, 2022.

8 TRAINING CONFIGURATION

This appendix provides the detailed training configuration for all model variants discussed in the main text. All models are trained using a modified Megatron-LM framework on NVIDIA H200 GPUs.

8.1 MODEL ARCHITECTURE AND TRAINING CONFIGURATION

Table 3 summarizes the architectural configuration, MoE-specific settings, and training hyperparameters.

Table 3: Model configuration for 1B-scale models: (a) base architecture, (b) MoE-specific settings, (c) training hyperparameters.

(a) Architecture

Parameter	Value
Encoder layers	12
Decoder layers	12
Hidden size	1536
Attention heads	12
KV channels	128
FFN hidden size	6144
Sequence length	1024
Vocabulary size	255
Position embed.	RoPE
Normalization	RMSNorm
Activation	SwiGLU

(b) MoE Settings

Parameter	Value
Num. experts	16
Top-K routing	3
Expert MLP ratio	0.25
Shared expert ratio	0.25
Router score fn	Sigmoid
Expert bias	Enabled
Router Z-loss	0.001
Aux loss coef.	0.0
Token dispatcher	AllToAll
Grouped GEMM	Enabled

(c) Training

Param.	Dense	MoE
Peak LR	1e-4	1e-4
Min LR	1e-5	1e-5
LR decay	Cosine	Cosine
Adam β_1	0.9	0.9
Adam β_2	0.95	0.95
Weight decay	0.1	0.1
Grad clip	1.0	1.0
Precision	BF16	BF16

8.2 STABILITY TECHNIQUES

We employ several techniques to ensure training stability:

- **QK LayerNorm:** Applied to both self-attention and cross-attention layers (see Appendix 13.0.2).
- **Embedding Gradient Shrinking:** Factor of 0.1 to prevent gradient spikes from the embedding layer.
- **Z-loss:** Coefficient of 0.001 to regularize large logits in attention and routing computations.
- **Sequence Parallelism:** Enabled to reduce memory footprint and improve throughput.

8.3 PARALLELISM STRATEGY

For Dense models, we use tensor parallelism (TP=2) across GPUs. For MoE models, we use expert parallelism (EP=8) with tensor parallelism disabled (TP=1) to efficiently distribute experts across devices.

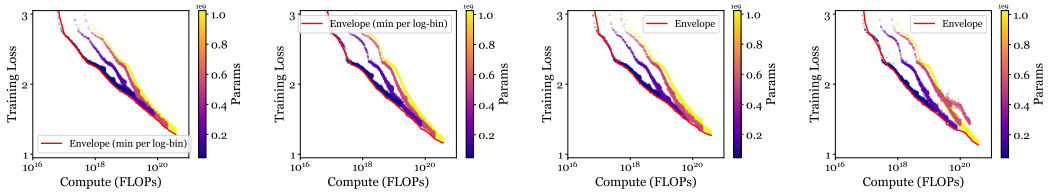
9 SCALING LAW DETAILS

Figure 4 presents the individual scaling law fits for each of the four architectural variants: symmetric Dense, symmetric MoE, asymmetric (5-of-6) Dense, and asymmetric (5-of-6) MoE. Across all variants, MoE models consistently achieve lower loss than their Dense counterparts at equivalent compute budgets.

10 EVALUATION SETUP

10.1 DOWNSTREAM TASK DESCRIPTIONS

We evaluate our models on eight diverse downstream tasks:



(a) Symmetric Dense (b) Symmetric MoE (c) Asymmetric Dense (d) Asymmetric MoE

Figure 4: Per-variant scaling law fits. Each panel shows the Chinchilla scaling law fit for a specific architectural configuration: (a) symmetric Dense, (b) symmetric MoE, (c) asymmetric (5-of-6) Dense, and (d) asymmetric (5-of-6) MoE.

① **Enzyme Catalytic Activity:** A regression task to predict the enzymatic turnover number (k_{cat}), which denotes the maximum chemical conversion rate of a metabolic enzyme.

② **GB1 Fitness Prediction:** A regression task to predict the fitness of the GB1 binding protein following mutations, measuring how mutations affect protein function.

③ **Fold Classification:** A classification task that aims to classify a protein sequence into one of $1,195$ known structural folds based on the SCOP database.

④ **Metal Ion Binding:** A classification task to predict whether a protein has binding sites for metal ions, which is crucial for a wide range of biological processes including catalysis and regulation.

⑤ **Solubility Prediction:** A binary classification task to assess whether a protein is soluble or insoluble. Protein solubility is a crucial design parameter for ensuring protein efficacy, particularly in the pharmaceutical domain. The dataset follows the DeepSol Khurana et al. (2018) methodology, where proteins with sequence identity of 30% or greater to any protein in the test set are removed from training and validation sets.

