FineRMoE: Dimension Expansion for Finer-Grained Expert with an Upcycling Approach

Ning Liao, Xiaoxing Wang†, Xiaohan Qin, Junchi Yan†
Shanghai Jiao Tong University
liaoning@sjtu.edu.cn
† Corresponding Authors

Code **Model**

Abstract

As revealed by the scaling law of Mixture-of-Experts models, model performance ceases improving once the granularity of the intermediate dimension exceeds the optimal threshold, limiting further gains from single-dimension fine-grained design. To address this bottleneck, we propose FineRMoE (FineR-Grained MoE), an architecture that extends fine-grained expert design to both intermediate and output dimensions, aiming to enhance expert specialization beyond the single-dimension limit. We further introduce a bi-level sparse forward computation paradigm and a specialized routing mechanism to govern the activation. In addition, to obviate the prohibitive cost of pre-training from scratch, we devise a generalized upcycling method to build FineRMoE in a cost-effective manner. Extensive experiments performed under the upcycling paradigm demonstrate the superior performance achieved by FineRMoE across ten standard benchmarks. Compared with the strongest upcycling baseline method, FineRMoE achieves significant parameter and inference efficiency gains.

1 Introduction

Mixture-of-Experts (MoE) has emerged as the prevailing architecture of Large Language Models (LLMs) [44, 3, 7]. By configuring the experts a markedly smaller intermediate size than that of the Feed-Forward Network (FFN), fine-grained expert design [19, 33] has been widely adopted for mitigating redundancy and improving specialization. As uncovered by the scaling law pertaining to fine-grained MoE [21, 34, 42, 32], the performance of MoE models scales positively with expert granularity in the intermediate dimension within a valid parameter regime, whereas a decline in performance is observed once expert granularity surpasses the optimal threshold. To break through this fundamental scaling limit of fine-grained MoE that is bound to the intermediate dimension alone, we investigate the feasibility of extending fine-grained expert design to additional dimensions, aiming to unlock further performance gains for MoE models beyond the scope of the original scaling law.

Rethinking the multi-head attention (MHA) [35], reducing the output dimension of QKV transformations drives heads toward distinct feature extraction. Analogously, reducing the output dimension of experts would encourage independent representations, which in turn suppresses redundancy and enhances specialization of experts, as will be verified in Fig. 4. Motivated by the preceding analysis, we propose the FineR-grained MoE (**FineRMoE**) architecture that generalizes fine-grained design beyond the intermediate dimension to the output dimension. To achieve flexible adjustment of the granularity of sparse experts and model scale, we define four hyper-parameters, i.e., granularity and expansion rate at the intermediate and output dimensions, for the joint architecture design.

Regarding the forward process of the sparse experts, existing MoE models primarily rely on the weighted sum for multi-expert fusion. It would disrupt the dimensional consistency of the computations before and after MoE layers, if the expert is fine-grained at the output dimension. We introduce

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

a novel bi-level sparse forward paradigm consisting of a sparse concatenation layer and a sparse sum layer as shown Fig. 1. Specifically, for each token, its output with restored dimension from the sparse experts is obtained in the sparse concatenation layer by concatenating selected dimension-reduced vectors. In the sparse sum layer, each of these vectors is computed as the weighted sum of outputs from sparsely activated finer-grained experts within its corresponding MoE group. Additionally, notwithstanding the sparsity at both the expert and vector levels inherent in the FineRMoE, we forgo maintaining two distinct routers for each sparse layer. Instead, a specialized routing mechanism is designed that employs only a single router network to simultaneously trigger expert activation in the sparse sum layer and candidate vector selection in the sparse concatenation layer, thereby avoiding conflict activation and reducing parameter cost associated with using two separate routers.

Despite the overwhelming advantages of MoE models, pre-training them from scratch remains prohibitively expensive due to the extensive computational budgets and large-scale, high-quality data required. To facilitate the efficient construction and training of MoE models, the upcycling [18, 15] paradigm has recently emerged. By leveraging a pre-trained dense LLM, upcycling converts the FFNs into MoE layers to avoid training experts from random initialization. Current upcycling methods are tailored to single-layer MoE models that fuse outputs from experts via weighted sum. They usually construct experts by duplicating the FFNs [17, 45] or partitioning them along the intermediate dimension [47, 11]. Consequently, these approaches are inapplicable to the proposed FineRMoE architecture with fine-grained design across both intermediate and output dimensions.

Based on the foregoing investigation, we contend that mainstream training-free upcycling methods can be unified under a single protocol, with the exception of methods requiring training for expert induction [29, 45]. To accomplish FineRMoE without training from scratch, we devise a novel upcycling method to instantiate finer-grained experts. By leveraging the four hyper-parameters defined in the FineRMoE, our upcycling develops a configurable mechanism for expert construction. It enables flexible partitioning and expansion of the pre-trained FFN along both its intermediate and output dimensions, thus applicable to both FineRMoE and conventional MoE architectures.

In this work, we primarily focus on the implementation approaches of upcycling. Experimentally, we build the FineRMoE based on the Qwen2.5 [41] with sizes of 0.5B, 1.5B, and 7B, by leveraging our upcycling method. Following extended training on 50B tokens, the resultant FineRMoE, equipped with 128 total experts and 2 activated experts, outperforms carefully curated baselines across ten benchmarks. Meanwhile, compared with the strongest upcycling baseline method, FineRMoE delivers **6× higher parameter efficiency, 281× lower prefill latency, and 136× higher decoding throughput** during inference. Our contributions include:

(1) We propose the FineRMoE architecture. To break the scaling bottleneck of MoE that confines expert granularity to the intermediate dimension alone, FineRMoE for the first time extends fine-grained expert design from the intermediate dimension to the output dimension. It introduces a bi-level sparse forward computation paradigm consisting of a sparse concatenation layer and a sparse sum layer to process the input tokens following a dimension reduction-then-restoration order.

(2) We introduce a specialized router mechanism. Despite the inherent bi-level sparsity in the FineRMoE, the routing mechanism employs only a single router network to govern the activation in both sparse layers, promoting the consistent activations and reducing parameter cost.

(3) We develop a generalized upcycling method. The method enables efficient expert construction by flexibly partitioning and expanding the pre-trained FFN along both intermediate and output dimensions. It is generally applicable to both FineRMoE and conventional MoE architectures.

(4) We provide extensive validation experiments. Building the FineRMoE with the proposed upcycling method achieves superior performance across ten benchmarks, along with remarkable efficiency on both parameters and inference. Ablation studies validate the effectiveness of FineRMoE.

2 Related Work

Mixture-of-Experts (MoE). It was initially proposed [13, 16] to scale model capacity while curb computational overhead [22, 4], rendering it a prevalent building block in contemporary LLMs [31, 38, 40, 23, 37, 20]. Earlier MoE models [9, 14] favor larger intermediate dimension to bolster per-expert capacity. Whereas LLMs [42, 32] released recently have embraced fine-grained experts [2] with lower intermediate dimension, which reduces redundancy and improves expert specialization [8]. Nonetheless, existing fine-grained MoE models confine this design to the intermediate dimension.

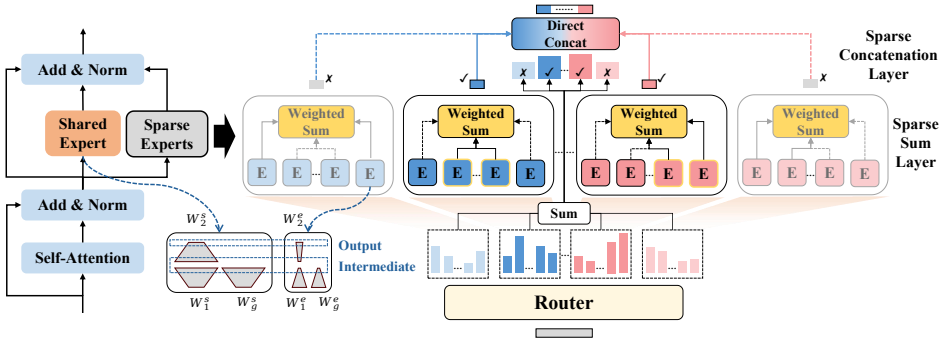


Figure 1: The architecture of FineRMoE. Unlike conventional MoEs that confine fine-grained design solely to the intermediate dimension, FineRMoE for the first time jointly expands it to the output dimension, enabling a bi-level sparse computation paradigm. It comprises a Sparse Sum Layer, which produces multiple dimension-reduced candidate vectors, and a Sparse Concatenation Layer, which selectively assembles these vectors to restore the output dimension. A single router with specially designed routing mechanism simultaneously steers the activation in the two sparse layers.

