DECO: Sparse Mixture-of-Experts with Dense-Comparable Performance on End-Side Devices

Anonymous ACL submission

Abstract

While Mixture-of-Experts (MoE) scales model capacity without proportionally increasing computation, its massive total parameter footprint creates significant storage and memory-access bottlenecks, which hinder efficient end-side deployment that simultaneously requires high performance, low computational cost, and small storage overhead. To achieve these properties, we present **DECO**, a sparse MoE architecture designed to match the performance of dense Transformers under identical total parameter budgets and training tokens. DECO utilizes the differentiable and flexible ReLU-based routing enhanced by learnable expert-wise scaling, which adaptively balances the contributions of routed and shared experts. Furthermore, we introduce NormSiLU, an activation function that normalizes inputs prior to SiLU operators, producing a more stable trend of routed-expert activation ratio and a higher intrinsic sparsity level. We also identify an empirical advantage in using non-gated MLP experts with ReLU-based routing, indicating the possibility of MoE architecture simplification. Experiments demonstrate that DECO, activating only 20% of routed experts, matches dense performance and outperforms established MoE baselines. Our specialized acceleration kernel delivers a 2.93× speedup on Jetson AGX Orin compared with dense inference. Code and checkpoints are available.

1 Introduction

The scale of large language models (LLMs) has grown rapidly to achieve consistent performance gains across diverse tasks. The rising training and deployment costs for massive LLMs have made mixture-of-experts (MoE) an increasingly prominent model architecture. The key property of MoE is the sparse activation, namely, activating a small subset of expert modules from a large pool of parameters. Therefore, MoE retains high capacity and strong performance while substantially reducing computation costs.

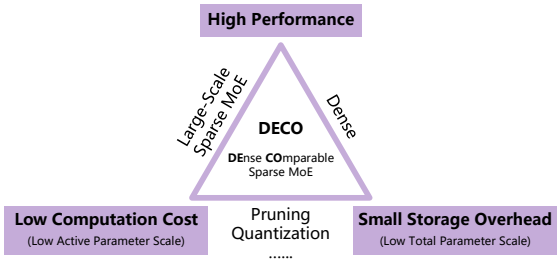

Figure 1: The "ideal triangle" of end-side MoE. Beyond the high performance and reduced computational cost of sparse MoE, the model should maintain a minimal storage footprint, achieving high performance within dense-comparable total parameter budgets.

As a research hotspot, MoE has been extensively studied, from architecture design (Liu et al., 2024; Cai et al., 2025) to scaling laws and compute-optimal settings (Krajewski et al., 2024; Tian et al., 2025). Prior work primarily pursues two objectives: **high performance** and **low computation cost**. However, when it comes to end-side deployment, a third non-negligible objective emerges: **small storage overhead**. Concretely, MoE with a huge number of total parameters demands substantial storage space. More critically, large MoE models may incur high memory-access costs when transferring experts between GPU high-bandwidth memory and shared memory (Li et al., 2025), or when moving offloaded parameters from disk/flash storage to GPU/NPU memory of end-side devices. Such latency can erode the efficiency gains afforded by sparse computation.

Therefore, as shown in Figure 1, an ideal MoE model for end-side deployment should satisfy the above three objectives. To pursue this "ideal triangle", we pose the question:

> *Can a sparse MoE model achieve performance comparable to a dense model, given the same total parameter budget and the same number of training tokens?*

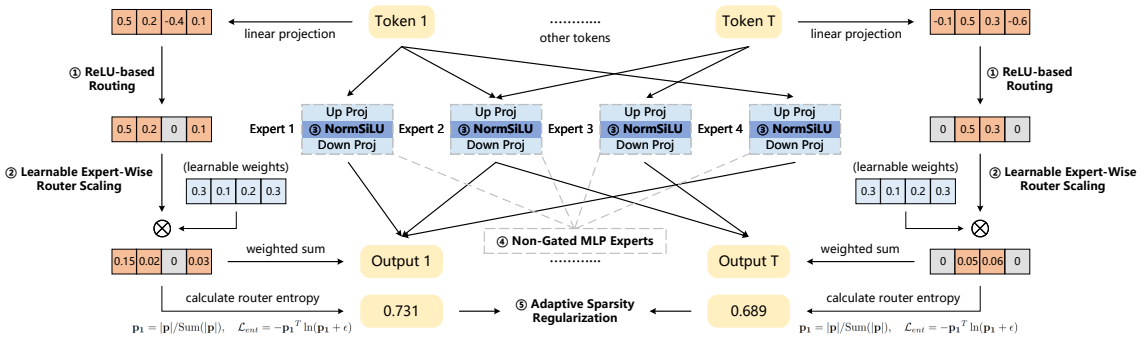

Figure 2: The overall architecture of DECO. For router design, we adopt ReLU-based routing enhanced by learnable expert-wise router scaling. For expert design, we propose NormSiLU as a better routed-expert activation function and employ non-gated MLP experts. For precise sparsity control, we employ adaptive sparsity regularization.

A closely related study by Li et al. (2025) identifies the optimal settings of DeepSeek-V3-style MoE architectures that enable them to surpass dense models under matched total parameters and computation budget. However, due to the low per-token computation of sparse MoE, in that work, MoE settings are trained on substantially more tokens under the same computation budget. We adopt a stricter setting that requires exactly the same number of training tokens.

To achieve this goal, we propose **DECO** (Figure 2), a sparse MoE architecture that achieves **DE**nse-**CO**mparable performance through a fundamental revision of MoE design.

For router design, conventional MoE models generally adopt TopK routing, which is non-differentiable and enforces a uniform activation ratio across all tokens. To overcome this issue, we adopt **ReLU-based routing**, a differentiable paradigm that enables flexible token-dependent activation ratios. Moreover, to mitigate output scale imbalances between shared and routed experts, while simultaneously accounting for potential expert heterogeneity, we introduce **learnable expert-wise router scaling**. This mechanism involves learnable scaling factors to calibrate the contribution of individual routed experts.

The expert design is similarly optimized for stability and efficiency. Empirical analysis reveals that coupling ReLU-based routing with vanilla SiLU-activated experts results in two critical problems: a surging routed-expert activation ratio (Figure 4) and vanishing SiLU output magnitudes (Figure 6). To resolve these issues, we propose **NormSiLU**, which applies dual-stage normalization prior to the SiLU operator. NormSiLU stabilizes activation trends and reduces the activation ratio, alleviating the need for aggressive sparsity regularization. It also produces more stable and significant SiLU output magnitudes, promoting better utilization of expert parameters. Beyond the activation function, we employ **non-gated MLP experts** rather than standard gated variants, as they exhibit superior empirical compatibility with ReLU-based routing. Finally, to precisely control the activation ratio, we design an **adaptive sparsity regularization** that auto-scales the regularization strength.

As shown in Section 4, at the activation ratio of 20%, DECO demonstrates performance comparable to dense models when matched for total parameters and training tokens. DECO also surpasses established MoE baselines of the same scale and activation ratio.

Finally, we implement a tailored acceleration kernel for DECO to test its practical inference acceleration value. Based on CUTLASS (Thakkar et al., 2023), the kernel leverages tensor cores to improve computational throughput and reduces memory-access overhead by exploiting the sparse activation. Overall, the kernel achieves a speedup of 2.93$\times$ compared with vanilla dense inference on the real edge device of Jetson AGX Orin.

2 Related Works

To achieve high performance while curbing computational growth, MoE has recently risen as the mainstream architecture. An MoE typically comprises three components: a router, a set of experts, and an auxiliary training objective.

Router design. The router computes weights assigned to each expert and selects which experts to activate. It generally consists of a linear projection, an activation function, and post-processing of router scores.