⑥ **Stability Prediction:** A regression task to predict the concentration of protease at which a protein can retain its folded state. Understanding protein stability during protease interaction is valuable for therapeutic development. The dataset originates from Rocklin et al. Rocklin et al. (2017) and was collected within the TAPE benchmark Rao et al. (2019). We use the Spearman Correlation Coefficient (SRCC) as the evaluation metric.

⑦ **Cloning Classification:** A classification task to predict whether a protein is stable at a certain experimental stage during protein structure determination. Protein structure determination includes a series of experimental stages to yield stable proteins for X-ray crystallography, and each step corresponds to a "stage tag" indicating stability.

⑧ **Material Production:** A binary classification task to predict whether a protein sequence fails at the protein material stage during protein structure determination experiments.

10.2 Finetuning Methodology

We evaluate all models using probing on encoder embeddings. For sequence-level classification and regression tasks, we extract the embedding by average pooling of the final layer of the encoder's output for each input protein sequence.

For most tasks (Enzyme Catalytic Activity, GB1 Fitness Prediction, Metal Ion Binding, Solubility Prediction, Stability Prediction, Cloning Classification, and Material Production), we use a 2-layer MLP prediction head with a hidden size of 128. For Fold Prediction, which involves classification over 1,195 classes, we use a single-layer linear classifier to avoid overfitting. Our current evaluation does not include token-level or multi-sequence interaction tasks.

We adhere to the official training, validation, and test splits and evaluation metrics from prior work Chen et al. (2024). For tasks lacking a predefined validation set, we randomly hold out 10% of the training data for validation purposes. The prediction head is trained using the Adam optimizer with a peak learning rate of 1e-4.

11 Additional Experiments

11.1 Mixture-of-Experts vs. Dense

Following prior work in natural language processing (Shazeer et al., 2017; Fedus et al., 2022; Groeneveld et al., 2024), we validate the training efficiency of MoE models compared to their dense counterparts in the protein domain. These studies have consistently shown that MoE models can achieve the performance of larger dense models with significantly lower training computation cost.

We conduct a controlled experiment comparing three models: a 500M dense model, a 1B dense model, and a sparse MoE model with 500M active parameters and 1.6B total parameters. Our MoE architecture consists of 8 experts per layer, with the top 2 experts activated for each token. The dense models were trained using 64B tokens from the UniRef (Suzek et al., 2007) database, while the MoE model only utilized 32B tokens. All experiments were conducted on identical hardware (16 H200 GPUs).

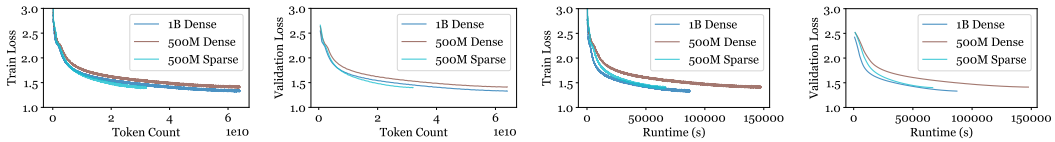


Figure 5: Training and validation loss curves for a 500M dense, a 1B dense, and a MoE (500M active, 1.6B total) model. From left to right: training loss vs. tokens, validation loss vs. tokens, training loss vs. time, validation loss vs. time. The MoE model demonstrates faster convergence.

As shown in Figure 5, the MoE model converges substantially faster than its dense counterparts. Notably, the 500M MoE model reaches a lower validation loss than the 500M dense model using only half the number of tokens and approximately half the training time. It also achieves a better validation loss than the 1B dense model, despite the latter being trained on twice as many tokens. This confirms that the computational benefits of the MoE architecture translate effectively to the protein sequence domain.

12 Additional Analysis

12.1 Representation Collapse Analysis and Mitigation

12.1.1 Analysis of Representation Collapse in MoE Models

The performance disparity observed between Dense and MoE models on regression and fine-grained classification tasks raises a critical question regarding the quality of their learned representations. While MoE models achieve comparable performance on binary classification, their inferiority on tasks requiring continuous or high-cardinality predictions suggests a potential degradation in the expressiveness of their latent space.