Empirical studies on the scaling law of fine-grained MoE [21, 34] demonstrate that, within a rational scope, higher expert granularity contributes to better performance; nevertheless, model performance tends to deteriorate when expert granularity goes beyond its optimal point. To this end, we aim to extend it to the output dimension of each expert for further specialization. Similarly, MH-MoE [39] is inspired by MHA to enhance granular understanding but emphasizes token partitioning, neglecting fine-grained expert design even in the intermediate dimension. In contrast, we devote our attention to the fine-grained expert design regardless of token splitting.

Upcycling. To circumvent the prohibitive computational and data demands of training MoE models from scratch, upcycling methods [24, 45, 29] have recently emerged. The vast majority of upcycling techniques instantiate experts via training-free strategies. One line of methods [17, 36] initialize experts by replicating pre-trained FFNs, while another line of work [47, 11] partitions the FFNs along the intermediate dimension to yield multiple fine-grained experts. Current upcycling approaches fall short of supporting the proposed FineRMoE architecture. To bridge this gap, we propose an upcycling method that enables the construction of FineRMoE, while remaining fully compatible with the two types of prevalent training-free expert construction methods.

3 Method

3.1 FineRMoE Architecture

As depicted in the Fig. 1 left, the FineRMoE architecture consists of the shared expert and the sparse finer-grained experts, outputs of which are summed directly for later process. Each expert from the two types includes 3 weight matrices: the up projection weight $\boldsymbol{W}_1$, the gate weight $\boldsymbol{W}_g$, and the down projection weight $\boldsymbol{W}_2$. Given the LLM with hidden dimension as h and an input $\boldsymbol{x} \in \mathbb{R}^h$, the output $\boldsymbol{y} \in \mathbb{R}^h$ of each expert is calculated as:

$$\boldsymbol{y}_i = \boldsymbol{x}\boldsymbol{W}_1 \odot \mathrm{SiLU}(\boldsymbol{x}\boldsymbol{W}_g), \quad \boldsymbol{y} = \boldsymbol{y}_i\boldsymbol{W}_2. \tag{1}$$

Denoting the intermediate size of the shared expert as H, the shared expert is thus composed of $\boldsymbol{W}_1^s \in \mathbb{R}^{h\times H}$, $\boldsymbol{W}_g^s \in \mathbb{R}^{h\times H}$, and $\boldsymbol{W}_2^s \in \mathbb{R}^{H\times h}$.

Setting the shared expert as a reference, sparse finer-grained experts are materialized through 4 hyper-parameters. For clarity, we introduce them in a top-down manner as shown in the Fig. 1 right, which includes the sparse concatenation layer and the sparse sum layer.

In the **sparse concatenation layer**, we define the *output granularity* G_O measuring how many times larger is the hidden dimension h of the LLM as compared to the output dimension h_e of the sparse experts, which is calculated as:

$$G_O = h/h_e. \tag{2}$$

The *output expansion rate* R_O is defined as the number of candidate dimension-reduced vectors to be selected from for each concatenation component. Therefore, the output $\boldsymbol{O} \in \mathbb{R}^h$ of this layer is the

Algorithm 1 Pseudocode of the Router Mechanism in a Pytorch-like style.

```
# x: input    L: number of tokens in the input    N: number of experts
# G_O: output granularity    R_O: output expansion rate
# G_I: intermediate granularity    R_I: intermediate expansion rate
# T_I: number of activated experts in each group in the sparse sum layer
# Calculate the router score
score = Router(x)                                                    # L × N
# Calculate the expert mask concerning the expert activation within each group in the sparse sum layer
group_score = score.view(L, G_O R_O, G_I R_I)                        # L × G_O R_O × G_I R_I
_, group_act = TopK(group_score, T_I, dim=-1)                        # L × G_O R_O × T_I
sum_mask = zeros(L, G_O R_O, G_I R_I)                                # L × G_O R_O × G_I R_I
sum_mask.scatter(-1, group_act, True)                                # L × G_O R_O × G_I R_I
# Calculate the expert mask concerning the candidate vector selection in the sparse concatenation layer
cc_score = sum(group_score, dim=-1)                                  # L × G_O R_O
cc_score = cc_score.view(L, G_O, -1)                                 # L × G_O × R_O
cc_act = argmax(cc_score, dim=-1, keepdim=True)                      # L × G_O × 1
cc_mask = zeros(L, G_O, R_O).scatter(-1, cc_act, True)               # L × G_O × R_O
cc_mask = cc_mask.view(L, -1).unsqueeze(-1)                          # L × G_O R_O × 1
cc_mask = cc_mask.repeat(1,1,G_I R_I)                                # L × G_O R_O × G_I R_I
# Element-wise AND operation on the two masks
final_mask = sum_mask & cc_mask                                      # L × G_O R_O × G_I R_I
final_mask = final_mask.view(L, -1)                                  # L × N
# Calculate the final scores and indices of activated experts
final_score = score.masked_fill(~final_mask.bool(), float(-inf))
probs, indices = TopK(final_score, G_O T_I, dim=-1)                  # L × G_O T_I
```

concatenation of G_O components $\boldsymbol{O}_i \in \mathbb{R}^{h_e}$, and each component is selected from R_O candidate vectors $\boldsymbol{O}_i^j \in \mathbb{R}^{h_e}$, which is the output of its corresponding group of experts in the sparse sum layer. Therefore, the concatenation process is formulated as:

$$\begin{aligned} \boldsymbol{O} = \mathrm{Concat}(\boldsymbol{O}_0, ..., \boldsymbol{O}_i, ..., \boldsymbol{O}_{G_O-1}), \quad \boldsymbol{O}_i = \mathrm{Top1Select}(\boldsymbol{O}_i^0, ...\boldsymbol{O}_i^j, ..., \boldsymbol{O}_i^{R_O-1}), \\ \boldsymbol{O}_i^j \in \mathbb{R}^{h_e}, \quad i \in [0, G_O - 1], j \in [0, R_O - 1]. \end{aligned} \tag{3}$$

The $\mathrm{Top1Select}$ chooses the candidate vector with the highest corresponding router score as the concatenation component of the output, as detailed in Eq. 8 in the Sec. 3.2.

In the **sparse sum layer**, the experts are divided into multiple groups with each group consisting of N_g finer-grained experts, the sparse weighted sum of which is the candidate vector $\boldsymbol{O}_i^j$ in the sparse concatenation layer. We define the *intermediate granularity* G_I as measuring how many times larger is the intermediate size H of the shared expert as compared to the intermediate size H_e of sparse experts. The *intermediate expansion rate* R_I is defined as how many times larger is the total sum of the intermediate size of sparse experts in a group as compared to the intermediate size of the shared expert. The definitions of these two hyper-parameters are as below:

$$G_I = H/H_e, \quad R_I = (N_g \cdot H_e)/H. \tag{4}$$

As each candidate vector in the sparse concatenation layer corresponds to a group of experts in the sparse sum layer, the total number of sparse experts N is calculated as:

$$N = G_O \cdot R_O \cdot N_g = G_O \cdot R_O \cdot G_I \cdot R_I. \tag{5}$$

Each finer-grained expert $\boldsymbol{E}_k, k \in [0, N-1]$ is thus composed of the up projection weight $\boldsymbol{W}_{1k}^e \in \mathbb{R}^{h \times H_e}$, the gate weight $\boldsymbol{W}_{gk}^e \in \mathbb{R}^{h \times H_e}$, and the down projection weight $\boldsymbol{W}_{2k}^e \in \mathbb{R}^{H_e \times h_e}$. According to the above, the candidate vector $\boldsymbol{O}_i^j$ in Eq. 3 is calculated as:

$$\begin{aligned} \boldsymbol{O}_i^j = \mathrm{WeightedSum}(\boldsymbol{E}_k, \boldsymbol{x}), i \in [0, G_O - 1], j \in [0, R_O - 1], \\ k \in [i \cdot G_I \cdot R_I \cdot R_O + j \cdot G_I \cdot R_I, i \cdot G_I \cdot R_I \cdot R_O + (j+1) \cdot G_I \cdot R_I - 1]. \end{aligned} \tag{6}$$

The $\mathrm{WeightedSum}$ is conducted based on the router score corresponding to the sparse experts in each group, which will be detailed in Eq. 7 in Sec. 3.2.

