Layerwise Recurrent Router for Mixture-of-Experts

[1]Zihan Qiu* [2]Zeyu Huang* [3]Shuang Cheng [4]Yizhi Zhou [5]Zili Wang
[2,6]Ivan Titov [7]Jie Fu†
[1]Alibaba Group, [2]University of Edinburgh, [3]ICT, Chinese Academy of Sciences,
[4]Nanjing University, [5]INF Technology [6]University of Amsterdam [7]Shanghai AI Lab
qzh11628@gmail.com, zeyu.huang@ed.ac.uk, fujie@pjlab.org.cn

Router is too weak to fully utilize expert parameters?

- Mixture of Experts (MoE) uses router to **conditionally** and **sparsely** assign each input token to its corresponding experts (FFN)
  - MoE can scale the model size and keep computational costs nearly unchanged
- With a cost of parameter inefficiency:
  - In DeepspeedMoE, a 52B MoE only performs comparably to a standard 6.7B model.
  - In DeepseekMoE, a fine-grained 16B MoE is comparable with a standard 7B model.
- The router might be the bottleneck for utilizing expert parameters:
  - Typically, the router is parameterized as one lightweight linear layers, which may limit its capacity to explore the optimal token-expert combination.
  - OpenMoE (Xue et al., 2024) finds the routing converges to the token-id-based routing very quickly, which means the token-expert combination is not explored.
  - Some works even show hash functions (Roller et al., 2021), stochastic routing (Zuo et al., 2021), and fixed-random router (Chen et al., 2023) also works well.
- The learnable router component in MoE needs further enhancement!

Capturing Cross-Layer Routing Dependencies with a Recurrent Router

- The routing in consecutive layers can be viewed as a sequence
  - The routing results of the i-th layer is conditioned on previous layers' decisions.
  - Existing attempts to improve router still operate independently in each layer.
- We introduce a lightweight Gated Recurrent Unit (GRU) to capture the cross-layer routing dependencies!

Capturing Cross-Layer Routing Dependencies with a Recurrent Router

- The routing in consecutive layers can be viewed as a sequence
- We introduce a lightweight Gated Recurrent Unit (GRU) to capture the cross-layer routing dependencies