To investigate this, we analyze the geometric properties of the embeddings generated by the encoder. Following methodologies in representation learning Ethayarajh (2019); Zhang et al. (2023); Chi et al. (2022), we quantify the "clumping" or anisotropy of the embedding space using three metrics:

- **Average Cosine Similarity (ACS):** Measures the average alignment between randomly sampled pairs of embeddings. Higher ACS indicates a narrower cone of representations.

$$\text{ACS} = \frac{1}{|P|} \sum_{(i,j)\in P} \frac{\mathbf{e}_i \cdot \mathbf{e}_j}{\|\mathbf{e}_i\|\|\mathbf{e}_j\|} \tag{5}$$

- **Isotropy Score:** An entropy-based measure derived from the eigenvalue distribution of the covariance matrix. Lower isotropy indicates that variance is dominated by a few principal directions (anisotropy).

$$\text{Isotropy} = \frac{-\sum_{k=1}^{d} p_k \log(p_k)}{\log(d)} \tag{6}$$

where p_k are normalized eigenvalues of the covariance matrix.

- **Intrinsic Dimensionality (ID):** The minimum number of principal components required to explain 90% of the total variance. A lower ID implies the representations occupy a lower-dimensional manifold.

Table 4: Embedding geometric properties comparison between Dense and MoE models. MoE shows representation collapse signs.

Metric	Dense	MoE	Diff
ACS ↓	0.710	0.837	+0.127
Isotropy ↑	0.493	0.465	-0.028
ID ↑	46.9	39.9	-7.0

Feature Collapse in Sparse Models. We computed these metrics across all eight downstream datasets for both symmetric Dense and symmetric MoE models pretrained on identical compute budgets. As summarized in Table 4 (per-task breakdown in Appendix 12.1.3), MoE models exhibit a consistent trend of *representation collapse* compared to their dense counterparts. Specifically, MoE embeddings show a significantly higher ACS ($+0.127$) and lower Intrinsic Dimensionality (-7.0).

This indicates that the hidden states of MoE models are confined to a narrower cone and a lower-dimensional manifold. We hypothesize that the routing mechanism, by discretizing the computation path, implicitly encourages the hidden states to cluster around the centroids of expert specializations, thereby reducing the global smoothness and expressiveness of the representation space. This "feature collapse" likely limits the model's capacity to resolve fine-grained differences required for regression tasks, explaining the performance gap observed in Figure 3.

12.1.2 Mitigating Collapse via Routing Weights

The analysis in Section 12.1.1 suggests that the final hidden states of MoE models suffer from information loss due to representation collapse. However, the MoE architecture generates an additional set of signals during the forward pass: the *routing weights*. These weights represent the router's decision-making process and encode how the input token relates to the specialized knowledge within the experts.

We hypothesize that while the continuous hidden states may collapse, the routing patterns retain complementary high-dimensional information about the input. To verify this, we investigate combining the encoder's hidden states with routing weights, a technique we term Mixture-of-Experts Embeddings (MoEE), inspired by recent work Li & Zhou (2024).

Method. For each input sequence, we extract: (1) the standard hidden state embedding e_{HS} from the encoder's final layer via average pooling, and (2) the routing weight embedding e_{RW} by averaging the routing probabilities across all tokens for each MoE layer. Since routing weights and hidden states have different dimensions, we first aggregate the per-layer routing weights using three strategies (concatenation, summation, or averaging across layers), and then concatenate the resulting routing embedding with the hidden state embedding. Specifically: ① **Concat:** $e_{\text{RW}} = [r_1; r_2; \ldots; r_L]$, concatenating routing weights from all L layers. ② **Sum:** $e_{\text{RW}} = \sum_{l=1}^{L} r_l$, element-wise summation across layers. ③ **Average:** $e_{\text{RW}} = \frac{1}{L}\sum_{l=1}^{L} r_l$, element-wise averaging across layers. The final embedding is $e_{\text{final}} = [\text{normalize}(e_{\text{HS}}); \text{normalize}(e_{\text{RW}})]$.

Results. As shown in Figure 6, incorporating routing weights notably improves MoE model performance across all three evaluated tasks: Enzyme Catalytic Activity, Fitness Prediction (both regression tasks measured by Spearman ρ), and Fold Prediction (a classification task measured by accuracy). All three MoEE variants (Concat, Sum, Average) substantially outperform the baseline MoE model that uses only hidden state embeddings. This confirms our hypothesis that routing weights compensate for the representation collapse observed in the hidden states, effectively restoring the semantic richness required for complex downstream tasks. The routing decisions made across different layers appear to encode high-dimensional information about how the input is processed through different expert pathways, providing a richer representation of the input sequence. This finding opens an interesting