3.2 Router Mechanism

Despite the intrinsic sparsity in the two layers of FineRMoE, we eschew the deployment of two distinct routers which will cause conflict activations analyzed in Sec. 4.3. Instead, as presented in Algorithm 1, the mechanism with only a single router is devised to simultaneously select the dimension-reduced vectors in the sparse concatenation layer and activate the experts within each group in the sparse sum layer. Specifically, after calculating the initial `score` by the single router in Line 5–6, the mechanism computes the activation mask over sparse experts from two perspectives.

The first perspective corresponds to Line 7–11 computes the mask for expert activation within each group in the sparse sum layer. By dividing the initial `score` into $G_O R_O$ groups with each group containing $G_I R_I$ elements, T_I experts with higher scores will be activated for the weighted sum and produce the candidate vector. Therefore, within a group of experts with indices l ranging from 0 to $G_I R_I - 1$, the $\mathrm{WeightedSum}$ in Eq. 6 is calculated as:

$$\begin{aligned} \mathrm{WeightedSum}(\boldsymbol{E}_l, \boldsymbol{x}) &= \sum_{l \in \mathrm{ActSet}} \boldsymbol{E}_l(\boldsymbol{x}) \cdot \texttt{group_score}[\mathrm{any}, \mathrm{any}, l], \\ \mathrm{ActSet} &= \texttt{group_act}[\mathrm{any}, \mathrm{any}, :], \end{aligned} \tag{7}$$

where any refers to any position of tokens and any group of experts. The other perspective corresponds to Line 12–18 computes the mask concerning the selected vectors in the sparse concatenation layer. In detail, each concatenation component is selected from R_O candidate vectors. The $\mathrm{Top1Select}$ in Eq. 3 chooses the candidate with the highest `cc_score`, which is the sum of scores of all experts in the corresponding group, and it is formulated as:

$$\mathrm{Top1Select}(\boldsymbol{O}_i^j) = \boldsymbol{O}_i^{\texttt{cc_act}[\mathrm{any},\texttt{i},\texttt{0}]}, \tag{8}$$

where any refers to any position of tokens. After that, the mask is then broadcast to all experts, by which the group of experts corresponding to the selected vector are not masked.

The `final_mask` is obtained by the element-wise AND operation between these two masks in Line 19–21. After applying the `final_mask` on the initial `score` and thus obtaining the `final_score`, $G_O T_I$ experts with higher score are activated for computation, as in Line 22–24.

Based on the router mechanism design, the computation process of a token sequence in the sparse experts is shown in Fig. 3 in Appendix A. After the router assigns each token to its activated experts, the tokens are permuted and dispatched to the corresponding experts for parallel forward. Upon completion, they are restored to the original order. Within each expert group, the outputs pertaining to the same token are aggregated into a dimension-reduced vector via weighted sum. Then, the sparsely selected vectors are directly concatenated to yield the final outputs of the sparse experts.

3.3 Upcycling for FineRMoE

Training MoEs from scratch is notoriously expensive. As an efficient paradigm of training MoEs, existing upcycling methods are tailored to single-layer, weighted-sum MoEs, rendering them inapplicable to the proposed FineRMoE architecture. To this end, we present an upcycling method for training the FineRMoE efficiently. Given a pre-trained dense LLM with the FFNs composed of the up projection weight $\boldsymbol{W}_1^p \in \mathbb{R}^{h \times H}$, the gate weight $\boldsymbol{W}_g^p \in \mathbb{R}^{h \times H}$, and the down projection weight $\boldsymbol{W}_2^p \in \mathbb{R}^{H \times h}$, the shared expert in FineRMoE is initialized by copying the pre-trained FFNs:

$$\boldsymbol{W}_1^s = \boldsymbol{W}_1^p, \quad \boldsymbol{W}_g^s = \boldsymbol{W}_g^p, \quad \boldsymbol{W}_2^s = \boldsymbol{W}_2^p. \tag{9}$$

As for the sparse finer-grained experts with index k ranging from 0 to $N - 1$, their weights $\boldsymbol{W}_{1k}^e \in \mathbb{R}^{h \times H_e}$ and $\boldsymbol{W}_{gk}^e \in \mathbb{R}^{h \times H_e}$ are constructed by splitting the $\boldsymbol{W}_1^p$ and $\boldsymbol{W}_g^p$ along the intermediate dimension, while the weight $\boldsymbol{W}_{2k}^e \in \mathbb{R}^{H_e \times h_e}$ is constructed by splitting the $\boldsymbol{W}_2^p$ along both the intermediate and output dimensions. The detailed expert construction is formulated as:

$$\begin{aligned} &i = (k \bmod (G_I R_I)) \bmod G_I, \quad j = \lfloor \frac{k}{R_O G_I R_I} \rfloor, \\ &\boldsymbol{W}_{1k}^e = \boldsymbol{W}_1^p[:, i \cdot \frac{H}{G_I} : (i+1) \cdot \frac{H}{G_I}], \boldsymbol{W}_{gk}^e = \boldsymbol{W}_g^p[:, i \cdot \frac{H}{G_I} : (i+1) \cdot \frac{H}{G_I}], \\ &\boldsymbol{W}_{2k}^e = \boldsymbol{W}_2^p[i \cdot \frac{H}{G_I} : (i+1) \cdot \frac{H}{G_I}, j \cdot \frac{h}{G_O} : (j+1) \cdot \frac{h}{G_O}], \end{aligned} \tag{10}$$

Table 1: The performance comparison of the proposed FineRMoE against baselines. #P/B: Total parameter size. #AP/B: Activated parameter size. Hell.: HellaSwag. Wino.: WinoGrande. ARCC.: ARC-Challenge. ARCE.: ARC-Easy. AGIE.: AGIEval.

	#P/B	#AP/B	MMLU	BBH	Hell.	Wino.	ARCC.	ARCE.	AGIE.	MBPP	GSM8K	GPQA	AVG
Base Model: Qwen2.5-0.5B													
PT	0.49	0.49	47.50	23.18	51.84	56.67	34.81	68.90	28.07	29.40	41.70	29.80	41.19
CT	0.49	0.49	46.18	32.04	52.71	57.46	35.58	68.06	27.31	28.00	38.51	27.23	41.31
C32A2	10.36	0.95	44.52	31.12	51.82	55.56	34.84	67.42	28.31	33.80	37.32	28.12	41.28
S16A4	0.63	0.40	26.04	12.95	30.45	51.70	22.95	46.46	25.47	0.00	1.90	23.88	24.18
DU	2.83	0.94	24.36	6.93	28.51	50.67	21.93	40.70	25.86	0.00	2.27	25.00	22.62
NVShard	2.83	0.63	39.49	27.98	49.44	55.56	34.56	65.40	26.79	16.80	26.84	28.57	37.14
FineRMoE	1.68	0.65	45.43	30.81	51.74	55.88	34.90	67.55	28.19	33.40	37.60	28.35	**41.39**
Base Model: Qwen2.5-1.5B													
PT	1.54	1.54	60.87	43.33	67.84	64.88	54.86	81.02	39.83	43.40	65.73	32.14	55.39
CT	1.54	1.54	60.70	45.15	68.68	65.43	53.41	80.72	38.91	41.20	66.11	31.47	55.18
C32A2	37.62	2.94	59.35	46.78	68.68	65.30	52.90	80.01	43.26	50.60	65.73	32.37	56.50
S16A4	1.78	0.91	25.06	23.41	35.06	51.70	25.34	50.38	25.91	1.00	3.03	22.32	26.32
DU	9.87	2.93	25.93	10.17	30.07	52.41	20.99	43.27	25.52	0.00	1.90	24.78	23.50
NVShard	9.88	1.78	56.96	41.13	66.72	63.30	49.49	77.82	36.78	33.80	64.67	31.03	52.14
FineRMoE	5.64	1.85	59.64	47.09	68.17	65.43	53.33	80.81	43.14	50.40	65.81	32.37	**56.62**
Base Model: Qwen2.5-7B													
PT	7.62	7.62	74.28	68.30	80.26	76.32	63.65	87.12	56.20	63.60	84.31	35.04	68.91
CT	7.62	7.62	72.97	69.59	80.47	75.53	63.65	87.33	52.67	64.00	85.37	34.60	68.62
C32A2	184.42	13.33	74.60	70.16	80.38	77.03	63.31	86.32	55.91	71.60	83.62	36.28	69.92
S16A4	7.62	3.34	35.79	31.76	47.04	56.59	34.64	60.98	28.94	17.20	21.30	23.44	35.77
DU	47.54	13.32	25.71	17.98	34.07	51.85	25.51	51.30	25.65	0.00	2.81	23.66	25.85
NVShard	47.55	7.63	69.81	66.67	79.05	74.27	59.81	84.97	49.36	53.20	83.62	31.03	65.18
FineRMoE	26.65	7.94	73.08	71.22	79.51	75.69	63.40	86.95	56.70	69.40	85.37	39.06	**70.04**

where mod is the modulo operation, and $\lfloor \cdot \rfloor$ calculates the integer part of the division. Therefore, by configuring the 4 hyper-parameters G_I, R_I, G_O, R_O, our upcycling method can be applicable to existing MoEs beyond to the proposed FineRMoE architecture. Moreover, **the above expert construction process could ensure that the output of each MoE layer is as consistent as possible with the output of the original dense model's FFN layer during the initial training phase.**

By setting $G_O, R_O, G_I = 1$ and R_I as the duplication times, upcycling that build MoEs by replicating FFNs [17, 36] can be fulfilled. By setting $G_O, R_O, R_I = 1$ and G_I as the split times, one can achieve upcycling via partitioning the FFNs along the intermediate dimension [47, 11].

