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

Ning Liao, Xiaoxing Wang[†]*, Xiaohan Qin, Junchi Yan*[†]

liaoning@sjtu.edu.cn



Motivation

Granularity Bottleneck

Within a suitable parameter regime, MoE performance improves with finer intermediate-dimension experts, but degrades beyond an optimal granularity.

Inspired by the MHA mechanism.

FineRMoE Architecture: Extends fine-grained expert design to the output dimension, further reducing parameter redundancy.

Bi-level Sparse Computation: Resolves the dimensional mismatch between MoE and attention layers caused by reducing expert dimensionality.

Unified Routing Mechanism: Uses a single router to coordinate activation across the two sparse layers within the MoE forward pass, reducing redundant and conflicting computations.

Prohibitive Cost

Training MoE models from scratch is expensive.

Upcyling dense models into MoE

Unified Upcycling Method : Supports both FineRMoE and single-layer MoE architectures. For single-layer MoE, it accommodates either FFN replication or partitioning along the intermediate dimension, addressing the limited generality of existing expert upcycling methods.

Architecture

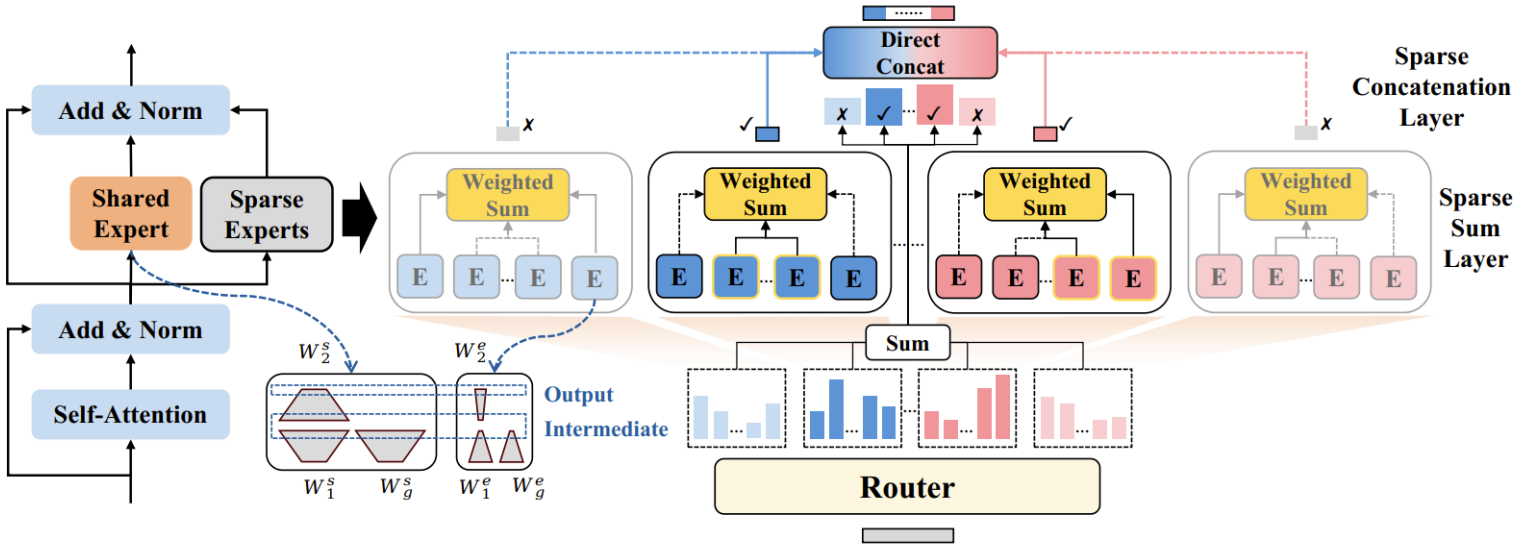


Figure 1: The architecture of FineRMoE. Unlike conventional MoEs that confine fine-grained design solely to the intermediate dimension, FineRMoE for the first time jointly expands it to the output dimension, enabling a bi-level sparse computation paradigm. It comprises a Sparse Sum Layer, which produces 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.