The activation function controls the expert selection pattern. Many MoE designs use TopK, which forces each token to activate a fixed num-

ber of experts (Jiang et al., 2024; Dai et al., 2024). However, TopK is criticized for its inflexibility (an input-invariant number of active experts) and non-differentiability. TopP (Huang et al., 2024) selects experts by a threshold p, activating experts until their cumulative router score reaches at least p, thereby permitting token-dependent activation ratios. MoE++ (Jin et al., 2024) retains TopK but introduces zero-computation experts, which indirectly allows variable computation cost. To improve differentiability, ReMoE (Wang et al., 2024b) and BlockFFN (Song et al., 2025b) adopt ReLU for expert selection. Since ReLU naturally produces considerable zero outputs while remaining differentiable, it enables smoothly learnable activation ratios and delivers performance advantages.

Post-processing of router scores primarily normalizes expert weights to preserve a consistent output scale. Most designs use Softmax as the score normalizer. DeepSeek-V3 (Liu et al., 2024) instead applies element-wise Sigmoid followed by unit-sum normalization, which helps mitigate extremely skewed score distributions. Notably, DeepSeek-V3 also introduces a scalar scaling factor applied to router scores, which helps balance contributions between shared and routed experts.

In this work, DECO adopts ReLU-based routing, and replaces the fixed scalar scaling factor with learnable expert-wise router scaling factors, providing flexibility and accommodating potential heterogeneity in expert output scales.

Expert design. In most mainstream MoE models, each expert is a standard gated MLP with SiLU activation (SwiGLU) (Shazeer, 2020). DeepSeek-MoE (Dai et al., 2024) shows the benefits of introducing fine-grained experts and shared experts but retains the SwiGLU backbone.

DECO refines expert design by introducing NormSiLU as the expert activation, which resolves the issues of surging routed-expert activation ratio and vanishing SiLU output magnitudes. Moreover, we find that with ReLU-based routing, non-gated MLP experts empirically bring a more stable trend of activation ratio than the gated variant.

Auxiliary training objective. Aside from the language modeling loss, MoE models generally introduce auxiliary training objectives. The most common one is for load balancing, typically implemented via the auxiliary loss proposed by Fedus et al. (2022). To alleviate auxiliary-loss interference with language modeling, DeepSeek-V3 adopts a loss-free load-balancing policy without a differentiable objective (Wang et al., 2024a). MoE models with variable activation ratios often incorporate a sparsification objective. For example, ReMoE applies adaptive L1-norm regularization, and BlockFFN employs chunk-wise sparsification (Wang et al., 2024b; Song et al., 2025b).

Inspired by ReMoE, DECO uses an adaptive sparsity regularization, whose coefficient auto-scales to precisely control the sparsity level. We also replace the L1-norm with router entropy to improve numerical stability.

3 Methodology of DECO

We propose DECO, a sparse MoE architecture with performance comparable to dense variants, given the same total number of parameters and training tokens. We split our design into three components: the router (Section 3.1), experts (Section 3.2), and adaptive sparsity regularization (Section 3.3).

3.1 Router Design

ReLU-based routing. Distinct from conventional TopK routing, DECO incorporates ReLU in the router to determine expert activation. As demonstrated in prior studies (Yao et al., 2025; Song et al., 2025b), ReLU is fully differentiable, inherently induces sparsity, and supports token-dependent activation ratios. These attributes render ReLU a robust and flexible routing function.

Learnable expert-wise router scaling. To balance the output scales of routed and shared experts, DECO applies a scaling operator to the routing scores before they are multiplied by the expert outputs. We extend the fixed scalar scaling factor of DeepSeek-V3 (Liu et al., 2024) to a learnable vectorized one. This modification accommodates the potential heterogeneity across routed experts by assigning them distinct, learnable coefficients.

Formally, given the hidden dimension d_h, the expert number N_e, and the input hidden state $\mathbf{x} \in \mathbb{R}^{d_h}$, the router score $\mathbf{p}$ can be computed as follows:

$$\mathbf{p} = \boldsymbol{\alpha} \odot \mathrm{ReLU}(\mathbf{W}_{router}^{T}\mathbf{x}), \tag{1}$$

where $\mathbf{W}_{router} \in \mathbb{R}^{d_h \times N_e}$ and $\boldsymbol{\alpha} \in \mathbb{R}^{N_e}$ are learnable weights, and $\odot$ is element-wise multiplication.

3.2 Expert Design

Non-gated MLP experts. While gated MLP is widely considered superior to the non-gated variant (Shazeer, 2020), we observe that non-gated experts exhibit more favorable properties within

Algorithm 1 Pseudocode of NormSiLU.

```
# Nt: num_tokens; De: dim_expert; Dh: dim_hidden
# Ne: num_experts; Na: num_active_experts
rms_norm = RMSNorm(dim=dim_expert)  # learnable RMSNorm layer, weight shape is [De]

def NormSiLU(input_hidden, up_proj_weight, intermediate_state):
    # input_hidden: [Nt, Dh], the input hidden state of MoE
    # up_proj_weight: [Ne, De, Dh], the up-projection weights of experts
    # intermediate_state: [Nt, Na, De], the expert intermediate state produced by a
    sparse linear operation between "input_hidden" and "up_proj_weight"

    # inter-expert mean normalization: conducted at the "num_experts" dimension
    up_proj_avg = mean(up_proj_weight, dim=0)  # [De, Dh], fixed during inference
    intermediate_avg = matmul(input_hidden, up_proj_avg.T)  # [Nt, De]
    intermediate_state -= intermediate_avg.unsqueeze(1)

    # intra-expert RMS normalization: conducted at the "dim_expert" dimension
    intermediate_state = rms_norm(intermediate_state)
    return SiLU(intermediate_state)  # standard SiLU activation
```

the specific context of ReLU-based routing. Concretely, in a ReLU-activated MoE, non-gated experts obtain a more stable trend of activation ratio, whereas gated variants exhibit a sharply increasing trend. This inherent stability implies that a significantly lower regularization penalty is required to achieve a target sparsity threshold, thereby alleviating the negative impact on performance.

NormSiLU. We introduce NormSiLU (Algorithm 1) as an enhanced activation for MoE experts, prepending a dual-stage normalization to the SiLU non-linearity within each expert.

First, inter-expert mean normalization is mathematically equivalent to centering the per-expert up-projection weights around the cross-expert mean, ensuring the pre-activation input distribution is approximately zero-centered. This adjustment stabilizes the SiLU activation distribution within experts. Second, intra-expert RMS normalization is applied to maintain consistent activation magnitudes. We find that this dual-stage normalization not only prevents internal expert activations from vanishing, but also promotes a steady activation ratio at the router level. A theoretical justification for its rationality is presented in Appendix E.

Given the expert intermediate dimension d_e, the structure of a DECO expert is formally defined as:

$$\begin{aligned} \mathbf{x}_{up} &= \mathrm{SparseLinear}(\mathbf{x}, \mathbf{W}_{up}), \\ \mathbf{x}'_{up} &= \mathrm{NormSiLU}(\mathbf{x}, \mathbf{W}_{up}, \mathbf{x}_{up}), \\ \mathbf{y} &= \mathrm{SparseLinear}(\mathbf{x}'_{up}, \mathbf{W}_{down}), \end{aligned} \tag{2}$$

where $\mathbf{W}_{up} \in \mathbb{R}^{N_e \times d_e \times d_h}$ and $\mathbf{W}_{down} \in \mathbb{R}^{N_e \times d_h \times d_e}$ are the up-projection and down-projection weights, respectively. SparseLinear operator facilitates sparse linear operations by involving only active experts at inference time.