4 Experiments

Our work primarily focuses on the upcycling approach, for a fair comparison, we ensure that all comparisons are based on the same dense pre-trained base models with the same training data recipe. We provide the experimental setup including training and evaluation details in Appendix B, with supplemental analysis and ablation studies in Appendix C–F. Further validation on the generality of upcycling different model families into FineRMoE is presented in Appendix G.

4.1 Baseline Comparison

Based on Qwen2.5 [41] with sizes of 0.5B, 1.5B and 7B, the baselines are as below:

PT: The official pre-trained models. **CT**: Continued training the dense models directly. **C32A2** [17]: Copying the pre-trained FFN for 32 times as experts and 2 of them are activated. We implement C32A2 by setting $G_I = 1, R_I = 32, G_O = 1, R_O = 1$. **S16A4** [47]: Splitting the pre-trained FFN for 16 times as experts and 4 of them are activated. We implement S16A4 by setting $G_I = 16, R_I = 1, G_O = 1, R_O = 1$. **DU** [25]: Replicating pre-trained FFN for 8 times first, then re-initializing 50% of the parameters in each weight matrix in each expert, and 2 experts are activated. **NVShard** [11]: Splitting the pre-trained FFN for 8 times and replicating all split parts for 8 times, resulting in 64 experts in total and 8 of them are activated.

According to Sec. 4.5 analyzed later, the 4 hyper-parameters of FineRMoE are configured as: $G_I = 32, R_I = 1, G_O = 2, R_O = 2$, leading to 128 total experts, and $T_I = 1$, leading to $G_O T_I = 2$ activated experts. Experiments for baseline comparison are performed by training on 50B tokens.

Table 2: The effectiveness validation of the finer-grained design in the proposed FineRMoE based on Qwen2.5-1.5B by training on 10B tokens. FG: Fine-Grained. Dim: Dimension.

Settings	Pre-Trained	No FG	Inter FG	Out FG	Inter&Out FG
Intermediate FG	-	×	✓	×	✓
Output FG	-	×	×	✓	✓
G_I,R_I,G_O,R_O	-	1,1,1,2	32,1,1,2	1,1,2,2	32,1,2,2
Intermediate Dim	8960	8960	280	8960	280
Output Dim	1536	1536	1536	768	768
#A-Experts / #Experts	- / -	1 / 2	1 / 64	2 / 4	2 / 128
#A-Param / #Param (B)	1.54 / 1.54	2.93 / 4.09	1.82 / 4.09	3.70 / 5.63	1.85 / 5.64
MMLU	60.87	51.13	56.25	59.52	59.30
BBH	43.33	34.37	41.18	45.49	45.97
HellaSwag	67.84	66.59	66.26	67.52	67.51
WinoGrande	64.88	59.91	63.54	65.90	65.27
ARC-C	54.86	49.83	51.02	52.39	53.41
ARC-E	81.02	78.28	78.96	80.26	80.35
AGIEval	39.83	34.28	39.99	42.24	41.81
MBPP	43.40	32.60	41.00	49.60	50.00
GSM8K	65.73	41.39	62.24	67.25	66.34
GPQA	32.14	29.69	30.13	31.03	32.14
Average	55.39	47.81	53.06	56.12	**56.21**

As shown from the results in Table 1, **FineRMoE outperforms PT and CT over 1 point on sizes of 1.5B and 7B. Moreover, FineRMoE attains an average performance improvement of at least 70% over S16A4 and DU across all three model sizes, and more than 4 points advantages compared with NVShard.** Notably, while continued training on the same data slightly degrades the performance of the dense models compared to the pre-trained version, FineRMoE produces substantial gains. For dense models, all parameters are activated during both training and inference, meaning that continual pre-training with new data affects the entire model. This often leads to catastrophic interference as new knowledge conflicts with previously acquired knowledge. While the sparse activation of MoE models enables the model to acquire new knowledge efficiently while preserving its pre-trained capabilities with fewer conflicts.

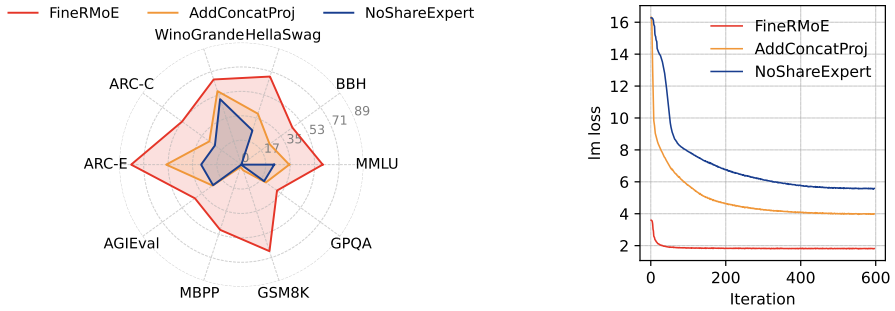
Although **C32A2 constructs MoE models with more than 6 times of parameters than ours**, FineRMoE still achieves better performance, indicating its high parameter efficiency and effective expert learning. In contrast, though S16A4 minimizes parameter overhead, its performance collapses, which may be caused by the lack of shared expert. Subsequent ablation study on the shared expert in the Sec. 4.4 reproduces analogous observations, evidencing that shared experts are essential for sparse fine-grained experts. Besides, Drop-Upcycling achieves the performance far inferior to that of FineRMoE. We infer the reason as the existence of part of re-initialized parameters. In the paper of Drop-Upcycling [25], experiments are performed by training for 500B tokens. For a fair comparison with other methods, we perform training for 50B tokens. Consequently, Drop-Upcycling begins with a higher training loss and converges more slowly. This demonstrates that FineRMoE also exhibits data-efficiency compared with Drop-Upcycling in building MoE models from dense models. In addition, FineRMoE achieves an average performance advantage of around 4 points than NVShard across three sizes, validating the effectiveness of our method.

Except for the benchmark performance, we also compare the inference efficiency of the resulted models. Appendix C demonstrates that our FineRMoE also achieves the optimal inference efficiency with the highest throughput. Notably, **compared with the strongest upcycling baseline method, FineRMoE's TTFT (*178.3 ms*) is 281 times faster than that of C32A2 (*50245.9 ms*), and its throughput (*27.3 tokens/s*) is 136 times higher than that of C32A2 (*0.2 tokens/s*)**.

Our experiments confirm the efficacy of the proposed upcycling method. Moreover, they demonstrate that the FineRMoE architecture achieves a superior trade-off between parameter efficiency and performance, along with optimal inference efficiency, outperforming all baseline methods.

4.2 Effectiveness Validation of Finer-Grained Design

To evaluate the effectiveness of the finer-grained design in the FineRMoE architecture, we experiment based on the Qwen2.5-1.5B by training on 10B tokens for efficiency. Four settings are included: (1) no fine-grained design (**No FG**); (2) fine-grained design only on the intermediate dimension (**Inter**



(a) Performance comparison.

(b) Language model loss.

Figure 2: The ablation study on the architecture of FineRMoE based on Qwen2.5-1.5B.

Table 3: The ablation study on router design based on Qwen2.5-1.5B.