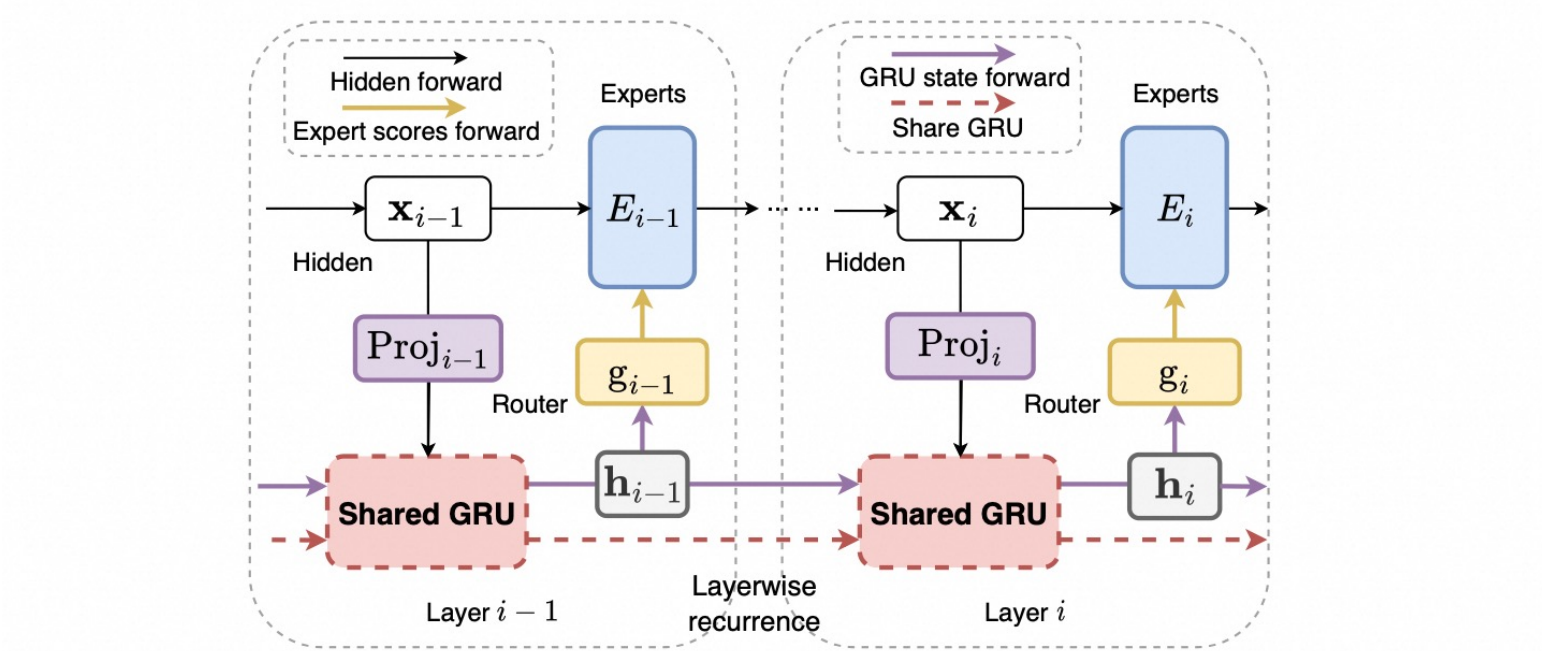

Figure 1: Recurrent router for Mixture-of-Experts. In the i-th layer, the hidden state $\mathbf{x}_i$ is **I.** projected to $\mathbf{x}'$ with alower hidden dimension (Eq. 4), **II.** combined with previous layer's GRU output $\mathbf{h}_{i-1}$, and processed through the cross-layer-shared GRU to produce the current layer's GRU output, $\mathbf{h}_i$ (Eq. 5). **III.** layer i's router uses this output to select experts and executes standard MoE computation (Eq. 6). Such operation doesn't introduce sequence-level recurrence and can be efficiently implemented, as shown in Tab. 1 and Tab. 3.

Main Results

Table 1: Performance of RMoE and baselines on two language modeling tasks, Enwiki8 and WikiText-103. **Params** means the non-embedding model parameters and (router parameters). Notice we don't separate unlearnable parameters in HyperMoE and RandomSMoE. **Mem** means the peak GPU memory usage with the same batch-size configurations. **Speed** is the average time for 1k training steps. Results demonstrate that the RMoE outperforms baseline models and achieves comparable memory usage and speed as the standard SMoE.

Algorithm	**Enwiki8 (BPC)↓**		**WikiText (PPL)↓**		**Params**	**Mem**	**Speed**
	val	test	val	test	(M)	(GB)	(s/1k steps)
SMoE	1.152	1.128	31.279	33.061	36.08 (0.04)	47.92	960.2
HyperMoE	1.162	1.139	31.918	33.374	48.41 (12.4)	49.69	962.0
SMoE-MLP	1.164	1.137	31.430	33.142	36.79 (0.75)	48.70	964.1
RandomSMoE	1.163	1.135	31.938	33.410	36.08 (0.04)	47.72	961.4
CosineMoE	1.148	1.122	31.367	33.047	36.08 (0.04)	48.68	962.4
XMoE	1.150	1.125	31.265	32.926	36.13 (0.09)	48.70	967.5
RMoE	**1.141**	**1.116**	**30.939**	**32.867**	36.51 (0.47)	49.46	972.9

Main Results

Table 3: SMoE and RMoE's pre-training costs and evaluation results in selected informative lm-evaluation-harness tasks. 'sft' means supervised fine-tuning on the Alpaca dataset. The task names and metrics for short names in the table are: '**ARC-e**' for ARC-Easy, acc; '**Hella**' is for Hellaswag, acc-norm; '**Piqa**' for PIQA, acc-norm; '**Lamb**' for LAMBADA, acc. Each model has approximately 0.53B activated parameters out-of 0.91B parameters. RMoE introduces about 3.5M additional parameters relative to SMoE.

Algorithm	Training	ARC-e	Hella	Piqa	Sciq	Lamb	Avg↑
SMoE Speed: 48.87 s/step Mem: 48.00 GB	pre-train 20B tokens	47.14	35.51	64.69	76.2	14.61	47.63
	+sft	50.93	35.82	65.61	74.7	17.81	48.97
	+sft (freeze router)	50.59	35.78	66.32	74.7	18.18	49.11
	pre-train 40B tokens	52.57	40.85	67.74	83.4	26.74	54.26
	+sft	53.70	42.07	68.61	83.5	32.80	56.13
	+sft (freeze router)	53.45	41.94	68.88	83.1	32.06	55.88
RMoE Speed: 49.07 s/step Mem: 48.69 GB	pre-train 20B tokens	47.01	35.91	65.23	78.7	19.13	49.20
	+sft	48.53	36.90	66.21	79.6	24.74	51.20
	+sft (freeze router)	49.24	36.79	66.16	79.7	24.32	51.24
	pre-train 40B tokens	51.18	41.38	67.79	83.6	32.58	55.31
	+sft	53.20	43.05	68.55	83.8	37.16	57.15
	+sft (freeze router)	53.11	43.16	68.77	82.8	37.57	57.08