3.3 Adaptive Sparsity Regularization

To effectively control the sparsity level, we adopt an adaptive sparsity regularization, based on the router entropy loss and a dynamic scaling algorithm for the coefficient.

Router entropy is a sparsification loss applied to a normalized router score, which is calculated as:

$$\mathbf{p_1} = |\mathbf{p}|/\mathrm{Sum}(|\mathbf{p}|), \mathcal{L}_{ent} = -\mathbf{p_1}^T \ln(\mathbf{p_1} + \epsilon) \tag{3}$$

where $\mathbf{p}$ is the router score defined in Equation 1. The router entropy loss, $\mathcal{L}_{ent}$, is multiplied by a coefficient λ and added to the training objective.

Instead of using a static coefficient, inspired by ReMoE (Wang et al., 2024b), we adaptively scale λ according to the current sparsity level. Specifically, if the current sparsity falls below the target sparsity, λ is scaled by a factor $\eta > 1$ for the subsequent iteration; otherwise, λ is divided by η. In this way, DECO maintains a stable activation ratio centered precisely at the desired sparsity level.

4 Experiments

4.1 Main Results

To demonstrate the architectural rationality of DECO, we introduce the following baselines: Dense (i.e., a standard LLaMA-style Transformer using SwiGLU FFNs (Touvron et al., 2023)), TopP (Huang et al., 2024), DeepSeek-V3 (Liu et al., 2024), ReMoE (Wang et al., 2024b), and BlockFFN (Song et al., 2025b). Four total parameter scales are involved: Small (0.11B), Medium (0.24B), Large (0.53B), and XLarge (1.18B).

All settings are trained on the same high-quality data mixture. The model performance is evaluated by two metrics: Perplexity (PPL) on the C4 English

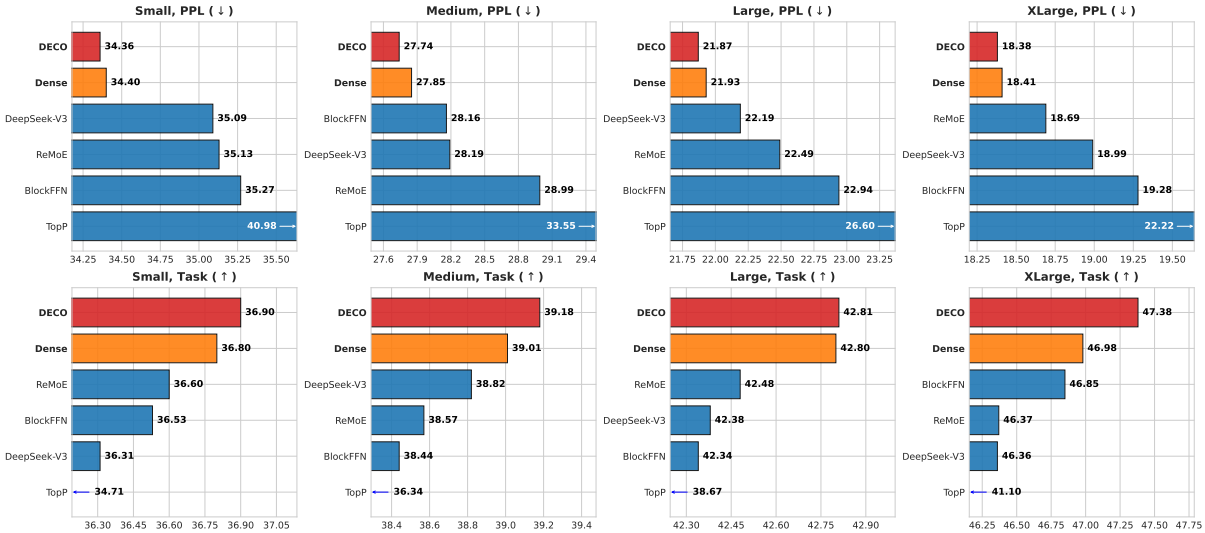

Figure 3: The evaluation results of DECO versus baseline settings. "PPL" and "Task" indicate the C4 validation perplexity and the average accuracy (%) on downstream benchmarks, respectively. DeepSeek-V3 uses gated MLP experts, and ReMoE uses non-gated ones. This is due to their better performance than the opposite settings; see Section 4.4 for detailed discussions.

validation set (Raffel et al., 2020), and average accuracy across a suite of commonsense reasoning benchmarks. See Appendix B for more details about experimental settings and Appendix D for repeated experiments.

To ensure a rigorous comparison, within each group of *the same total parameter count*, *the number of training tokens is also held consistent* at around 40 times the parameter count. All non-FFN components, including attention layers and embedding layers, remain identical within each group. All MoE settings within a group share the same routed-expert activation ratio (*around 20% on the training data*) and intermediate dimension of the shared expert. Furthermore, we ensure that all routed experts have close parameter counts to maintain consistent expert granularities.

From the results shown in Figure 3 (evaluation results on individual benchmarks are shown in Appendix F), we derive the following conclusions:

(1) *Dense comparability*: With an average routed-expert activation ratio of 20%, DECO achieves performance parity with the Dense baseline. This holds true under the same total parameter budget and training token volume, demonstrating DECO's efficiency in maintaining dense-level representation power with reduced active computation.

(2) *Performance superiority*: Under the same routed-expert activation ratio, shared-expert dimensions, and expert granularity, DECO surpasses existing MoE baselines in both perplexity and downstream task performance.

4.2 Effect of Expert-Wise Router Scaling

	Small		Medium	
	PPL (↓)	Task (↑)	PPL (↓)	Task (↑)
Fixed	34.43	36.74	27.94	39.08
Scalar	35.16	36.63	27.92	38.54
DECO	**34.36**	**36.90**	**27.74**	**39.18**

Table 1: The ablation results on the effect of learnable expert-wise router scaling. "Fixed" indicates one fixed scaling factor, while "Scalar" includes one shared learnable scalar scaling factor.

To demonstrate the effect of DECO's router scaling design, we experiment on two ablation settings: "*Fixed*" adopts a constant scaling factor for all routed experts, and "*Scalar*" involves a single learnable scalar scaling factor shared by experts. Both ablation settings are initialized with the same values as DECO. Evaluation results in Table 1 reveal the performance benefits of learnable vectorized scaling factors. We further explore the impact of initialization value in Appendix C.

As empirical justification for this design, we analyze the distribution of expert output norms on the C4 validation set and find clear heterogeneity across the output scales of routed experts. For example, in the first MoE layer of DECO (Medium), the average output norm of all experts is 0.23, while the maximum value exceeds 1.14. This validates our hypothesis that applying expert-specific, learnable vectorized factors is essential to accommodate the varying output scale across different experts.

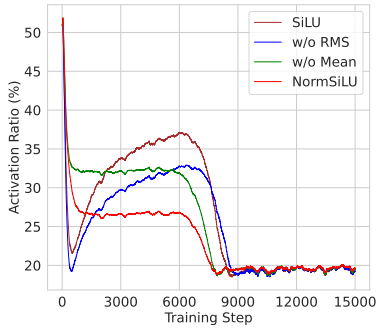

Figure 4: The trend of activation ratio of DECO (Small) and ablation settings each removing one step of NormSiLU. The baseline "SiLU" and "w/o RMS" settings show *surging routed-expert activation ratio* before being pulled back by regularization.

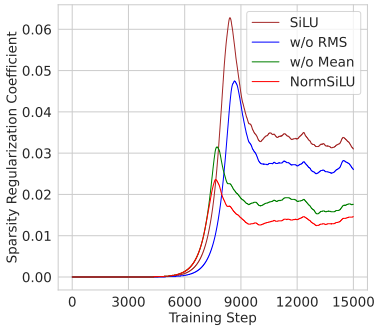

Figure 5: The trend of the regularization coefficient of DECO (Small) and ablation settings each removing one step of NormSiLU. The settings "SiLU" and "w/o RMS" show significantly higher coefficients, which potentially harm performance.