Settings	MMLU	BBH	Hell.	Wino.	ARCC.	ARCE.	AGIE.	MBPP	GSM8K	GPQA	AVG
Separate Router	59.31	43.62	67.73	64.88	52.05	79.80	38.62	45.00	66.19	32.37	54.96
Single Router	59.30	45.97	67.51	65.27	53.41	80.35	41.81	50.00	66.34	32.14	**56.21**

FG); (3) fine-grained design only on the output dimension (**Out FG**); and (4) fine-grained design on both intermediate and output dimensions (**Inter&Out FG**).

According to Table 2, the poorest performance arises when fine-grained design is removed from both intermediate and output dimensions. Introducing fine-grained design only on the intermediate dimension brings noticeable improvement, but the performance still remains below that of the pre-trained model. In comparison, applying fine-grained design only to the output dimension produces a substantial gain by an average of 0.73 compared with the pre-trained model. This output-only fine-grained design outperforms its intermediate-only counterpart by 3 points, underscoring the critical role of specialization at the output level. Remarkably, with fine-grained design applied across both two dimensions, FineRMoE achieves the best average performance. It also exhibits superior parameter efficiency, attaining better performance with fewer activated parameters. Furthermore, as detailed in the analysis on Fig. 4 in Appendix D, the lowest average similarity among experts in each layer is achieved by FineRMoE, demonstrating the high specialization of its experts. These experiments systematically validate the effectiveness of the finer-grained design in our FineRMoE architecture, contributing to enhanced expert specialization and improved performance beyond intermediate fine-grained design only.

4.3 Effectiveness Validation of Router Design

To validate the effectiveness of the proposed router mechanism, which employs a single router to control the activation in the two sparse layers, we implement a version with two separate routers. For a quick validation, results achieved by training with 10B tokens in Table 3 demonstrates that the single-router design achieves better performance.

We analyze the reason as that the single-router mechanism ensures that the vectors selected in the sparse concatenation layer are composed of outputs from experts with relatively higher routing scores. In contrast, using two separate routers could lead to a discrepancy: an expert highly scored by the router of sparse sum layer might not be selected by the router of the sparse concatenation layer. Consequently, the MoE layer's output might be dominated by experts with lower scores from the sparse sum layer, leading to performance degradation. This comprehensively validates the effectiveness of the proposed unified router mechanism.

4.4 Ablation Study on FineRMoE Architecture

To further explore the design criteria of the FineRMoE architecture, we compare two variants:

AddConcatProj: In the multi-head attention [35], a linear layer is adopted for projection after concatenating the outputs from each head. To explore its impact, we implement the variant by adding a linear projection layer after the concatenation operation in the FineRMoE.

Table 4: The ablation study on fine-grained configurations based on Qwen2.5-1.5B.

G_I	G_O	#Experts	#A-Experts	#Param/B	#A-Param/B	Inter-Dim	Out-Dim	Performance
Pre-trained		-	-	1.54	1.54	8960	1536	55.39
2	2	8	2	5.63	2.74	4480	768	53.71
4	2	16	2	5.63	2.26	2240	768	54.13
4	4	32	4	8.72	2.65	2240	384	55.33
8	2	32	2	5.63	2.02	1120	768	55.00
8	4	64	4	8.72	2.22	1120	384	56.09
16	2	64	2	5.64	1.90	560	768	55.45
16	4	128	4	8.72	2.00	560	384	55.98
16	8	256	8	14.90	2.21	560	192	56.08
32	**2**	**128**	**2**	**5.64**	**1.85**	**280**	**768**	**56.21**
32	4	256	4	8.74	1.91	280	384	56.25
32	8	512	8	14.92	2.03	280	192	56.01
64	2	256	2	5.65	1.83	140	768	56.14
64	4	512	4	8.76	1.88	140	384	56.26
64	8	1024	8	14.97	1.97	140	192	56.34

NoShareExpert: To explore the necessity of the shared expert, which has been adopted in main experiments, we implement the variant by removing the shared expert.

For an efficient study, the experiments of the variants are conducted based on Qwen2.5-1.5B by training on 10B tokens. From the performance comparison in Fig. 2 (a), FineRMoE consistently surpasses its two architectural variants by large margins on all benchmarks. Besides, as shown in Fig. 2 (b), at the beginning of training, FineRMoE exhibits a much lower initial language model (lm) loss compared to its two variants. Throughout the training process, FineRMoE demonstrates the quickest convergence speed, ultimately achieving the lowest lm loss value at the end of training.

The ablation results clearly demonstrate the effectiveness of our architectural design. First, adding a projection layer after concatenation harms performance, which may be due to the lack of effective initialization, thus leading to inadequate training of the linear layer. Second, the shared expert is essential, removing it causes poor convergence during training when using fine-grained sparse experts, which is consistent with the results achieved by S16A4 in Table 1.

4.5 Ablation Study on Fine-Grained Configurations

With fine-grained design applied across both intermediate and output dimensions, we delve into the performance impact of different fine-grained configurations. With $R_I = 1, R_O = 2$, experiments are conducted by setting $G_I \in [2, 4, 8, 16, 32, 64]$ and $G_O \in [2, 4, 8]$ based on Qwen2.5-1.5B by training on 10B tokens for a quick study. The statistics of number of experts, the amount of parameters, and the performance are presented in Table 4. Details are given in Appendix E.

Regarding G_I, the results show that the finer-grained MoEs with the intermediate granularity no fewer than 8 can exceed the performance of the pre-trained model through upcycling training with suitable output granularity. This underscores the importance of fine-grained design along the intermediate dimension in reducing expert redundancy and enhancing model performance. More notably, for each G_I configuration, increasing G_O from 2 to 8 generally contributes to a stable and consistent performance gain for the built MoE models. This further highlights that **building on intermediate-dimension fine-grained design, higher output-dimension granularity can further boost expert specialization and thereby realize additional gains in overall model performance**. Besides, ablation studies on the number of activated experts T_I, intermediate expansion rate R_I, and output expansion rate R_O are analyzed in Appendix F. Considering the trade-off between parameter efficiency and performance, we ultimately set $G_I = 32$, $R_I = 1$, $G_O = 2$, $R_O = 2$ in the main experiments above.

5 Conclusions

Beyond the intermediate dimension, we propose FineRMoE expanding the fine-grained expert design in MoE models to the output dimension. We devise a bi-level sparse forward computation paradigm and a specially designed router mechanism. For efficient MoE model training, we propose a generalized upcycling method not only applicable to the FineRMoE architecture, but also compatible with existing MoEs. Based on Qwen2.5 with sizes of 0.5B, 1.5B and 7B, we build FineRMoE with 128 total experts and 2 activated via our upcycling method. Experiments show it achieves the best performance, as well as significant efficiency on both parameters and inference speed.

Limitations. Due to constraints in compute and data resources, the potential of FineRMoE in pre-training from scratch remains unexplored. Ongoing research efforts will address this gap to further corroborate the advantages of the FineRMoE architecture.

6 Acknowledgements

This work was partly supported by the Scientific Research Innovation Capability Support Project for Young Faculty (U40) of the Ministry of Education of China, SRICSPYF-ZY2025019, and Shanghai Municipal Universities Industry-Academia-Research Joint Innovation Program.

References

[1] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[2] Enric Boix-Adsera and Philippe Rigollet. The power of fine-grained experts: Granularity boosts expressivity in mixture of experts. *arXiv preprint arXiv:2505.06839*, 2025.

[3] Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025.

[4] 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.

[5] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[7] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[8] 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.

[9] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International conference on machine learning*, pp. 5547–5569. PMLR, 2022.

[10] 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, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL `https://zenodo.org/records/12608602`.

[11] Ethan He, Abhinav Khattar, Ryan Prenger, Vijay Korthikanti, Zijie Yan, Tong Liu, Shiqing Fan, Ashwath Aithal, Mohammad Shoeybi, and Bryan Catanzaro. Upcycling large language models into mixture of experts. *arXiv preprint arXiv:2410.07524*, 2024.

[12] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

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

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

[15] Wangyi Jiang, Yaojie Lu, Hongyu Lin, Xianpei Han, and Le Sun. Improved sparse upcycling for instruction tuning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 9485–9498, 2025.

[16] Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

[17] Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. Sparse upcycling: Training mixture-of-experts from dense checkpoints. In *The Eleventh International Conference on Learning Representations*, 2023.

[18] Ning Liao, Xiaoxing Wang, Zehao Lin, Weiyang Guo, Feng Hong, Shixiang Song, Geng Yu, Zihua Zhao, Sitao Xie, Longxuan Wei, et al. Innovator: Scientific continued pretraining with fine-grained moe upcycling. *arXiv preprint arXiv:2507.18671*, 2025.