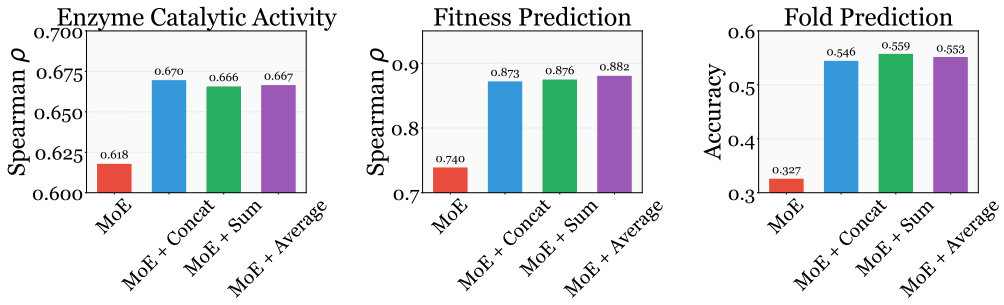


Figure 6: Impact of incorporating routing weights into MoE embeddings. Aggregating routing weights across layers via concatenation, summation, or averaging, then combining with hidden states, notably improves MoE performance across all three tasks: Enzyme Catalytic Activity (Spearman ρ), Fitness Prediction (Spearman ρ), and Fold Prediction (Accuracy).

direction for improving MoE embeddings without additional training, by leveraging the inherent structure of sparse architectures to mitigate their representational limitations.

12.1.3 Detailed Representation Collapse Metrics

Table 5 provides the per-task breakdown of embedding geometric properties for Dense and MoE models.

Table 5: Per-task embedding geometric properties for Dense and Sparse (MoE) models. ACS: Average Cosine Similarity (lower is better), Isotropy: Isotropy Score (higher is better), ID: Intrinsic Dimensionality (higher is better). MoE models consistently show higher ACS and lower ID across all tasks, indicating representation collapse.

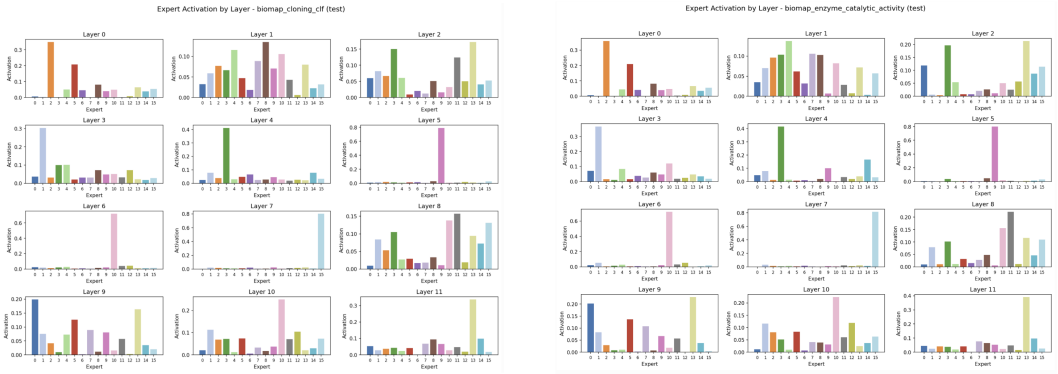
Task	Dense ACS	Sparse ACS	Dense Isotropy	Sparse Isotropy	Dense ID	Sparse ID
Cloning Classification	0.627	0.792	0.527	0.484	55	45
Enzyme Catalytic Activity	0.762	0.862	0.498	0.474	46	40
Fitness Prediction	0.995	0.999	0.376	0.391	18	19
Fold Prediction	0.594	0.767	0.518	0.488	52	44
Material Production	0.650	0.805	0.531	0.492	55	46
Metal Ion Binding	0.649	0.796	0.517	0.493	53	46
Solubility Prediction	0.654	0.820	0.520	0.497	54	47
Stability Prediction	0.750	0.855	0.457	0.401	42	32
Average	**0.710**	**0.837**	**0.493**	**0.465**	**46.9**	**39.9**

12.2 Expert Activation Imbalance on Downstream Tasks

Although our MoE models employ expert bias during pre-training to achieve load balancing (as described in Section 6.1), we observe that expert activation becomes notably imbalanced when the model is applied to downstream tasks. Figure 7 visualizes the expert activation patterns across different encoder layers for two representative downstream tasks.

As shown in Figure 7, the expert activation distribution is relatively balanced in the early and late layers, but exhibits significant imbalance in the middle layers. We hypothesize that this phenomenon arises from the distribution shift between pre-training data (UniRef protein sequences) and downstream task-specific sequences. The middle layers, which are responsible for learning intermediate-level features, appear to be more sensitive to this distribution mismatch.

This activation imbalance may contribute to the representation collapse discussed in Section 12.1.1, as certain experts become over-utilized while others remain underutilized, leading to less diverse representations. Exploring methods to maintain balanced expert activation during inference on out-of-distribution data remains an important direction for future work.