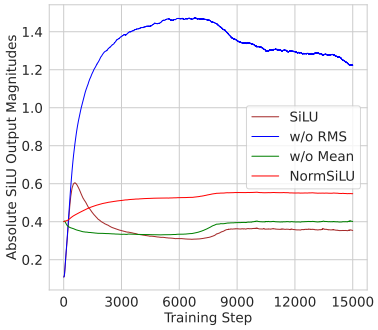

Figure 6: The average absolute output magnitudes of SiLU within routed experts of DECO (Small) and ablation settings each removing one step of NormSiLU. The settings "SiLU" and "w/o Mean" show *vanishing SiLU output magnitudes*.

4.3 Effect of NormSiLU

	Small		Medium	
	PPL (↓)	Task (↑)	PPL (↓)	Task (↑)
w/o Mean	34.57	36.85	27.77	38.88
w/o RMS	35.77	36.78	27.97	39.04
SiLU	35.69	36.46	28.05	38.85
NormSiLU	**34.36**	**36.90**	**27.74**	**39.18**

Table 2: The ablation results on the effect of NormSiLU. The marker "w/o" means removing a specific normalization step, and "SiLU" is the vanilla SiLU operator without any normalization.

To validate the effect of NormSiLU, which incorporates both inter-expert mean normalization and intra-expert RMS normalization, we evaluate three ablation settings: "*w/o Mean*" removes inter-expert mean normalization, "*w/o RMS*" removes intra-expert RMS normalization, and "*SiLU*" is the standard SiLU operator without any normalization.

As shown in Table 2, both normalization steps contribute positively, with intra-expert RMS normalization providing a more substantial gain. To investigate the underlying mechanisms of NormSiLU, we track three critical variables throughout the training process: the routed-expert activation ratio, the sparsity regularization coefficient, and the average absolute output magnitudes of SiLU within experts.

As illustrated in Figure 4, the activation ratios of "SiLU" and "w/o RMS" surge rapidly during the initial training phase. While the adaptive regularization pulls back this surge, Figure 5 reveals that these settings require a significantly higher regularization coefficient, which typically degrades overall performance. Conversely, NormSiLU and "w/o Mean" maintain stable activation trends, with NormSiLU achieving the lowest activation ratio. We conclude that **intra-expert RMS normalization mitigates the uncontrolled growth of activation ratios, while inter-expert mean normalization further promotes sparsity level**.

Moreover, analysis of the internal SiLU output magnitudes (Figure 6) reveals that "SiLU" and "w/o Mean" exhibit considerably lower magnitudes. This suggests that expert neurons (i.e., parameter columns/rows) in these settings are potentially under-utilized and less significantly activated. In contrast, **inter-expert mean normalization effectively addresses this issue, ensuring more robust activation and utilization of expert neurons**.

4.4 Effect of Expert Gating

Setting	Small		Medium		Large	
	PPL (↓)	Task	PPL (↓)	Task	PPL (↓)	Task
DS-V3 (GA)	**35.09**	36.31	**28.19**	**38.82**	**22.19**	**42.38**
DS-V3 (NG)	35.16	**36.73**	28.28	38.47	22.33	42.02
ReMoE (GA)	36.02	**36.61**	29.55	38.15	22.78	42.27
ReMoE (NG)	**35.13**	36.60	**28.99**	**38.57**	**22.49**	**42.48**
DECO (GA)	39.77	35.78	32.46	37.29	22.13	42.29
DECO (NG)	**34.36**	**36.90**	**27.74**	**39.18**	**21.87**	**42.81**

Table 3: The ablation results on the effect of expert gating. "GA" and "NG" indicate gated MLP experts and non-gated MLP experts, respectively. "DS-V3" indicates DeepSeek-V3.

To investigate whether expert gating significantly influences MoE performance, we conduct experiments on three architectures: DeepSeek-V3, ReMoE, and DECO. DeepSeek-V3 is a well-

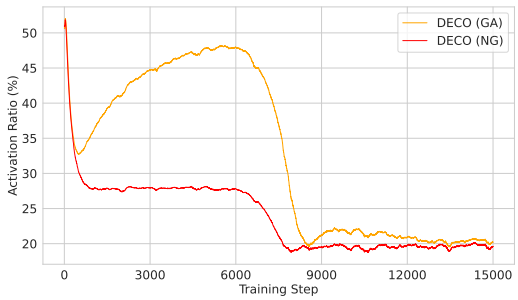

Figure 7: The trend of routed-expert activation ratio of DECO (Small) using different expert gating policies.

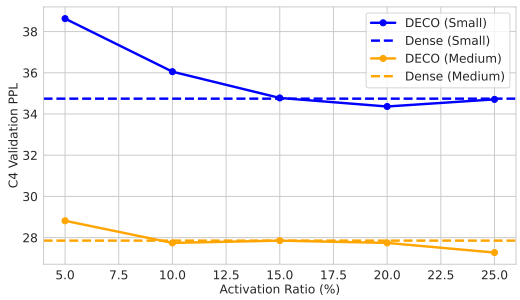

Figure 8: The impact of the routed-expert activation ratio on the performance of DECO.

performing MoE architecture using TopK routing, while ReMoE and DECO use ReLU-based routing to implement a flexible activation ratio. For each architecture, we compare non-gated MLP experts (NG) against gated MLP experts (GA).

As demonstrated by Table 3, **for ReLU-based routing, non-gated MLP experts generally surpass gated counterparts**. Conversely, for standard TopK routing (e.g., DeepSeek-V3), gated experts provide a marginal performance gain, though the difference is not significant.

We attribute this disparity to the training dynamics that emerge when gated experts are paired with flexible, threshold-based routing. As illustrated in Figure 7, DECO (GA) exhibits highly unstable activation trends, characterized by a drastic surge in the routed-expert activation ratio that must be aggressively counteracted by sparsity regularization. In contrast, DECO (NG) maintains a stable activation trajectory, requiring substantially less regularization and thereby preserving model performance.

Mechanistically, this divergence may stem from gradient behavior. Compared with non-gated variants, gated experts (e.g., SwiGLU) contain more multiplicative interactions that produce highly dynamic output scales, sending massive gradient signals back to the router. Because ReLU-based routing couples activation directly to the logit threshold, this gradient surge drastically destabilizes the activation ratio. Conversely, in TopK-routed architectures like DeepSeek-V3, the hard constraint of activating a fixed number of experts per token effectively masks this logit-induced instability, rendering the overall performance largely insensitive to the choice of expert gating.

4.5 Effect of Activation Ratio

We evaluate DECO across a wide range of routed-expert activation ratios, from 5% to 25%. The experimental results, illustrated in Figure 8, yield two primary observations:

(1) Positive correlation with performance: DECO's performance scales positively with the activation ratio. This aligns with findings in standard TopK MoE, where increased activation ratios correspond to higher per-token computational investment, typically resulting in superior performance.

(2) Scale-dependent comparability thresholds: The activation ratio required for DECO to achieve parity with dense models varies by scale. Specifically, the "Small" setting reaches dense-level performance at a 15% activation ratio, whereas "Medium" requires only 10%. This suggests that as DECO scales up, it may attain dense-comparable parameter efficiency at lower activation ratios. This observation is consistent with recent literature suggesting that the optimal activation ratio decreases as total parameter count increases (Zhao et al., 2025a). Further investigation on larger-scale models is required to validate this scaling trend.