[19] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

[20] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

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

[22] Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2):275–293, 2014.

[23] Meta. Llama 4 maverick: Open multimodal MoE at 400b scale, April 2025. URL `https://ai.meta.com/blog/llama-4-multimodal-intelligence/`. Meta AI Blog.

[24] Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. Olmoe: Open mixture-of-experts language models, 2024. URL `https://arxiv.org/abs/2409.02060`.

[25] Taishi Nakamura, Takuya Akiba, Kazuki Fujii, Yusuke Oda, Rio Yokota, and Jun Suzuki. Drop-upcycling: Training sparse mixture of experts with partial re-initialization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=gx1wHnf5Vp`.

[26] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

[27] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[28] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[29] Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Rozière, Jacob Kahn, Daniel Li, Wen-tau Yih, Jason Weston, et al. Branch-train-mix: Mixing expert llms into a mixture-of-experts llm. *arXiv preprint arXiv:2403.07816*, 2024.

[30] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

[31] Yehui Tang, Xiaosong Li, Fangcheng Liu, Wei Guo, Hang Zhou, Yaoyuan Wang, Kai Han, Xianzhi Yu, Jinpeng Li, Hui Zang, et al. Pangu pro moe: Mixture of grouped experts for efficient sparsity. *arXiv preprint arXiv:2505.21411*, 2025.

[32] Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

[33] Meituan LongCat Team, Bei Li, Bingye Lei, Bo Wang, Bolin Rong, Chao Wang, Chao Zhang, Chen Gao, Chen Zhang, Cheng Sun, et al. Longcat-flash technical report. *arXiv preprint arXiv:2509.01322*, 2025.

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

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[36] Aditya Vavre, Ethan He, Dennis Liu, Zijie Yan, June Yang, Nima Tajbakhsh, and Ashwath Aithal. Llama 3 meets moe: Efficient upcycling. *arXiv preprint arXiv:2412.09952*, 2024.

[37] Bin Wang, Bojun Wang, Changyi Wan, Guanzhe Huang, Hanpeng Hu, Haonan Jia, Hao Nie, Mingliang Li, Nuo Chen, Siyu Chen, et al. Step-3 is large yet affordable: Model-system co-design for cost-effective decoding. *arXiv preprint arXiv:2507.19427*, 2025.

[38] Tianwen Wei, Bo Zhu, Liang Zhao, Cheng Cheng, Biye Li, Weiwei Lü, Peng Cheng, Jianhao Zhang, Xiaoyu Zhang, Liang Zeng, et al. Skywork-moe: A deep dive into training techniques for mixture-of-experts language models. *arXiv preprint arXiv:2406.06563*, 2024.

[39] Xun Wu, Shaohan Huang, Wenhui Wang, Shuming Ma, Li Dong, and Furu Wei. Multi-head mixture-of-experts. *Advances in Neural Information Processing Systems*, 37:94073–94096, 2024.

[40] Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You. Openmoe: an early effort on open mixture-of-experts language models. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 55625–55655, 2024.

[41] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, December 2024. URL `https://arxiv.org/abs/2412.15115`. Last revised 2025-01-03.

[42] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[43] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[44] Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, et al. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*, 2025.

[45] Qizhen Zhang, Nikolas Gritsch, Dwaraknath Gnaneshwar, Simon Guo, David Cairuz, Bharat Venkitesh, Jakob Foerster, Phil Blunsom, Sebastian Ruder, Ahmet Üstün, et al. Bam! just like that: Simple and efficient parameter upcycling for mixture of experts. *Advances in Neural Information Processing Systems*, 37:56304–56321, 2024.

[46] Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*, 2023.

[47] Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. Llama-moe: Building mixture-of-experts from llama with continual pre-training. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15913–15923, 2024.

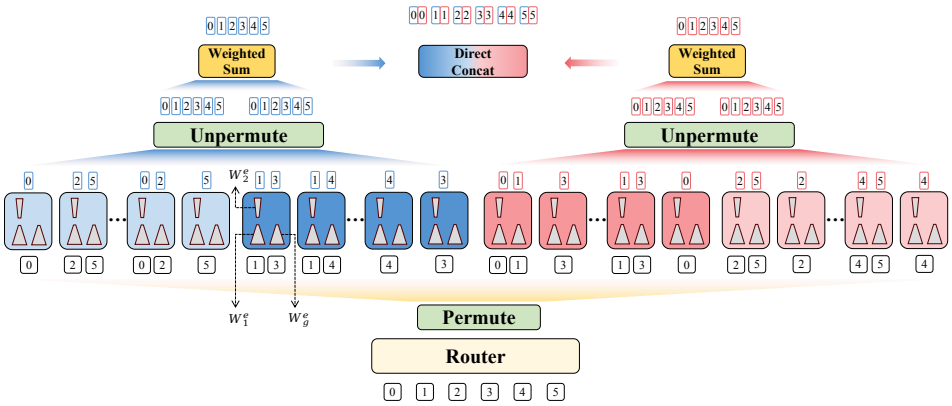


Figure 3: The forward computation process of a sequence of tokens in the sparse experts of FineRMoE. For a given input sequence, the router first calculates the set of activated experts for each token. The tokens are then permuted to allow for parallel expert computation. After processing by the experts, the outputs are unpermuted to their original token order. For each token, the outputs from its activated experts are combined via a weighted sum to form dimension-reduced components. Finally, these components are concatenated to produce the final, dimension-restored output.

A Forward Computation Process

Based on the router mechanism presented in Sec. 3.2, the computation process of a sequence of tokens in the sparse experts of FineRMoE is demonstrated in Fig. 3.

B Experimental Setup

B.1 Load Balancing Loss

During the training phase, the FineRMoE is optimized with the sum of language modeling loss and the weighted load balancing loss. Specifically, we follow DeepSeek-V2 [19] for the load balancing loss design. Given the number of experts as N, the number of activated experts per token as $G_O T_I$, the number of tokens in a sequence as L, the score of assigning the t-th token to the i-th expert as $s_{i,t}$, the load balancing loss L_{lbl} is calculated as:

$$\begin{aligned} L_{lbl} &= \alpha \sum_{i=1}^{N} f_i P_i, \\ P_i &= \frac{1}{L} \sum_{t=1}^{L} s_{i,t}, \\ f_i &= \frac{N}{G_O T_I L} \sum_{t=1}^{L} \mathbf{I}_{i,t}, \end{aligned} \tag{11}$$

where $\mathbf{I}_{i,t}$ is an indicator function, which equals to 1 when the t-th token selects the i-th expert, otherwise 0. Throughout all experiments, the weight α of the load balancing loss is set as 0.001.

B.2 Implementation Details

We implement and train FineRMoE using the Megatron-LM [1] [28] framework for its parallelization flexibility, based on the Qwen2.5 [41]. The training data is prepared by mixing and refining publicly available pre-training corpora, including English webpage data, Chinese webpage data, English knowledge data, Chinese knowledge data, code data, math data, the pile, wiki, and book data. Aside

[1] https://github.com/NVIDIA/Megatron-LM/tree/6ba97dd37150a6bfba03d31808674211cf2a4d0d

Table 5: The training hyper-parameters in experiments.

Hyper-parameter	Setting
Optimizer	Adam
Adam_beta1	0.9
Adam_beta2	0.95
Clip Grad	1.0
Learning Rate	1e-5
Minimum Learning Rate	1e-7
LR Decay Style	Cosine
Precision	BF16
Micro Batch Size	1
Global Batch Size	2048
Sequence Length	8192
Max Padding Length	8192

from the different fine-grained and parallelization configurations, the data and detailed training settings are kept consistent across all experiments. For each training experiment, the number of warmup steps is 1% of the total training steps. Training hyper-parameters are summarized in Table 5.

We evaluate the models covering areas including knowledge, reasoning, code, and math using the widely-recognized Language Model Evaluation Harness [10] framework. The evaluation benchmarks include: MMLU [12], BBH [30], HellaSwag [43], WinoGrande [27], ARC-Challenge (ARC-C) [5], ARC-Easy (ARC-E) [5], AGIEval [46], MBPP [1], GSM8K [6], GPQA [26].

