RRD: Routing-and-Residual Distillation for Efficient MoE Recovery in Large Language Models

Hoyoon Byun HOYUN.BYUN@YONSEI.AC.KR
Yonsei University, South Korea

Kangjun Noh SHWNDNJS58@YONSEI.AC.KR
Yonsei University, South Korea

Somin Kim 2021111050@YONSEI.AC.KR
Yonsei University, South Korea

Heedong Kim HEEDONGDONG@YONSEI.AC.KR
Yonsei University, South Korea

Jaeyoon Shim SJY21YS@YONSEI.AC.KR
Yonsei University, South Korea

Sungjun Lim LSJ9862@YONSEI.AC.KR
Yonsei University, South Korea

Youngjun Choi CHOIYJ9803@YONSEI.AC.KR
Yonsei University, South Korea

Kyungwoo Song* KYUNGWOO.SONG@YONSEI.AC.KR
Yonsei University, South Korea

Abstract

Mixture-of-Experts (MoE) architectures improve inference efficiency by activating only a small subset of parameters for each token. Recent dense-to-MoE conversion methods transform pretrained dense large language models into sparse MoEs through expert initialization, but practical top-K routing prevents the converted model from fully reproducing the original dense computation. We view this recovery gap as arising from two coupled challenges: selecting appropriate experts and recovering information missed by the selected experts. We propose *Routing-and-Residual Distillation* (RRD), a teacher-guided framework that distills routing targets from the original dense model and repurposes shared experts to recover the remaining representation gap. Experiments demonstrate that teacher-guided routing substantially improves sparse MoE recovery and that combining routing and residual recovery yields more faithful dense-to-MoE transfer.[1]

1. Introduction

Large language models (LLMs) improve with scale, but scaling increases inference latency, memory use, and energy cost [2, 4, 11, 22]. Mixture-of-Experts (MoE) architectures reduce this cost by activating only a small parameter subset per token, increasing conditional capacity without full dense computation [6, 13, 20]. Because training MoEs from scratch remains expensive, dense-to-MoE conversion upcycles pretrained dense LLMs by reusing dense MLP parameters as experts [7, 12, 15,

* Corresponding Author

1. Code: https://github.com/MLAI-Yonsei/routing_and_residual_distillation

© H. Byun, K. Noh, S. Kim, H. Kim, J. Shim, S. Lim, Y. Choi & K. Song*.

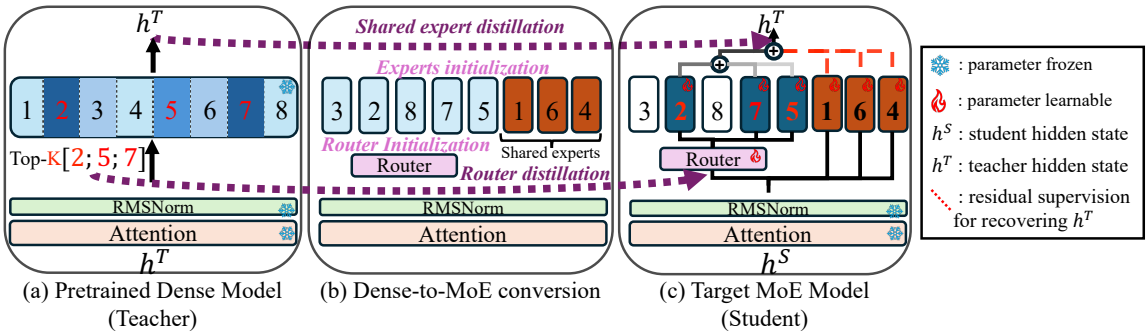

Figure 1: RRD combines split-initialized routed experts with shared residual correction, guided by teacher-derived top-K routing targets.

16, 18, 27]. Careful expert initialization can preserve much of the dense model's performance while enabling sparse inference.

Practical top-K routing, however, still leaves a recovery gap: each token activates only a subset of routed experts, whereas the dense model uses all feed-forward components. The converted model must therefore learn which experts to activate and how to recover intermediate representations not sufficiently preserved by them. Standard fine-tuning and logit-level distillation provide limited guidance, because language modeling only indirectly trains the router and output-level supervision does not specify expert selection or layer-wise recovery.

We propose *Routing-and-Residual Distillation* (RRD), illustrated in Figure 1, to use these dense-model signals for post-conversion recovery. RRD derives top-K routing targets from activation magnitudes within each expert group and distills them into the router. It also trains always-active shared experts to reconstruct the intermediate representations that routed experts alone do not sufficiently preserve. This design gives existing MoE components complementary roles: routed experts perform token-specific sparse computation, while shared experts reduce the remaining representation gap. RRD jointly optimizes language modeling with teacher-guided distillation of both routing decisions and intermediate representations.