➢ Sparse Sum Layer

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

➢ Sparse Concatenation Layer

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

Router Mechanism

- **A single router is employed to control the activation of both sparse layers in FineRMoE, thereby reducing the redundant and conflicting computations that would otherwise arise from using two separate routers.**

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
```

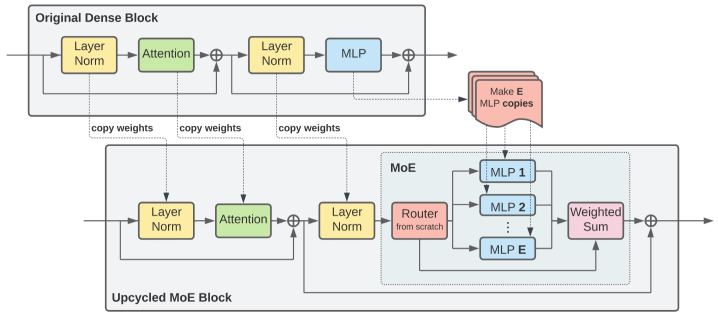
Unified Upcycling

$$i = (k \bmod (G_I R_I)) \bmod G_I, \quad j = \lfloor \frac{k}{R_O G_I R_I} \rfloor,$$
$$\boldsymbol{W}^e_{1k} = \boldsymbol{W}^p_1[:, i \cdot \frac{H}{G_I} : (i+1) \cdot \frac{H}{G_I}], \boldsymbol{W}^e_{gk} = \boldsymbol{W}^p_g[:, i \cdot \frac{H}{G_I} : (i+1) \cdot \frac{H}{G_I}],$$
$$\boldsymbol{W}^e_{2k} = \boldsymbol{W}^p_2[i \cdot \frac{H}{G_I} : (i+1) \cdot \frac{H}{G_I}, j \cdot \frac{h}{G_O} : (j+1) \cdot \frac{h}{G_O}],$$

➢ Replicate FFN (C32A2)

$G_O, R_O, G_I = 1$

R_I : Number of Experts

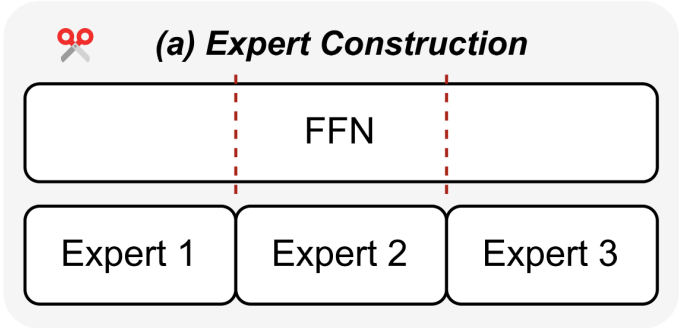


32 coarse-grained experts, activates 2.

➢ Split FFN along Inter-dim (S16A4)

$G_O, R_O, R_I = 1$

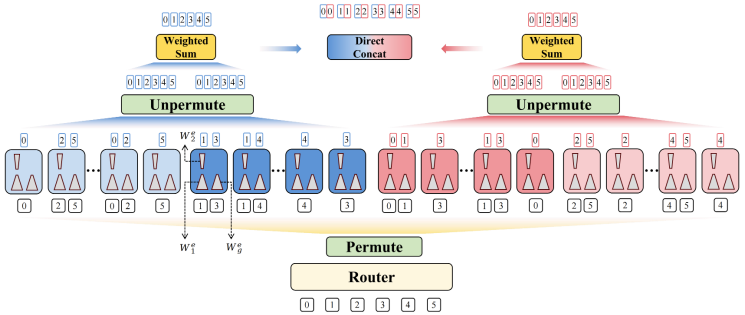
G_I : Number of Experts



16 fine-grained experts, activates 4.

➢ FineRMoE

$G_I = 32, R_I = 1,$
$G_O = 2, R_O = 2.$



128 finer-grained experts, activates 2.

Best Performance with High Inference Efficiency

Best Overall Performance: FineRMoE achieves the best aggregate performance across 10 benchmarks at three model scales.
Parameter Efficiency: FineRMoE uses less than **1/6** of the total parameters of C32A2, its closest competitor in performance.