C Inference Efficiency Analysis

We complement the primary performance evaluation with an analysis of inference efficiency. Consistent with settings in Sec. 4.1, we select the C32A2, S16A4, Drop-Upcycling, NVShard and our FineRMoE, which are realized by continued training on 50B tokens based on Qwen2.5-7B, for comparison. Typically, model inference is divided into the prefill and decoding stages. For the prefill stage, the efficiency is evaluated by latency, specifically the Time to First Token (TTFT), which measures the duration from receiving the input to generating the first output token. A lower TTFT indicates higher prefill efficiency. For the decoding stage, the efficiency is evaluated by throughput, measured in tokens generated per second (tokens/s). A higher throughput indicates greater efficiency in the decoding stage. Given the input "*Give me a short introduction to large language model.*", the comparisons of inference efficiency and generated texts are presented in Table 6.

FineRMoE achieves the highest throughput. Owing to a parameter size far exceeding those of other methods, C32A2 demonstrates considerably inferior efficiency during both prefill and decoding stages, even though it achieves competitive performance to FineRMoE in Table 1. It is particularly noteworthy that FineRMoE outperforms C32A2 in benchmark performance while also delivering dramatically superior inference efficiency: a TTFT of 178.3 ms, which is **281 times faster** than C32A2's 50245.9 ms, and a throughput of 27.3 tokens/s, which is **136 times higher** than C32A2's 0.2 tokens/s. Although the smaller activation model size of S16A4 and NVShard affords them greater prefill efficiency, they exhibit lower efficiency during the decoding stage, and markedly poorer performance in Table 1. Furthermore, due to the partial re-initialization of experts, the model built by Drop-Upcycling suffers from severe performance degradation. This results in highly repetitive output that continues until it reaches the preset maximum length of 32000 tokens.

In addition, we compare the inference efficiency under varying input lengths for a comprehensive validation, results are presented in Table 7. Consistent with the findings in Table 6, FineRMoE achieves the highest inference throughput across all tested input-length configurations, which fully demonstrates its high efficiency during the decoding phase.

As comprehensively evidenced by the detailed comparisons in Table 1, Table 6 and Table 7, the proposed FineRMoE achieves optimal performance across ten benchmarks, as well as maintaining remarkable efficiency in both parameters and inference. This comprehensively demonstrates the superiority of the FineRMoE architecture compared to existing ones.

Table 6: The comparisons of the inference efficiency between C32A2, S16A4, DU, NVShard and our FineRMoE. The models are constructed by continued training on 50B tokens on Qwen2.5-7B.

Models	C32A2	S16A4	DU	NVShard	FineRMoE
#Param (B)	184.42	7.62	47.54	47.55	26.65
#A-Param (B)	13.33	3.34	13.32	7.63	7.94
#Generated Tokens	107	47	32000	332	110
TTFT (ms) ↓	50245.9	**78.5**	84.8	137.8	178.3
Throughput (tokens/s) ↑	0.2	24.0	25.6	18.9	**27.3**
Generated Text					
C32A2	*Large language models (LLMs) are artificial intelligence (AI) models that are trained on vast amounts of text data to understand and generate human language. They are designed to recognize patterns in language, understand context, and generate responses that are coherent and contextually appropriate. LLMs are used in a wide range of applications, including natural language processing (NLP), machine translation, and text summarization. They are also used in conversational agents, such as chatbots and virtual assistants, to provide human-like interactions with users.*				
S16A4	*Large language models are a type of artificial intelligence that can generate text based on a vast amount of training data. They are designed to understand and generate human-like text, often used in chatbots, customer service, and other applications.*				
DU	*Large language model (LLM) is a powerful tool for natural language processing, machine learning, and data science. It is a type of machine learning that uses a deep neural network to learn from data and make predictions. The goal of LLM is to understand and interpret human language, and to extract insights from it. Here is a short introduction to LLM:Introduction:LLM is a type of machine learning that uses a deep neural network to learn from data and make predictions. It is a popular tool for natural language processing, machine learning, and data science. LLM is a type of machine learning that uses a deep neural network to learn from data and make predictions. The goal of LLM is to understand and interpret human language, and to extract insights from it.* ***[Repeat...]***				
NVShard	*A large language model (LLM) is a type of artificial intelligence (AI) model that is designed to understand and generate human-like text. These models are typically based on deep learning techniques, particularly transformer architectures, which are capable of processing and understanding the context of large amounts of text data. LLMs are trained on vast datasets that include books, articles, websites, and other forms of text. During training, the model learns to predict the next word in a sequence based on the preceding words, which allows it to generate coherent and contextually relevant text. The size of the model, often measured in terms of the number of parameters it contains, is a key factor in its performance. Larger models tend to have more parameters and can capture more complex patterns in the data, leading to better performance on various natural language processing tasks.* *Some of the most well-known large language models include GPT-3 (Generative Pre-trained Transformer 3) by OpenAI, BERT (Bidirectional Encoder Representations from Transformers) by Google, and T5 (Text-to-Text Transfer Transformer) by Google. These models have been used for a wide range of applications, including language translation, question-answering, text summarization, and even creative writing.* *The development of large language models has been a significant advancement in the field of AI, as it has enabled machines to understand and generate human language in a way that was previously thought to be beyond the capabilities of computers. However, these models also raise important ethical and privacy concerns, as they can be used to generate misleading or harmful content and may inadvertently perpetuate biases present in the training data.*				
FineRMoE	*Large language models (LLMs) are artificial intelligence (AI) systems that are trained on vast amounts of text data to understand and generate human-like language. They are designed to comprehend context, answer questions, and generate coherent and relevant text based on the input they receive. LLMs are trained using deep learning techniques, such as transformers, which allow them to process and generate text in a way that is similar to human language processing. They are used in a wide range of applications, including natural language processing, machine translation, and chatbots.*				

Table 7: The inference efficiency comparison with different input token lengths.

	C32A2	S16A4	DU	NVShard	FineRMoE
Input Token Length=418					
TTFT (ms)	69581.5	86.2	**71.2**	252.0	316.2
Throughput (tokens/s)	0.2	24.5	25.6	18.0	**29.9**
Input Token Length=1509					
TTFT (ms)	75480.3	**115.2**	129.4	294.1	402.0
Throughput (tokens/s)	0.3	24.8	27.4	18.1	**30.1**
Input Token Length=3391					
TTFT (ms)	80488.7	**179.0**	248.8	380.6	484.3
Throughput (tokens/s)	0.3	24.5	26.3	18.0	**30.3**

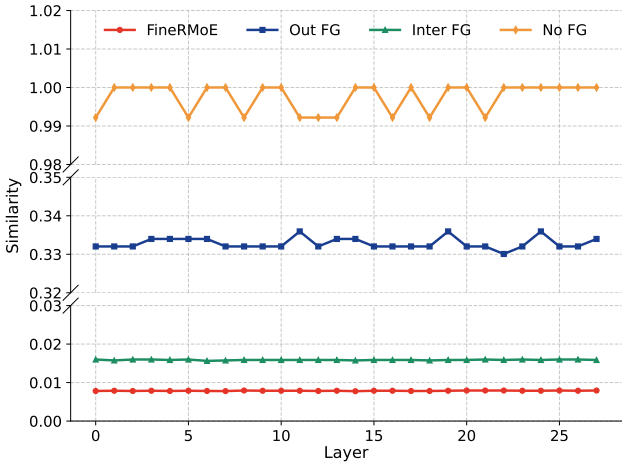


Figure 4: The average similarity among the sparse experts across all layers in the effectiveness validation of finer-grained design. The corresponding models are trained based on Qwen2.5-1.5B on 10B tokens for efficiency.

D Expert Similarity Analysis

To provide an intuitive demonstration of the benefits conferred by the finer-grained design, we additionally evaluate the average similarity among the sparse experts in each layer. The analyzed MoE models are trained via the four distinct settings in Sec. 4.2, i.e., fine-grained design only on intermediate dimension, output dimension, both the two dimensions, and no fine-grained design. Specifically, we enumerate all pairwise combinations of experts in each layer, calculate their cosine similarities, and then average across all combinations to derive the average expert similarity for all layers under each configuration, as delivered in Fig 4.

Notably, with the lack of fine-grained design, the model obtained by simply replicating the pre-trained FFNs demonstrates the highest expert similarity, indicating severe redundancy and explaining its poorest performance in Table 2. When fine-grained design is applied to only one dimension, the expert similarity is lower. Given that the intermediate granularity $G_I = 32$ is greater than the output granularity $G_O = 2$, applying the fine-grained design only to the output dimension leads to a slightly higher degree of expert similarity than applying it to the intermediate dimension. It is noteworthy that our proposed FineRMoE architecture, which employs the fine-grained design on both the intermediate and output dimensions, exhibits the lowest level of expert similarity across all four settings. This signifies that the experts have attained alleviated redundancy and a high degree of specialization, thereby leading to the superior performance in Table 2.