2. Methodology

We study dense-to-MoE recovery under sparse top-K routing. Split initialization decomposes a dense MLP into disjoint expert slices, but top-K routing activates only a subset of them. The resulting MoE must therefore learn to route each token to informative experts while compensating for the representations lost by sparse top-K expert selection. We propose *Routing-and-Residual Distillation* (RRD), a teacher-guided framework that supervises expert routing and residual recovery using layer-wise signals from the frozen dense teacher.

2.1. Problem setup and split-initialized student

Consider a Transformer with L blocks and hidden dimension d. For each converted layer ℓ, let $f_\ell^T : \mathbb{R}^d \to \mathbb{R}^d$ denote the teacher dense MLP with intermediate dimension d_f. We denote the teacher and student residual-stream states by h_ℓ^T and h_ℓ^S, and their pre-MLP normalized states by

$\tilde{h}_\ell^T = \mathrm{LN}_\ell(h_\ell^T)$ and $\tilde{h}_\ell^S = \mathrm{LN}_\ell(h_\ell^S)$. For token position t, we write $h_{\ell,t}^T$ and $h_{\ell,t}^S$; when clear, we omit ℓ and t.

Split initialization partitions the teacher intermediate dimension into E disjoint slices of width $d_e = d_f/E$, defining slice-wise functions $\{f_{\ell,i}^T\}_{i=1}^E$. Under the usual row-disjoint split of SwiGLU-style MLPs, the teacher MLP admits the exact all-slice decomposition

$$f_\ell^T(\tilde{h}_\ell^T) = \sum_{i=1}^{E} f_{\ell,i}^T(\tilde{h}_\ell^T). \tag{1}$$

Thus, split initialization preserves the dense MLP only when all E slices are used, whereas top-K routing activates only $K < E$ experts.

The student replaces each dense MLP with a set of split-initialized experts $\{e_{\ell,i}\}_{i=1}^E$, a router $g_{\ell,\theta} : \mathbb{R}^d \to \mathbb{R}^E$, and an always-active shared expert $s_\ell : \mathbb{R}^d \to \mathbb{R}^d$. Each $e_{\ell,i}$ is initialized from the corresponding teacher slice $f_{\ell,i}^T$, and the shared expert has an intermediate dimension of d_e by default. We also consider a partition-based variant in which shared and routed experts are initialized from disjoint groups of the original dense MLP neurons [16].

Given $\tilde{h}_\ell^S$, the router produces probabilities $p_{\ell,\theta}(i \mid \tilde{h}_\ell^S) = \mathrm{softmax}(g_{\ell,\theta}(\tilde{h}_\ell^S))_i$ and selects $S_{\ell,\theta}(\tilde{h}_\ell^S) = \arg\mathrm{topK}_i\, g_{\ell,\theta}(\tilde{h}_\ell^S)_i$ with $|S_{\ell,\theta}| = K$. We renormalize the selected probabilities as

$$\tilde{p}_{\ell,\theta}(i \mid \tilde{h}_\ell^S) = p_{\ell,\theta}(i \mid \tilde{h}_\ell^S) / \sum_{j \in S_{\ell,\theta}} p_{\ell,\theta}(j \mid \tilde{h}_\ell^S) \quad \text{for } i \in S_{\ell,\theta}. \tag{2}$$

The student MoE block combines the normalized top-K routed path with the always-active shared expert:

$$\mathrm{MoE}_\ell(\tilde{h}_\ell^S) = \sum_{i \in S_{\ell,\theta}(\tilde{h}_\ell^S)} \tilde{p}_{\ell,\theta}(i \mid \tilde{h}_\ell^S) e_{\ell,i}(\tilde{h}_\ell^S) + s_\ell(\tilde{h}_\ell^S) = \mathrm{RE}_\ell(\tilde{h}_\ell^S) + s_\ell(\tilde{h}_\ell^S). \tag{3}$$

Here, RE_ℓ denotes the routed, non-shared expert path. Equation 3 defines the student MoE computation, where the routed path performs sparse token-specific processing and the shared expert acts as an always-active correction path. The frozen dense model provides the teacher signals used in the following distillation losses.

2.2. Routing distillation

For each token, the teacher slices differ in their contribution to the dense MLP output. We use these slice activations to define a teacher-side top-K routing target. For t-th token at layer ℓ, we set

$$S_{\ell,t}^\star = \arg\mathrm{topK}_{i \in \{1,\ldots,E\}} \left\| f_{\ell,i}^T(\tilde{h}_{\ell,t}^T) \right\|_2. \tag{4}$$

This target is computed from the frozen teacher forward pass for each layer and token, providing direct supervision for expert selection.

The exact oracle would search over all $\binom{E}{K}$ subsets and choose the one whose sum best reconstructs the dense output, which is infeasible at scale. Equation 4 provides a practical surrogate under split initialization, where each expert slice corresponds to a disjoint subset of teacher MLP units.