(a) Cloning Classification

(b) Enzyme Catalytic Activity

Figure 7: Expert activation frequency across encoder layers on downstream tasks. Despite achieving balanced expert utilization during pre-training via expert bias, we observe significant activation imbalance on downstream tasks, particularly in the middle layers. This suggests that out-of-distribution sequences from downstream datasets trigger different routing patterns than those seen during pre-training.

13 Ablation Studies

13.0.1 Expert Granularity

Expert granularity refers to the trade-off between the number and size of experts in an MoE layer. For instance, a `Top-2 of 8` configuration can be replaced by a `Top-4 of 16` setup if the FFN hidden dimension of each of the 16 experts is halved. The primary motivation for this approach is the exponential increase in combinatorial flexibility. A greater number of possible expert combinations for each token enhances the model's capacity to route inputs to more specialized pathways, potentially leading to more effective learning and better performance.

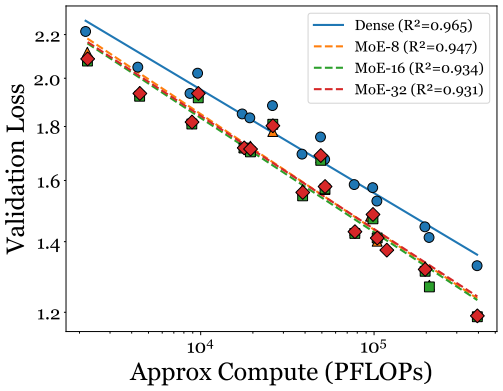


Figure 8: Scaling law comparison across Dense and MoE variants with different expert counts. All MoE configurations converge to comparable loss values while outperforming Dense.

Recent studies have highlighted the benefits of fine-grained experts. Dai et al. (2024) proposed fine-grained expert segmentation to decompose diverse knowledge into more specialized experts, demonstrating that a massive increase in potential expert combinations (e.g., from $\binom{16}{2} = 120$ to $\binom{64}{8} \approx 4.4 \times 10^9$) enhances knowledge acquisition. Similarly, Muennighoff et al. (2024) found that more granular experts consistently improve training and validation loss, although they observed diminishing returns as granularity increased further. Krajewski et al. (2024) also established that compute-optimal models warrant more granular experts, particularly at larger compute budgets.

The performance cost of high granularity is not without precedent; as noted by Zhang et al. (2025b) and Luo et al. (2025), the benefits of fine-grained experts are not "free." Activating k experts requires duplicating the token representation k times before dispatching them, which proportionally increases communication overhead. In our distributed training environment, activating 8 or 16 experts per token likely introduces a communication bottleneck that outweighs the marginal performance gains.

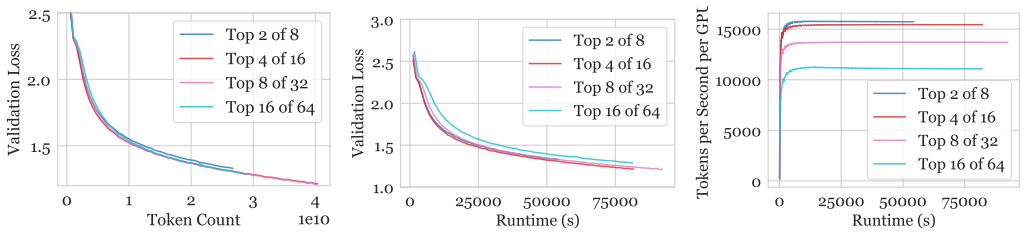


Figure 9: The impact of expert granularity on model convergence and throughput. From left to right: validation loss vs. token count, validation loss vs. wall-clock time, and token throughput vs. wall-clock time. Increasing granularity from `Top-2 of 8` to `Top-4 of 16` improves convergence, but further increases show diminishing returns and hurt throughput.

13.0.2 QK LayerNorm

Some prior works have reported training stability improvements from adding a layer normalization step after the query ($\mathbf{Q}$) and key ($\mathbf{K}$) projections in the attention mechanism, a technique often referred to as "QK LayerNorm" Dehghani et al. (2023); Muennighoff et al. (2024). As noted by Dehghani et al. (2023), this instability often manifests as divergent training loss caused by extremely large values in the attention logits. QK LayerNorm can prevent this by normalizing the queries and keys before the dot-product computation, thus avoiding numeric overflows and stabilizing the network, especially during low-precision training. The attention weights are computed as follows:

$$\texttt{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \texttt{softmax}\left(\frac{\texttt{LN}(Q)\texttt{LN}(K)^\top}{\sqrt{d_k}}\right)\mathbf{V} \tag{7}$$

where $\mathbf{W}_Q$ and $\mathbf{W}_K$ are the query and key weight matrices, and d_k is the dimension of the keys.