E Detailed Evaluation on Fine-Grained Configurations

The detailed evaluation results across ten benchmarks in the ablation study of fine-grained configurations are delivered in Table 8.

F Ablation Study on T_I, R_I, R_O

In our main experiments, we aim to achieve superior performance with high parameter efficiency for the FineRMoE model. To this end, while maintaining sparsity in both the concatenation and sum layers, we adopt the following settings to minimize both the total and activated parameters: the number of activated experts per group in the sparse sum layer $T_I = 1$, the intermediate expansion rate $R_I = 1$, and the output expansion rate $R_O = 2$.

In this section, we further explore the impact of T_I, R_I, R_O to the performance compared with the base configuration of $G_I = 32$, $R_I = 1$, $G_O = 2$, $R_O = 2$. Specifically, by setting $T_I \in [2, 4]$,

Table 8: Evaluation details across 10 benchmarks in the ablation study on fine-grained configurations based on Qwen2.5-1.5B.

G_I	G_O	MMLU	BBH	HellaSwag	WinoGrande	ARC-C	ARC-E	AGIEval	MBPP	GSM8K	GPQA
2	2	58.19	42.45	66.56	64.09	51.19	78.28	40.41	46.00	63.84	26.12
4	2	58.15	43.46	66.44	63.69	51.88	80.13	41.80	44.40	63.46	27.90
4	4	58.97	44.68	67.14	65.11	51.62	80.30	41.15	47.20	66.11	31.03
8	2	58.16	45.48	66.79	64.48	50.77	79.12	39.85	46.60	65.50	33.26
8	4	59.17	47.00	67.46	64.88	53.24	80.39	42.16	50.00	66.03	30.58
16	2	59.06	45.51	67.24	64.01	52.82	79.88	41.55	48.00	66.11	30.36
16	4	59.56	46.14	67.72	64.01	54.01	80.47	42.37	47.00	66.41	32.14
16	8	59.53	45.72	67.52	64.64	53.24	80.51	42.14	50.40	66.03	31.03
32	2	59.30	45.97	67.51	65.27	53.41	80.35	41.81	50.00	66.34	32.14
32	4	59.58	46.29	67.56	65.67	53.58	80.30	42.48	50.20	67.63	29.24
32	8	59.05	46.24	67.67	64.64	52.90	80.35	41.56	48.80	66.72	32.14
64	2	59.31	45.57	67.81	64.80	53.33	80.43	42.51	48.80	67.55	31.25
64	4	59.26	45.65	67.75	64.40	52.73	80.43	41.64	50.20	67.02	33.48
64	8	59.75	46.43	67.81	65.43	52.56	80.18	41.41	50.80	66.03	33.04

Table 9: The ablation study on T_I, R_I, R_O based on Qwen2.5-1.5B by training on 10B tokens.

Settings	Pre-Trained	Base	Ablation on T_I		Ablation on R_I		Ablation on R_O	
G_I,R_I,G_O,R_O	-	32,1,2,2	32,1,2,2	32,1,2,2	32,2,2,2	32,4,2,2	32,1,2,4	32,1,2,8
T_I	-	1	2	4	1	1	1	1
#Experts	-	128	128	128	256	512	256	512
#A-Experts	-	2	4	8	2	2	2	2
#Param / B	1.54	5.64	5.64	5.64	9.51	17.24	9.51	17.24
#A-Param / B	1.54	1.85	1.91	2.03	1.86	1.88	1.86	1.88
MMLU	60.87	59.30	59.37	59.76	59.14	59.68	59.44	58.99
BBH	43.33	45.97	46.97	46.11	45.38	45.12	46.15	45.72
HellaSwag	67.84	67.51	67.80	67.81	67.42	67.42	67.35	67.26
WinoGrande	64.88	65.27	63.85	64.33	64.72	65.59	63.93	65.11
ARC-C	54.86	53.41	52.90	53.41	53.92	53.07	53.33	51.88
ARC-E	81.02	80.35	80.47	80.35	79.97	80.05	80.35	79.84
AGIEval	39.83	41.81	41.87	42.03	41.68	41.45	41.64	41.84
MBPP	43.40	50.00	49.20	49.40	51.40	48.40	49.20	46.60
GSM8K	65.73	66.34	65.73	66.34	67.17	66.94	67.10	66.94
GPQA	32.14	32.14	32.59	33.71	31.92	33.26	31.03	33.04
Average	55.39	56.21	56.08	56.33	56.27	56.10	55.95	55.72

$R_I \in [2, 4]$, $R_O \in [4, 8]$, the ablation experiments are performed based on Qwen2.5-1.5B with 10B tokens trained for a quick investigation. Results are summarized in Table 9.

Compared with the pre-trained model, performance achieved by all ablation settings consistently exhibits remarkable advantages. Experimental results reveal that under the base fine-grained configuration, performance improvements can be stably achieved despite adjustments to T_I, R_I, R_O, thereby further corroborates the effectiveness of the proposed FineRMoE architecture and the upcycling method for efficient training. Compared with the base configuration, altering T_I, R_I, R_O leads to minor performance fluctuations. Therefore, considering the trade-off between performance gains and parameter efficiency, employing the base configuration represents the optimal choice.

G Generality on Different Model Families

To validate the generality of upcycling into the FineRMoE based on different model families, we perform experiments for further comparison. Within the Megatron-LM framework, conducting experiments on models with different tokenizers would require rebuilding the mmap for the whole dataset using its tokenizer, posing challenges to our experimental time and storage. Since Qwen2.5 and Qwen2/Qwen3 both use the Qwen2Tokenizer, we instead select these variants to conduct experiments for a quick validation on the generality. Specifically, we select Qwen2-1.5B and Qwen3-0.6B for a quick validation by training for 10B tokens. The performance achieved by the built FineRMoE compared with that of the pre-trained base model is shown in Table 10. Results validate that building

Table 10: The performance of the FineRMoE built on Qwen2-1.5B and Qwen3-0.6B with 10B tokens trained. Hell.: HellaSwag. Wino.: WinoGrande. ARCC.: ARC-Challenge. ARCE.: ARC-Easy. AGIE.: AGIEval.

	MMLU	BBH	Hell.	Wino.	ARCC.	ARCE.	AGIE.	MBPP	GSM8K	GPQA	AVG
Base Model: Qwen2-1.5B											
PT	55.96	36.55	66.94	65.27	43.86	72.73	36.62	37.80	46.90	30.14	49.28
FineRMoE	54.38	38.34	66.70	65.90	42.75	71.97	39.34	39.40	55.65	30.58	**50.50**
Base Model: Qwen3-0.6B											
PT	47.19	41.41	46.62	55.96	40.19	71.38	34.19	22.60	51.40	29.02	44.00
FineRMoE	47.61	42.13	49.61	58.48	40.36	70.79	32.20	26.00	48.45	29.69	**44.53**

on different base models, FineRMoE could still achieve notable performance improvements, proving its generality to different model families.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the contributions, including the finer-grained expert design, the bi-level sparse forward computation, the router mechanism and the generalized upcycling method. All quantitative claims are supported by experimental results in Sec. 4.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We present the limitations of this work in the Sec. 5, that is the potential of FineRMoE in pre-training from scratch has not been explored yet.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: This work does not include theoretical results.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We deliver the detailed experimental setup in Appendix B, including the load balancing loss, the training hyperparameters, and the data. Besides, we present the implementation of the router mechanism in a Pytorch-like style in Algorithm 1, which helps reproducing our work.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Code will be released upon acceptance. All evaluation datasets are publicly available.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We present all training and test details in Appendix B.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Error bars are not reported due to the high computational cost in training the FineRMoE. Besides, related works in this field do not report error bars, either.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We report the inference cost, including the TTFT and Throughput, in Table 6.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work presents the FineRMoE architecture and an associated upcycling method for efficient training. As a fundamental study in model design, the research involves no human subjects or sensitive data and raises no direct deployment concerns. The authors declare no conflicts of interest.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work does not introduce any potential societal impacts.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work does not release any datasets or models at this time.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All base models (Qwen2.5 series), frameworks (Megatron-LM), evaluation benchmarks have been properly cited.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new assets are introduced in the paper at this time.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing or human subjects are involved in this work.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

Justification: No human subjects are involved in this research.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The writing of this paper is polished with the assistance of LLMs for grammar checking and text refinement. Except for that, all core intellectual contributions, including the ideas, code development, experimental design, and result analysis, are conducted exclusively by the authors.