Inference Efficiency: FineRMoE achieves the highest decoding efficiency, with TTFT and throughput **281×** and **136×** those of C32A2, respectively.

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**

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.*				

Finer Granularity, Better Performance

➢Finer-Grained Experts Deliver the Best Performance with the Lowest Expert Redundancy

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

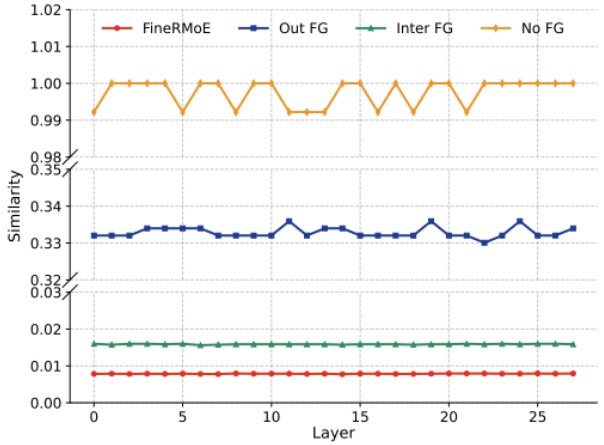


Figure 4: The average similarity among the sparse experts across all layers in the effectiveness validation of finer-grained design. The corresponding models are trained based on Qwen2.5-1.5B on 10B tokens for efficiency.

➢Finer Output-Layer Granularity Yields Better Overall Performance

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

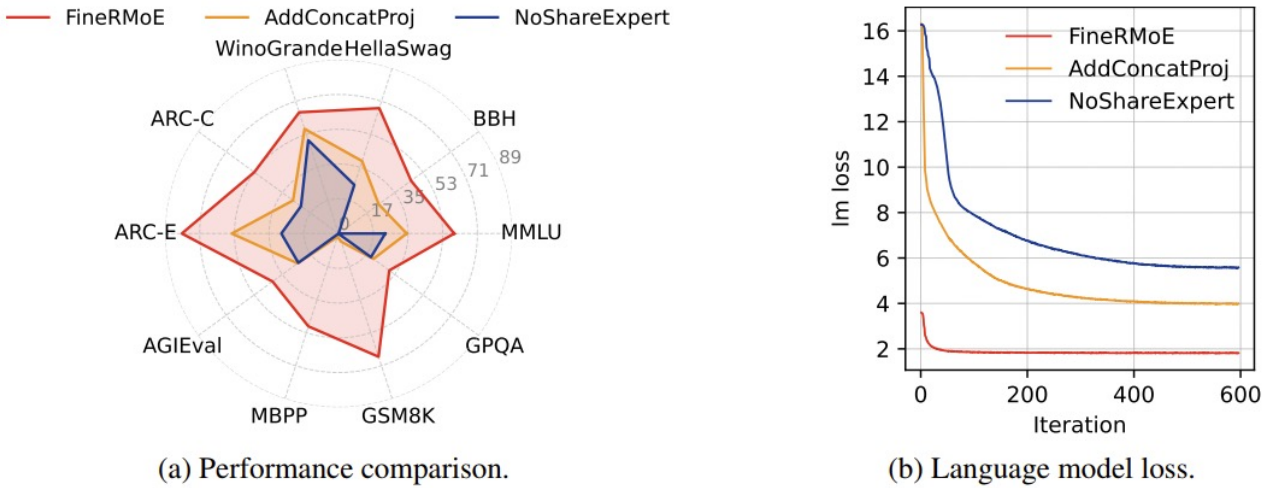
G_I	G_O	#Experts	#A-Experts	#Param/B	#A-Param/B	Inter-Dim	Out-Dim	Performance
Pre-trained		-	-	1.54	1.54	8960	1536	55.39
2	2	8	2	5.63	2.74	4480	768	53.71
4	2	16	2	5.63	2.26	2240	768	54.13
4	4	32	4	8.72	2.65	2240	384	55.33
8	2	32	2	5.63	2.02	1120	768	55.00
8	4	64	4	8.72	2.22	1120	384	56.09
16	2	64	2	5.64	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

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

Validating Architecture and Routing Effectiveness

- **Architecture Validation: Comparisons with a Post-Concatenation Projection Layer and a Variant without Shared Experts.**



(a) Performance comparison. (b) Language model loss.

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

- **Routing Validation: Comparison with Separate Routers Controlling Activation in Each Sparse Layer.**

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**


Thanks