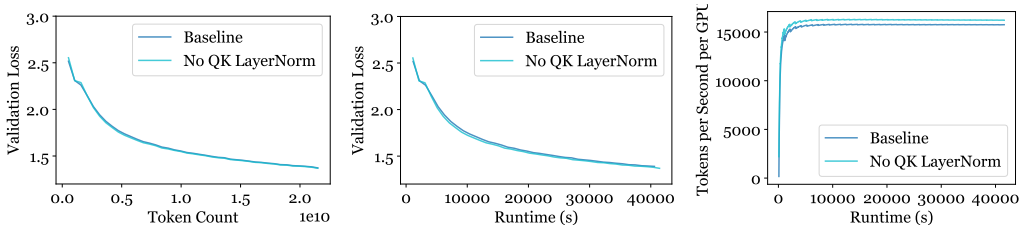


Figure 10: The impact of QK LayerNorm on training efficiency in small-scale experiments with a fixed peak learning rate of 1e-4. From left to right: validation loss vs. token count, validation loss vs. wall-clock time, and token throughput vs. wall-clock time. Removing QK LayerNorm slightly improves convergence per token and more significantly improves convergence per unit of time, while increasing token throughput by approximately 3%.

We investigated the impact of QK LayerNorm on our training dynamics. In our initial small-scale experiments with a fixed peak learning rate of 1e-4, as illustrated in Figure 10, we did not observe significant loss spikes regardless of whether QK LayerNorm was enabled. Removing QK LayerNorm led to a marginal improvement in convergence speed with respect to the number of tokens processed, and more significantly, its removal resulted in a more pronounced acceleration of convergence with respect to wall-clock time, accompanied by an approximately 3% increase in token throughput.

However, when scaling to asymmetric encoder-decoder architectures (as discussed in Section 5), we observed that QK LayerNorm became essential for preventing model divergence. Consequently, we enabled QK LayerNorm for all subsequent experiments to ensure training stability across different architectural configurations.

Importance of Cross-Attention Normalization. A notable implementation detail for encoder-decoder architectures like T5 is that the default Megatron training framework only applies QK LayerNorm to the self-attention layers, not to the cross-attention layers in the decoder. We found that this partial application of QK LayerNorm provided limited stability benefits. Only after extending

QK LayerNorm to include cross-attention layers did we observe substantial improvements in training stability, particularly for asymmetric configurations where the encoder-decoder imbalance exacerbates gradient flow issues.

13.0.3 SHARED EXPERT

The concept of a shared (or fixed) expert, which is always activated in addition to the dynamically routed experts, was proposed by Dai et al. (2024). The intuition is to encourage the shared expert to learn common, domain-general knowledge, thereby allowing the routed experts to focus on more specialized information. This approach is hypothesized to reduce redundancy and increase the model's total information capacity. This shared expert is sometimes referred to as a "residual expert", as the routed experts are formally only required to learn the residual output of the MoE layer.

However, empirical evidence has also suggested that including a shared expert can be detrimental to performance. Muennighoff et al. (2024) observed a marginal degradation in performance and hypothesized that this is because replacing a routed expert with a fixed one drastically reduces the number of possible expert combinations, thereby limiting the model's flexibility and representational power.

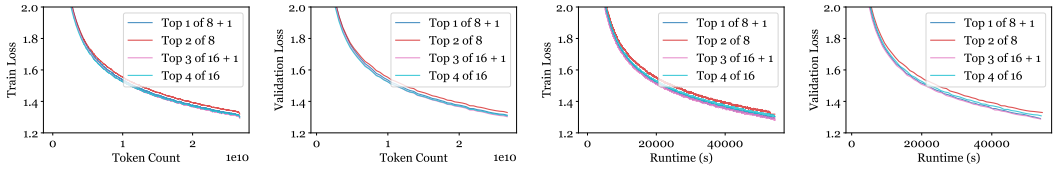


Figure 11: The impact of including a shared expert on model convergence. We compare two configurations within an 8-expert layer: activating two routed experts versus activating one shared and one routed expert.

To investigate this trade-off, we conducted experiments on configurations with 8 and 16 experts. In contrast to the findings of Muennighoff et al. (2024), we observed that the inclusion of a shared expert consistently improved convergence speed, both in terms of tokens processed and wall-clock time. This discrepancy may arise because we add a shared expert to the existing pool of experts rather than replacing a routed expert with it. In our setup, which utilizes expert parallelism for training acceleration, the memory overhead for activating `Top-1 of 7 + 1 shared` versus `Top-1 of 8 + 1 shared` is nearly identical, as each GPU holds at least two experts. The latter configuration, however, avoids potential load imbalance issues. Therefore, we opted to add an extra shared expert instead of replacing a routed one, and based on the positive results, we include a shared expert in our final MoE architecture.