Besides activation ratio, we also study the effect of other MoE-specific factors, including expert granularity and shared expert size; see Appendix A.

5 Practical Inference Acceleration

To demonstrate the practical utility of DECO for real-world deployment, we implement an acceleration kernel tailored for DECO, which leverages sparse activation to reduce both computational overhead and memory access latency incurred by inactive routed experts. We evaluate the kernel's efficacy on Spec-Bench (Xia et al., 2024), a comprehensive acceleration benchmark, using a single NVIDIA RTX 4090 (24GB) and Jetson AGX Orin (64GB) as the inference devices. We establish two baselines: **Dense**, which employs standard autoregressive decoding without sparsity optimizations, and **TopK**, which substitutes our ReLU-based routing with conventional TopK. To ensure a strictly controlled evaluation, all baselines and our pro-

Device	Setting	MT.	Trans.	Summ.	QA	Math	RAG	Average
RTX 4090 24GB	Dense	86.79	88.01	86.38	88.08	87.98	86.42	**87.28**
	TopK	194.12	203.75	193.27	196.70	195.97	198.12	**196.99**
	DECO	223.38	221.76	217.34	227.56	230.94	221.49	**223.75**
	Speedup	2.57×	2.52×	2.52×	2.58×	2.62×	2.56×	**2.56×**
Jetson AGX Orin 64GB	Dense	15.00	15.21	14.93	15.24	15.22	14.94	**15.09**
	TopK	39.28	41.70	38.94	40.03	40.12	40.16	**40.04**
	DECO	44.05	43.99	42.69	45.21	45.92	43.57	**44.24**
	Speedup	2.94×	2.89×	2.86×	2.97×	3.02×	2.92×	**2.93×**

Table 4: Single-GPU decoding speeds on Spec-Bench (token/sec) and average speedup ratios of our acceleration kernel (denoted as "DECO") relative to "Dense" on different devices.

posed acceleration kernel are implemented within FR-Spec (Zhao et al., 2025b), a high-performance, CUDA-optimized inference framework. Both the TopK baseline and our kernel share an identical average activation ratio of 20%.

As detailed in Table 4, our acceleration kernel substantially outperforms the Dense baseline, achieving an average speedup of 2.56× on an RTX 4090 and 2.93× on a Jetson AGX Orin. These results demonstrate that the sparse activation patterns induced by DECO effectively translate into tangible throughput improvements during practical inference. Furthermore, our kernel also outperforms standard TopK under identical sparsity levels. This efficiency advantage stems from the element-wise, in-place execution of the ReLU activation, which incurs a fundamentally lower computational complexity than the vector-wise sorting operations necessitated by TopK.

6 Discussion

Considering the intrinsic activation sparsity of "dense" language models, it is reasonable to expect MoE to be dense-comparable. The property of activation sparsity, indicating that only a small fraction of parameters contribute largely to final outputs, also exists in dense models. Recent studies (Song et al., 2025a; Luo et al., 2024; Zhang et al., 2024) indicate that for each token, only about 30%∼40% of neurons within a standard SwiGLU FFN provide non-negligible contributions. The remaining 60%∼70% of neurons "seem" to occupy computation, but actually contribute little and receive negligible gradient updates from the optimizer due to near-zero SiLU activation values. Actually, dense models can be regarded as a special form of sparse MoE, where the SiLU-activated gating projection functions as a "router", and neurons within up-down projections serve as "experts". Therefore, given an optimized architecture and a sufficient activation ratio, it is possible for a sparse MoE model to match dense performance.

Dense-comparability may be easier to achieve under more heterogeneous training data. In previous experiments, we adopt a diverse data mixture for training, including web texts, code, math, and many other data categories. On such heterogeneous data, DECO matches or exceeds the dense baselines across various parameter scales (Figure 3). However, this "dense-comparability" may be sensitive to the underlying data composition. When trained on FineWeb (Penedo et al., 2024), a less heterogeneous dataset, DECO (Small) has a slightly higher but close PPL compared with Dense (Small) (30.14 vs. 29.76), though exceeding Dense in task accuracy (38.65 vs. 38.03). A reasonable explanation is that high-entropy, multi-domain datasets are inherently better suited for sparse MoE. In such settings, the model can effectively process domain-specific tasks by activating only a specialized subset of its parameters within MoE layers. More studies are needed to verify this explanation.

7 Conclusion

In this work, we propose DECO, a novel MoE architecture designed to minimize computational and storage overhead while maintaining dense-comparable performance. Under an activation ratio of 20%, DECO not only matches the performance of dense models with an equivalent parameter count and training token budget, but also consistently outperforms existing MoE baselines. Our analysis demonstrates that DECO's success stems from a series of architectural refinements, including ReLU-based routing with learnable expert-wise scaling, as well as the integration of non-gated MLP experts utilizing the NormSiLU activation. Furthermore, practical acceleration is achieved on real hardware, confirming that the sparsity of DECO can be translated into tangible efficiency gains for deployment.

Limitations

One potential limitation of this work is that we do not conduct experiments involving the supervised fine-tuning (SFT) or reinforcement learning (RL) stages. Recent studies have indicated that MoE architectures may encounter unique challenges during post-training, such as RL instability resulting from fluctuating router activations (Zheng et al., 2025). Therefore, it is reasonable to assume that DECO may have similar issues. To address this limitation, we are currently training a larger, product-level DECO model optimized for edge-device deployment. This process will involve identifying potential issues during the stages of SFT and RL, and developing corresponding mitigation strategies.

Moreover, as already stated in this article, several hypotheses and empirical observations deserve more extensive verification. Specifically, it remains to be determined how the activation ratio threshold required for dense-comparability scales with model size, and how DECO's intrinsic sparsity and performance fluctuate across diverse data distributions or inference tasks. Investigating these factors is essential to further establish the robustness and theoretical foundation of our proposed architecture.

Ethical Considerations

This work presents DECO, a sparse MoE architecture optimized for end-side devices. By significantly reducing the active parameter scale and memory-access overhead without compromising representation power, our architecture promotes more energy-efficient deployment of LLMs. This efficiency translates to a reduced carbon footprint during inference and helps democratize advanced AI capabilities by making them accessible on resource-constrained edge hardware.

While our architectural improvements are fundamentally algorithmic, we also rely on large-scale web corpora (e.g., FineWeb, the Pile) and will inevitably inherit the historical biases, toxicity, and representational harms present in the pre-training data. We have conducted cleaning and detoxification to the data and used a rule-based program to remove personal information and offensive contents. We encourage researchers and practitioners utilizing our architecture to apply rigorous safety alignment and bias mitigation protocols before releasing downstream models. We use AI agents to refine our writing of this article.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, and 1 others. 2024. Deepseek LLM: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. 2025. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, and 1 others. 2024. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, and 1 others. 2020. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, and 1 others. 2024. MiniCPM: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*.

Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. 2024. Harder tasks need more experts: Dynamic routing in MoE models. *arXiv preprint arXiv:2403.07652*.

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, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Peng Jin, Bo Zhu, Li Yuan, and Shuicheng Yan. 2024. MoE++: Accelerating mixture-of-experts methods with zero-computation experts. *arXiv preprint arXiv:2410.07348*.

Jakub Krajewski, Jan Ludziejewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, and 1 others. 2024. Scaling laws for fine-grained mixture of experts. *arXiv preprint arXiv:2402.07871*.

Houyi Li, Ka Man Lo, Ziqi Wang, Zili Wang, Wenzhen Zheng, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. 2025. Can mixture-of-experts surpass dense llms under strictly equal resources? *arXiv preprint arXiv:2506.12119*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*.