Layerwise recurrence increases cross-layer mutual information

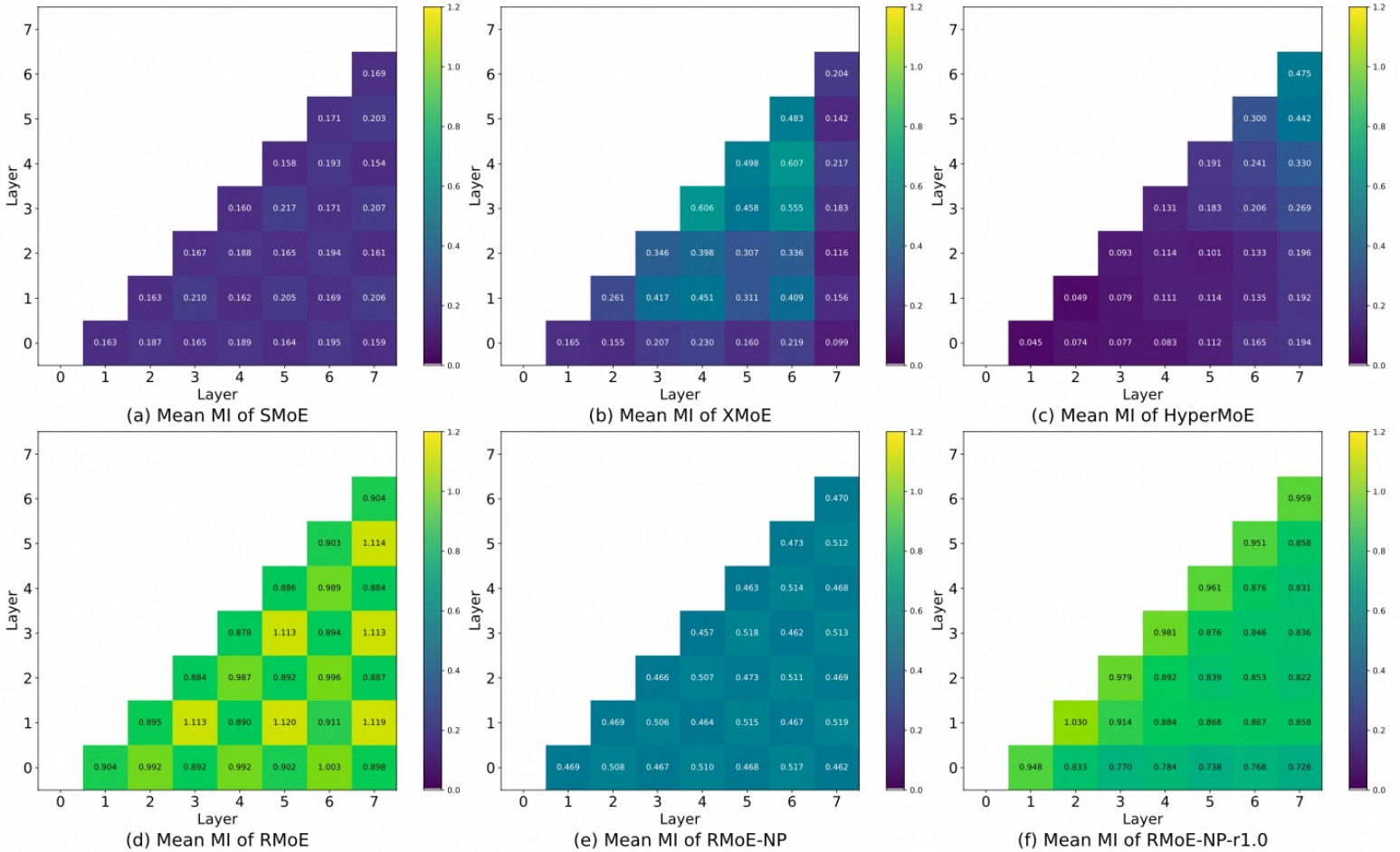

Figure 3: Heat maps of cross-layer mutual information (MI) for different methods. The (i-th row, j-th column) value represents MI between layers i and j. The **First Row** ((a) SMoE, (b) XMoE, (c) HyperMoE): All three methods have low cross-layer MI. **Second Row**((d) RMoE, (e) RMoE-NP, (f) RMoE-NP-r1.0): While RMoE has high cross-layer MI when disabled layerwise recurrent states passing, MI largely drops.