13.0.4 LOAD BALANCING STRATEGY

A critical challenge in training MOE models is ensuring a balanced load across experts. Imbalanced routing can lead to a vicious cycle where some experts receive fewer tokens, become undertrained, and are subsequently selected even less often, potentially leading to expert collapse. Furthermore, load imbalance can cause performance bottlenecks in distributed training environments by unevenly distributing computational load across GPUs.

To mitigate this, Fedus et al. (2022) proposed an auxiliary load balancing loss to penalize imbalanced routing. The loss, $\mathcal{L}_{LB}$, is computed by multiplying the fraction of tokens in a batch routed to an expert, f_i, with the average routing probability for that expert, P_i, summed across all N_E experts:

$$\mathcal{L}_{LB} = N_E \cdot \sum_{i=1}^{N_E} f_i \cdot P_i \tag{8}$$

This auxiliary loss is scaled by a loss weight α, an optional hyperparameter typically set to 0.01 to control the magnitude of the penalty. In our experiments involving this loss, we kept the weight fixed at $\alpha = 0.01$.

An alternative approach, expert bias, was proposed to achieve load balancing while avoiding the potentially harmful gradients introduced by an auxiliary loss term Wang et al. (2024). In this method, a learnable bias term b_i is added to the router logits $s_{i,t}$ for each expert before the TopK selection:

$$g_{i,t} = \begin{cases} s_{i,t} + b_i, & \text{if } (s_{i,t} + b_i) \in \texttt{TopK}, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Notably, the expert bias term b_i only influences the TopK selection for routing and does not participate in the weighted sum of the expert outputs. As the bias term is not updated via backpropagation, Wang et al. (2024) propose a periodic update rule based on the load imbalance from previous steps. This technique has been successfully used in models from the DeepSeek series to replace the auxiliary loss Dai et al. (2024).

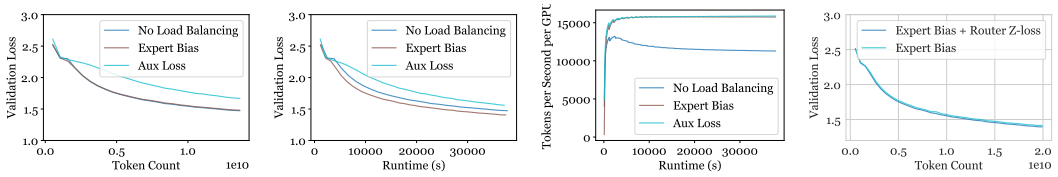


Figure 12: Comparison of load balancing strategies and router z-loss. From left to right: validation loss vs. tokens, validation loss vs. time, token throughput, and router z-loss impact.

We investigated the impact of these strategies on model convergence, as shown in Figure 12. We compared three settings: using no load balancing, using only expert bias, and using only the auxiliary loss. The model with no load balancing achieved a convergence rate per token comparable to, or even slightly faster than, the model with expert bias. However, its convergence with respect to wall-clock time was significantly slower due to much lower token throughput. Conversely, using the auxiliary loss resulted in a marked degradation in convergence speed, despite achieving the same high token throughput as the expert bias configuration. This finding contrasts with results from OLMoE Muennighoff et al. (2024), where the load balancing loss was beneficial. We hypothesize this discrepancy may stem from fundamental differences between the natural language and protein sequence domains. Given these results, we selected expert bias as our sole load balancing mechanism.

13.0.5 Router Z-loss

To further enhance training stability, Zoph et al. (2022) proposed the router z-loss. This auxiliary loss penalizes large logits entering the gating network, as such values can lead to numeric overflows in the large matrix multiplications within the MoE layer. The loss is computed by taking the log-sum-exp of the pre-router logits x_j for each expert, squaring the result, and averaging across the batch B, thereby assigning a larger penalty to larger logits:

$$\mathcal{L}_{RZ}(x) = \frac{1}{B} \cdot \sum_{i=1}^{B} \left(\log \sum_{j=1}^{N_E} \exp(x_j^{(i)}) \right)^2 \tag{10}$$

The loss is scaled by a weight β, which is commonly set to 0.001. In our experiments, we adhere to this standard value.