Yuqi Luo, Chenyang Song, Xu Han, Yingfa Chen, Chaojun Xiao, Zhiyuan Liu, and Maosong Sun. 2024. Sparsing Law: Towards large language models with greater activation sparsity. *arXiv preprint arXiv:2411.02335*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, and 1 others. 2024. The FineWeb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849.

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

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. SocialIQA: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473.

Noam Shazeer. 2020. GLU variants improve Transformer. *arXiv preprint arXiv:2002.05202*.

Chenyang Song, Xu Han, Zhengyan Zhang, Shengding Hu, Xiyu Shi, Kuai Li, Chen Chen, Zhiyuan Liu, Guangli Li, Tao Yang, and Maosong Sun. 2025a. ProSparse: Introducing and enhancing intrinsic activation sparsity within large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 2626–2644.

Chenyang Song, Weilin Zhao, Xu Han, Chaojun Xiao, Yingfa Chen, Yuxuan Li, Zhiyuan Liu, and Maosong Sun. 2025b. BlockFFN: Towards end-side acceleration-friendly mixture-of-experts with chunk-level activation sparsity. *arXiv preprint arXiv:2507.08771*.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2025. Nemotron-CC: Transforming common crawl into a refined long-horizon pretraining dataset. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2459–2475.

Vijay Thakkar, Pradeep Ramani, Cris Cecka, Aniket Shivam, Honghao Lu, Ethan Yan, Jack Kosaian, Mark Hoemmen, Haicheng Wu, Andrew Kerr, Matt Nicely, Duane Merrill, Dustyn Blasig, Fengqi Qiao, Piotr Majcher, Paul Springer, Markus Hohnerbach, Jin Wang, and Manish Gupta. 2023. CUTLASS.

Changxin Tian, Kunlong Chen, Jia Liu, Ziqi Liu, Zhiqiang Zhang, and Jun Zhou. 2025. Towards greater leverage: Scaling laws for efficient mixture-of-experts language models. *arXiv preprint arXiv:2507.17702*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. LLaMA 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Ziteng Wang, Jianfei Chen, and Jun Zhu. 2024b. ReMoE: Fully differentiable mixture-of-experts with ReLU routing. *arXiv preprint arXiv:2412.14711*.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. 2024. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 7655–7671.

Feng Yao, Junxia Cui, Ruohan Zhang, Liyuan Liu, Shibo Hao, Li Zhang, Chengyu Dong, Shuohang Wang, Jianfeng Gao, Jingbo Shang, and 1 others. 2025. DenseMixer: Improving moe post-training with precise router gradient. In *NeurIPS 2025 Workshop on Structured Probabilistic Inference {\&} Generative Modeling*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Zhengyan Zhang, Yixin Song, Guanghui Yu, Xu Han, Yankai Lin, Chaojun Xiao, Chenyang Song, Zhiyuan Liu, Zeyu Mi, and Maosong Sun. 2024. ReLU^2 wins: Discovering efficient activation functions for sparse LLMs. *arXiv preprint arXiv:2402.03804*.

Guoliang Zhao, Yuhan Fu, Shuaipeng Li, Xingwu Sun, Ruobing Xie, An Wang, Weidong Han, Zhen Yang, Weixuan Sun, Yudong Zhang, and 1 others. 2025a. Towards a comprehensive scaling law of mixture-of-experts. *arXiv preprint arXiv:2509.23678*.

Weilin Zhao, Tengyu Pan, Xu Han, Yudi Zhang, Ao Sun, Yuxiang Huang, Kaihuo Zhang, Weilun Zhao, Yuxuan Li, Jianyong Wang, and 1 others. 2025b. FR-Spec: Accelerating large-vocabulary language models via frequency-ranked speculative sampling. *arXiv preprint arXiv:2502.14856*.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, and 1 others. 2025. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

A Effect of Key MoE Hyperparameters

Compared with dense architectures, MoE models introduce unique hyperparameters that significantly influence performance, among which the activation ratio, expert granularity, and shared expert size are often considered the most important ones (Tian et al., 2025). Similarly, the performance of DECO is also sensitive to these factors, and the dense-comparability of DECO holds only under certain conditions. We have already studied the effect of the routed-expert activation ratio in Section 4.5, which has a significant impact on MoE performance. In this section, we characterize the impact of the remaining two MoE-specific factors.

A.1 Shared Expert Size

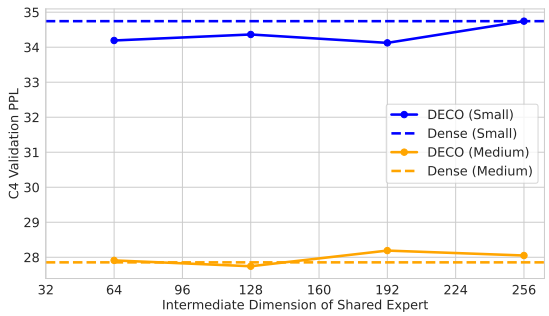

Figure 9: The impact of shared expert sizes on performance. Routed expert dimension is 64.

We evaluate the performance of DECO across various shared-expert sizes, proportional to the intermediate dimension of the shared expert. For rigorous comparison, we hold the total parameter count approximately constant. This necessitates a trade-off between the capacity of the shared expert and the number of routed experts.

As illustrated in Figure 9, when the intermediate dimension of the shared expert is around 1∼2 times that of a routed expert (fixed at 64), DECO achieves comparability with the dense baseline and exhibits relative insensitivity to the size of the shared expert. However, as the shared-expert dimension increases to 3∼4 times that of the routed experts, the perplexity (PPL) degrades significantly. This performance drop is attributed to the reduced number of routed experts necessitated by the fixed parameter budget. These findings corroborate the observations in Tian et al. (2025), suggesting that an oversized shared expert is unnecessary. Instead, a single shared expert with a scale comparable to that of a routed expert appears to be optimal.

A.2 Expert Granularity

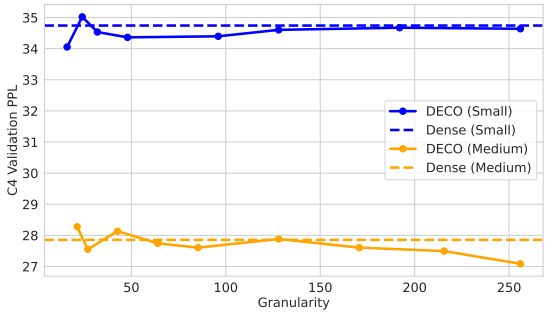

Figure 10: The impact of the expert granularity ($g = 4d_h/d_e$) on performance of DECO.

As illustrated in Figure 10, for "Medium" setting, we observe a monotonic improvement in perplexity (PPL) as granularity increases when $g > 120$. However, the "Small" setting shows reduced sensitivity to granularity, as it is potentially more difficult for its lower-capacity experts to capture the "long-tail" distribution of the data or to develop fine-grained specialization. For both settings, when granularity is below 60, performance fluctuates near the dense baseline. By contrast, for $g > 60$, DECO consistently achieves dense parity. These results suggest that finer expert granularity is essential for stable and competitive performance.

B Detailed Experimental Settings

Training datasets For all settings, we use the same pretraining dataset, which consists of a mixture of various data, including FineWeb (Penedo et al., 2024), Nemotron-CC (Su et al., 2025), Pile (Gao et al., 2020), Wikipedia, and many other collected raw corpora. The training data covers a wide range of categories, such as raw web text, math, and code. The data mixing ratio is carefully tuned through experiments on small-scale models.