Layerwise recurrence encourages expert diversity

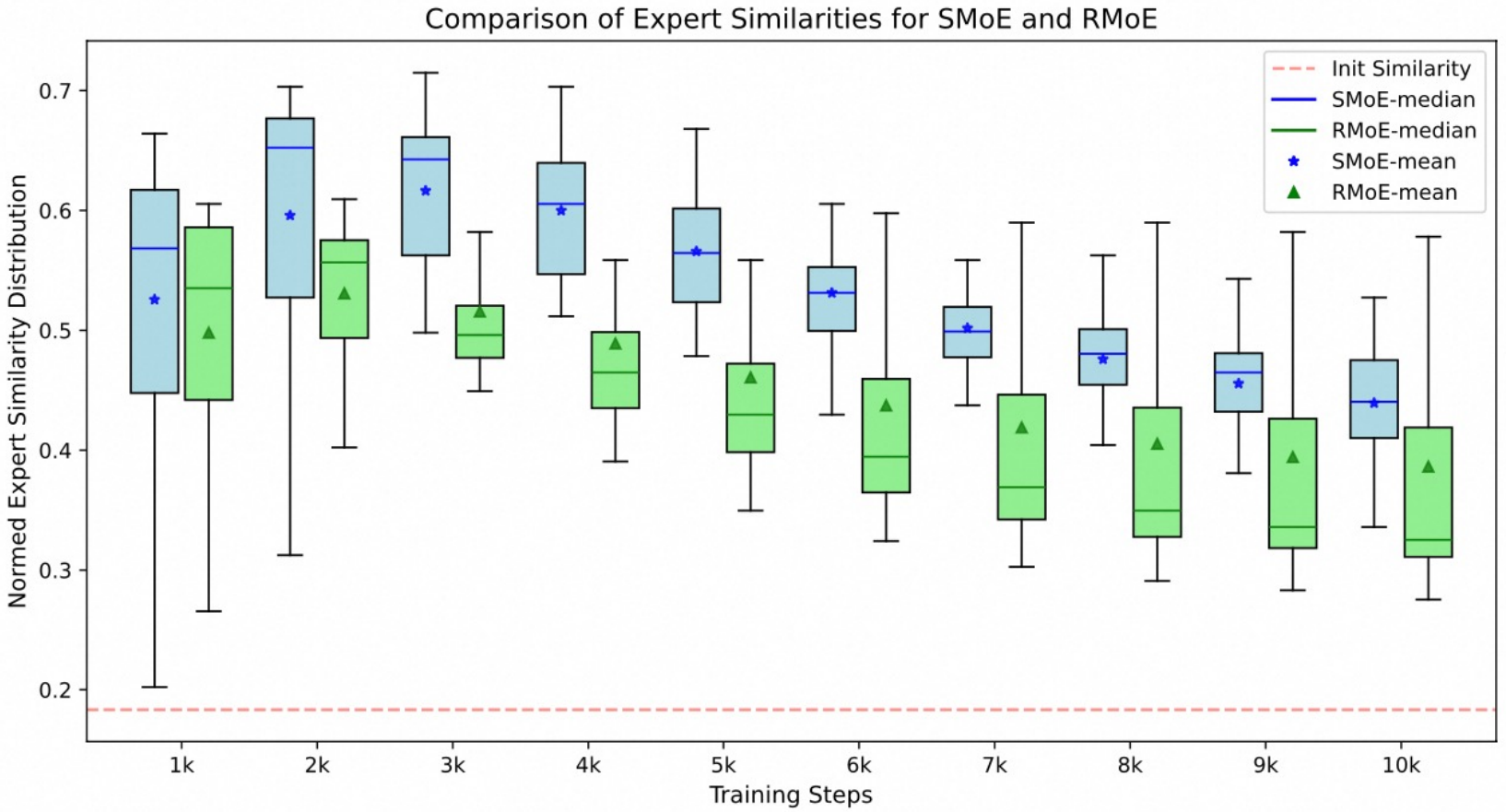

Figure 5: Experts similarity distribution across layers during large-scale pre-training. We plot box plots of expert similarity from checkpoints taken every 1k training steps (approximately 4B tokens), showing the expert similarity across the 24 layers of the model (with maximum, minimum, first quartile, median, and mean).

Thanks!