Our investigation into the impact of this loss, presented in the rightmost panel of Figure 12, revealed that including the z-loss improves convergence speed. This observation contrasts with the findings reported for OLMoE Muennighoff et al. (2024), where the z-loss was found to be detrimental to convergence. Further investigation may be required to determine whether this discrepancy stems from architectural differences (our T5-style encoder-decoder vs. OLMoE's decoder-only model) or the distinct statistical properties of protein sequence data compared to natural language. Given its positive impact in our setup, we retain the router z-loss in our final configuration.

13.1 General Pretraining Settings

We investigated several general pretraining settings including QK LayerNorm Dehghani et al. (2023); Muennighoff et al. (2024) and embedding shrinking factor Zeng et al. (2022). In small-scale

experiments with fixed learning rates, removing QK LayerNorm led to approximately 3% increase in token throughput without significant loss spikes. However, QK LayerNorm proved essential for stabilizing asymmetric architectures at larger scales, so we enabled it for all experiments. Notably, for T5-style encoder-decoder models, extending QK LayerNorm to cross-attention layers (beyond the default self-attention only) was critical for achieving robust training stability. Detailed ablation results are provided in Appendix 13.0.2.

13.1.1 Embedding Shrinking Factor

Training collapses in large language models are often preceded by "spikes" in the gradient norm, which are frequently caused by abnormally large gradients originating from the embedding layer Zeng et al. (2022). To mitigate this, the Embedding Layer Gradient Shrink (EGS) strategy was proposed. This technique stabilizes training by shrinking the embedding layer's gradients at each step. Let $\mathbf{E}$ be the word embedding matrix and α be the shrinking factor, the update is implemented as:

$$\mathbf{E}_{\texttt{new}} = \alpha \cdot \mathbf{E}_{\texttt{old}} + (1 - \alpha) \cdot \texttt{sg}(\mathbf{E}_{\texttt{old}}) \tag{11}$$

where $\texttt{sg}(\cdot)$ is the stop-gradient operator. This effectively interpolates between the updated embedding and its value from the previous step, dampening large, unstable updates.

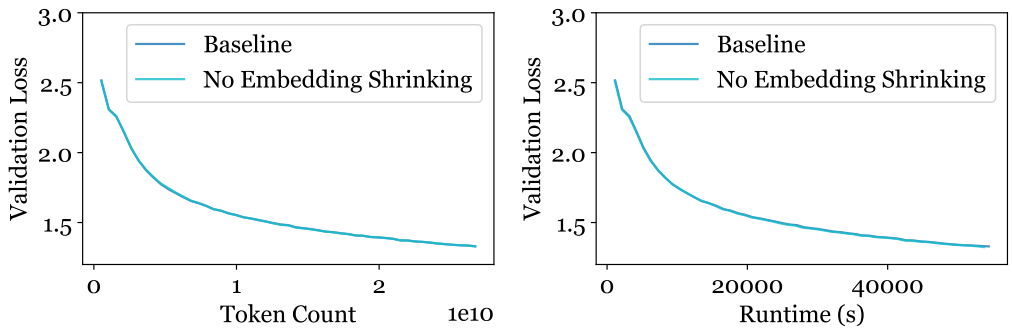


Figure 13: Training loss comparison for models trained with and without the Embedding Shrinking factor ($\alpha = 0.1$). Left: Validation loss vs. token count. Right: Validation loss vs. wall-clock time.

We experimented with an embedding shrinking factor of $\alpha = 0.1$, a value empirically shown to wipe out most gradient spikes Zeng et al. (2022). As illustrated in Figure 13, we found that applying the embedding shrinking factor did not negatively impact convergence speed or token throughput. Given its potential to stabilize training without incurring a performance penalty, we decided to retain this technique in our final configuration.

14 Limitations and Future Work

Our study reveals several limitations and directions for future research:

Representation Collapse in MoE Models. We identified that MoE models suffer from representation collapse, exhibiting higher embedding similarity and lower intrinsic dimensionality compared to dense models. While MoEE approach partially mitigates this issue by incorporating routing weights, developing training-time solutions to prevent collapse remains an open challenge.

Expert Activation Imbalance. Despite achieving balanced expert utilization during pre-training via expert bias, we observed significant activation imbalance on downstream tasks, particularly in middle layers (Appendix 12.2). This distribution shift between pre-training and downstream data suggests the need for more robust routing mechanisms that generalize across domains.

Asymmetric Architecture Instability. Our asymmetric encoder-decoder configurations showed inconsistent benefits, with the 500M scale exhibiting particularly challenging optimization behavior. While QK LayerNorm (including cross-attention) improves stability, asymmetric architectures do not reliably outperform symmetric baselines across all scales. Future work should explore more principled approaches to asymmetric design.