Evaluation benchmarks We evaluate the accuracy of models on various benchmarks with LMEval (Gao et al., 2024), including PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), ARC-C, ARC-E (Clark et al., 2018), WinoGrande (Sakaguchi et al., 2020), and LAMBADA (Paperno et al., 2016). The evaluation parameters follow the default dataset configurations provided by LMEval. All dataset usage is restricted to academic intents, which does not violate the terms of use and is consistent with their intended use. The languages covered mainly include English and Chinese.

Scale	d_h	d_e	d_s	N_e	d_{ff}	N_{layer}	n_{step}	lr	$batch_size$	n_{tokens}	η	λ_{init}
0.11B (Small)	768	64	128	42	1,920	16	15,000	$1.175e-3$	2.62×10^5	3.93×10^9	1.002	$1e-8$
0.24B (Medium)	1,024	64	128	57	2,560	20	15,000	$9.3e-4$	5.24×10^5	7.86×10^9	1.002	$1e-8$
0.53B (Large)	1,280	64	128	77	3,360	27	20,000	$7.8e-4$	1.05×10^6	2.10×10^{10}	1.001	$1e-8$
1.18B (XLarge)	1,792	64	128	102	4,480	32	23,000	$6.54e-4$	2.10×10^6	4.83×10^{10}	1.001	$1e-8$

Table 5: The hyperparameter settings of models. d_s, d_{ff}, N_{layer}, n_{step}, and n_{tokens} denote the intermediate dimension of the shared expert, the intermediate dimension of the dense FFN layer, the total number of hidden layers, the number of training steps, and the number of training tokens, respectively. η and λ_{init} are the coefficient multiplier and the initial value of the regularization coefficient as used in the adaptive sparsity regularization.

Baseline adjustments Since BlockFFN is originally designed to combine activation sparsity and speculative decoding for better acceleration, it introduces the chunk sparsification loss and activation locality loss to increase the union sparsity level of consecutive tokens as well as the activation similarity between neighboring tokens (Song et al., 2025b). However, as we do not target these objectives in this work, for fair comparison, we replace its original regularization with our own adaptive sparsity regularization.

Setting	PPL (↓)	Task (↑)
Small (two-loss)	35.62	36.23
Small (one-loss)	**35.27**	**36.53**
Medium (two-loss)	28.65	38.18
Medium (one-loss)	**28.16**	**38.44**

Table 6: The performance of BlockFFN using its original regularization (two-loss) and our own adaptive sparsity regularization (one-loss).

To justify such an adjustment, we implement BlockFFN strictly using its original "two-loss" setting, tuning the loss coefficients via grid search. From the results in Table 6, at the same target activation ratio of 20%, our "one-loss" setting is consistently better than the original "two-loss" regularization, demonstrating the rationality of our experimental setting for BlockFFN.

Hyperparameters To find the optimal hyperparameters, we first conduct a grid search on small-scale models. Then, following DeepSeek LLM (Bi et al., 2024), we assume a power-law relationship of both the optimal learning rate and batch size with respect to the computation budget. Thereby, we can extrapolate the hyperparameters for large-scale models using those for small-scale models. The detailed experimental hyperparameters are shown in Table 5. All MoE baseline settings and DECO use a dense FFN at the first layer, and adopt sparse MoE for the remaining $N_{layer} - 1$ layers. We use the WSD learning rate scheduler along the training process (Hu et al., 2024; Dubey et al., 2024), with 100 warmup steps at the beginning and 1,000 decay steps at the last stage. Our models are trained on 8∼16 NVIDIA A800 GPUs for 2 ∼ 64 hours, depending on the model scale.

C Impact of the Scaling Factor Initialization Value

Init Value	0.01	0.025	0.05	0.1	0.25	0.5	1.0
PPL (↓)	34.71	35.66	34.61	34.36	**34.26**	35.20	34.74
Task (↑)	36.86	36.57	36.64	**36.90**	36.72	36.57	36.54

Table 7: The performance of DECO (Small) when the learnable expert-wise scaling factors are initialized with different values.

In this section, we conduct experiments on DECO when its learnable expert-wise scaling factors are initialized with different values. The results are shown in Table 7. The performance of DECO does not change monotonically. Empirically, the best performance is achieved when the scaling factors are initialized around 0.1∼0.25. In our experiments in Section 4, we choose the initialization value of 0.1 for all DECO settings.

D Repeated Experiments

Due to the prohibitive GPU costs associated with large-scale pre-training, experiments in this domain are typically executed only once. To rigorously validate the reliability of our findings, we conducted repeated experiments at the "Small" and "Medium" scales, performing five independent runs with different random seeds for each setting. As detailed in Table 8, DECO demonstrates highly stable and competitive performance. The narrow standard deviations confirm that DECO does not suffer from high variance across runs. Furthermore, DECO not only tightly matches the Dense baseline on the "Small" scale, but it actually slightly outperforms it on the "Medium" scale, achieving a lower average

Setting	5-run PPL (↓)	Average PPL (↓)	5-run Task (↑)	Average Task (↑)
Dense (Small)	34.40 34.28 34.49 34.59 34.38	**34.43±0.105**	36.80 36.93 36.71 36.67 36.81	36.78±0.090
DECO (Small)	34.36 34.51 34.90 34.91 34.75	34.69±0.218	36.90 36.78 36.73 37.15 36.65	**36.84±0.174**
Dense (Medium)	27.85 27.77 27.15 27.79 28.07	27.73±0.307	39.01 38.85 39.14 38.96 38.93	38.98±0.096
DECO (Medium)	27.74 27.17 27.71 27.22 27.81	**27.53±0.276**	39.18 39.35 38.91 39.37 38.85	**39.13±0.217**

Table 8: The repeated experimental results on the "Small" and "Medium" scales.

perplexity (27.53 ± 0.276) and a higher average task score (39.13 ± 0.217). Together, these robust empirical results confirm that DECO consistently maintains comparable and occasionally superior performance to the Dense setting.

E Theoretical Justification for NormSiLU

To theoretically demonstrate the rationality of NormSiLU, we consider the post-activation output of all experts. Let $\mathbf{W}_{up} \in \mathbb{R}^{d_h \times (N_e d_e)}$ denote the concatenated up-projection weights of N_e experts, and let $\mathbf{x} \in \mathbb{R}^{d_h}$ be the input hidden state. The post-activation intermediate state is:

$$\mathbf{y} = \mathrm{SiLU}(\mathrm{Norm}(\mathbf{z})),\ \mathbf{z} = \mathbf{W}_{up}^T \mathbf{x} \in \mathbb{R}^{N_e d_e}. \quad (4)$$

For simplicity, we first assume that the operator "Norm" is implemented as a vanilla layer normalization (Ba et al., 2016) across the entire concatenated expert dimension. Let $\mathbf{g} = \nabla_{\mathbf{y}} \mathcal{L}$ be the upstream gradient from the language modeling loss $\mathcal{L}$. The gradient with respect to the weights $\mathbf{W}_{up}$ is formulated as:

$$\begin{aligned} &\nabla_{\mathbf{W}_{up}} \mathcal{L} = \mathbf{x}(\nabla_{\mathbf{z}} \mathcal{L})^T = \mathbf{x}(\mathbf{J}_{norm}^T \mathbf{D}_{silu} \mathbf{g})^T, \\ &\mathbf{u} = \mathrm{Norm}(\mathbf{z}) = \frac{\mathbf{z_0}}{||\mathbf{z}_0||_{rms}},\ \mathbf{z}_0 = \mathbf{z} - \bar{\mathbf{z}}, \end{aligned} \quad (5)$$

where $\mathbf{D}_{silu} = \mathrm{diag}(\sigma(\mathbf{u}) + \mathbf{u} \odot \sigma(\mathbf{u}) \odot (1 - \sigma(\mathbf{u})))$ is the diagonal Jacobian matrix of SiLU, σ is the sigmoid function, and $\bar{\mathbf{z}}$ denotes the mean of $\mathbf{z}$. Letting $n = N_e d_e$ be the total dimension of $\mathbf{z}$ and $\mathbf{1}$ be the all-ones vector, the Jacobian matrix of the layer normalization $\mathbf{J}_{norm}$ is:

$$\begin{aligned} \mathbf{J}_{norm} &= \frac{\partial \mathbf{u}}{\partial \mathbf{z}} \\ &= \frac{1}{||\mathbf{z}_0||_{rms}} \left(\mathbf{I}_n - \frac{1}{n} \mathbf{1}\mathbf{1}^T - \frac{\mathbf{z}_0 \mathbf{z}_0^T}{n ||\mathbf{z}_0||_{rms}^2} \right). \end{aligned} \quad (6)$$

Since the matrix inside the parentheses is an affine shift of a projection matrix, its spectral norm $||\mathbf{J}_{norm}||_2$ is strictly bounded by $1/||\mathbf{z}_0||_{rms}$. Furthermore, assuming the elements of $\mathbf{W}_{up}$ follow a zero-centered i.i.d. normal distribution, we can statistically approximate the RMS norm as:

$$||\mathbf{z}_0||_{rms} \approx \frac{1}{\sqrt{d_h n}} ||\mathbf{W}_{up}||_F \cdot ||\mathbf{x}||_2. \quad (7)$$

Applying the sub-multiplicative property of matrix norms, the Frobenius norm of the gradient is bounded by:

$$\begin{aligned} &||\nabla_{\mathbf{W}_{up}} \mathcal{L}||_F \\ &\leq ||\mathbf{x}||_2 \cdot ||\mathbf{J}_{norm}||_2 \cdot ||\mathbf{D}_{silu}||_2 \cdot ||\mathbf{g}||_2 \quad (8) \\ &\leq ||\mathbf{g}||_2 \cdot \mathcal{O}(||\mathbf{W}_{up}||_F^{-1}). \end{aligned}$$

Therefore, as long as $\mathbf{W}_{up}$ is initialized with a proper Frobenius norm, this normalization theoretically guarantees that the gradient scale remains bounded and invariant to the input magnitude, preventing gradient explosion.

However, the above paradigm possesses a critical systems-level flaw: computing a global layer normalization requires the explicit materialization of $\mathbf{z}$ across *all* experts. This inherently violates the core sparsity principle of MoE, as it forces the computation of inactive experts.

To resolve this bottleneck, our final implementation of NormSiLU decouples the operation into a dual-stage mechanism: an inter-expert mean normalization and an intra-expert RMS normalization. During inference, the global inter-expert averaging operator mathematically reduces to: $\mathrm{Avg}(\mathbf{W}_{up}^T \mathbf{x}) = \bar{\mathbf{w}}_{up}^T \mathbf{x}$, where $\bar{\mathbf{w}}_{up} \in \mathbb{R}^{d_h \times d_e}$ is the fixed average up-projection weight. By contrast, the RMS normalization is strategically restricted to the intra-expert dimension (d_e) and applied *only* to activated experts. This dual-stage design preserves the theoretical gradient-bounding stability while strictly adhering to the computational constraints of sparse inference.

F Evaluation Results on Individual Benchmarks

In this section, we provide the evaluation results of DECO and baselines on each individual benchmark. The results of "Small", "Medium", "Large", and "XLarge" settings are shown in Table 9, Table 10, Table 11, and Table 12, respectively.

	PIQA	SIQA	HellaSwag	ARC-E	ARC-C	WinoGrande	LAMBADA	**Avg.**
TopP	60.17	35.21	27.67	42.38	18.86	50.91	7.74	34.71
DeepSeek-V3 (GA)	60.12	35.93	27.85	44.15	19.45	50.75	15.89	36.31
DeepSeek-V3 (NG)	60.66	36.63	27.76	45.76	20.05	50.43	15.85	36.73
ReMoE (GA)	60.34	36.59	27.72	46.34	18.43	51.22	15.60	36.61
ReMoE (NG)	59.74	35.16	27.96	46.09	19.54	49.96	17.76	36.60
BlockFFN	60.83	36.28	27.62	44.87	19.45	50.12	16.55	36.53
DECO (GA)	58.22	35.93	27.65	42.51	19.03	51.30	15.84	35.78
Dense	60.77	35.57	28.03	45.88	20.22	50.59	16.51	**36.80**
DECO (NG)	60.94	36.85	27.96	45.84	20.22	50.67	15.80	**36.90**

Table 9: The evaluation scores of "Small" settings on individual benchmarks.

	PIQA	SIQA	HellaSwag	ARC-E	ARC-C	WinoGrande	LAMBADA	**Avg.**
TopP	61.37	36.49	28.57	45.75	19.03	50.51	12.63	36.34
DeepSeek-V3 (GA)	62.79	37.87	29.61	48.53	20.90	52.17	19.84	38.82
DeepSeek-V3 (NG)	63.28	37.67	29.64	49.07	20.65	49.25	19.76	38.47
ReMoE (GA)	62.62	37.67	29.10	48.65	20.05	50.67	18.28	38.15
ReMoE (NG)	63.22	36.64	29.40	49.71	19.71	50.83	20.45	38.57
BlockFFN	63.38	35.82	29.61	48.82	20.99	50.51	19.95	38.44
DECO (GA)	61.86	37.10	28.44	47.18	19.37	49.64	17.43	37.29
Dense	63.00	36.90	29.57	51.47	22.01	50.51	19.62	**39.01**
DECO (NG)	64.36	36.18	29.47	51.98	20.73	52.09	19.48	**39.18**

Table 10: The evaluation scores of "Medium" settings on individual benchmarks.

	PIQA	SIQA	HellaSwag	ARC-E	ARC-C	WinoGrande	LAMBADA	**Avg.**
TopP	64.53	36.44	31.42	50.55	22.53	51.07	14.13	38.67
DeepSeek-V3 (GA)	66.32	38.38	32.71	55.18	24.91	51.07	28.06	42.38
DeepSeek-V3 (NG)	65.94	37.87	32.75	54.67	23.38	52.49	27.03	42.02
ReMoE (GA)	66.05	38.69	32.08	55.26	24.32	52.33	27.19	42.27
ReMoE (NG)	66.92	38.79	32.66	55.93	24.57	51.41	27.10	42.48
BlockFFN	66.10	38.59	32.51	56.36	24.15	51.38	27.30	42.34
DECO (GA)	66.14	38.74	32.69	56.41	23.72	51.78	26.57	42.29
Dense	66.87	37.77	32.92	56.27	24.74	51.70	29.34	**42.80**
DECO (NG)	65.94	39.61	32.78	57.11	25.43	53.28	25.50	**42.81**

Table 11: The evaluation scores of "Large" settings on individual benchmarks.

	PIQA	SIQA	HellaSwag	ARC-E	ARC-C	WinoGrande	LAMBADA	**Avg.**
TopP	66.16	37.36	34.10	57.32	22.87	49.41	20.49	41.10
DeepSeek-V3 (GA)	69.53	39.92	36.65	62.21	28.41	53.28	34.52	46.36
ReMoE (NG)	68.93	40.07	37.43	61.20	28.07	53.43	35.46	46.37
BlockFFN	70.02	40.28	36.81	61.91	28.50	55.80	34.60	46.85
Dense	70.46	40.69	37.33	63.38	28.58	52.72	35.73	**46.98**
DECO (NG)	70.24	40.89	37.42	62.96	29.69	54.93	35.53	**47.38**

Table 12: The evaluation scores of "XLarge" settings on individual benchmarks.