We train the router to imitate this teacher-derived selection by converting $S^{\star}_{\ell,t}$ into a multi-hot vector $\mathbf{y}^{\star}_{\ell,t} \in \{0,1\}^{E}$, with ones at the selected experts. The routing loss is

$$\mathcal{L}_{\text{router}} = \text{BCE}\left(p_{\ell,\theta}(\cdot \mid \tilde{h}^{S}_{\ell,t}), \mathbf{y}^{\star}_{\ell,t}\right), \tag{5}$$

where BCE denotes binary cross-entropy over experts. This objective increases the probabilities of teacher-selected experts while suppressing those of the remaining experts.

2.3. Residual reconstruction with an always-active shared expert

Top-K routing cannot generally reproduce the full dense MLP output using routed experts alone. We therefore train the always-active shared experts as residual correction paths, whether they are implemented as additional modules [3, 25] or derived from partitions of the original dense MLP [16]. For t-th token at layer ℓ, we define the residual target as

$$r^{\star}_{\ell,t} = f^{T}_{\ell}(\tilde{h}^{T}_{\ell,t}) - \text{RE}_{\ell}(\tilde{h}^{S}_{\ell,t})_{\text{detach}}. \tag{6}$$

The first term is the dense teacher MLP output, and the detached second term is the routed component already produced by the selected experts. We train the shared expert with the layer-averaged RMSE loss

$$\mathcal{L}_{\text{shared}} = \frac{1}{L}\sum_{\ell=1}^{L}\sqrt{\mathbb{E}_{(x,t)}\left[\frac{1}{d}\left\|s_{\ell}(\tilde{h}^{S}_{\ell,t}) - r^{\star}_{\ell,t}\right\|_2^2\right]}. \tag{7}$$

The detach operation in Equation 6 restricts this loss to the shared expert.

2.4. Total objective and gradient routing

RRD combines language modeling, routing distillation, and shared-expert reconstruction:

$$\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_{\text{CE}} + \beta\,\mathcal{L}_{\text{router}} + \gamma\,\mathcal{L}_{\text{shared}}. \tag{8}$$

$\mathcal{L}_{\text{CE}}$ updates all trainable components, $\mathcal{L}_{\text{router}}$ updates the router and routed experts, and $\mathcal{L}_{\text{shared}}$ updates only the shared expert. We enforce this gradient assignment with detach operations, separating task, routing, and residual supervision while jointly optimizing the converted MoE.

3. Experiments & results

3.1. Experiment settings

We evaluate RRD on Llama-2-7B [24] and Qwen-2.5-7B [25]. Following CMoE [16], we convert dense MLPs into shared and routed experts using 64 calibration samples from WikiText-2 [14] and fine-tune all sparse models with the same 2,048 WikiText-2 samples. Our main evaluation uses E8A2S2, a 50% sparsity configuration with eight experts, two activated routed experts, and two shared experts per token. We compare RRD with training-free conversion and fine-tuning baselines under the same conversion and adaptation budget. Appendix B gives implementation details; Appendices C.2, and C.1 provide additional sparsity results and routing analysis.

Table 1: High-sparsity evaluation on Llama-2-7B and Qwen-2.5-7B with the E8A2S2 configuration. All methods use 50% sparsity, where each converted MoE layer activates half of the dense MLP computation. RRD uses the same 2K WikiText-2 adaptation budget as the fine-tuning baselines.

Method	Sparsity	PIQA	WinoGrande	ARC-E	ARC-C	HellaSwag	Avg. Accuracy	PPL WikiText-2	PPL C4
Llama-2-7B-E8A2S2									
CMoE	50%	59.19	55.25	41.96	25.94	44.00	45.27	14.02	20.88
CMoE + LoRA-FT	50%	*66.87*	**60.93**	*53.07*	30.89	55.58	*53.47*	*7.47*	*13.88*
CMoE + CE loss	50%	66.59	59.35	52.53	*32.25*	*56.63*	*53.47*	7.92	15.58
CMoE + RRD	50%	**67.41**	*59.51*	**55.13**	**32.42**	**56.84**	**54.26**	**7.42**	**13.00**
Qwen-2.5-7B-E8A2S2									
CMoE	50%	61.10	53.75	48.06	28.50	43.42	46.97	16.51	36.80
CMoE + LoRA-FT	50%	*63.49*	55.41	*54.67*	**36.09**	50.80	52.09	15.49	30.63
CMoE + CE loss	50%	**63.98**	*56.43*	54.38	33.79	**52.42**	*52.20*	*13.24*	*28.31*
CMoE + RRD	50%	**63.98**	**58.80**	**54.88**	*34.30*	*51.95*	**52.78**	**10.59**	**23.69**

Table 2: Full ablation study of RRD loss components on Llama-2-7B under the E8A2S2 setting. We remove each objective from the full RRD loss and report zero-shot accuracy on each benchmark.

	α	β	γ	PIQA	WinoG.	ARC-E	ARC-C	HellaS.	Avg.
w/o Cross-entropy loss	0.0	1.0	1.0	66.54	57.38	52.23	28.75	53.54	51.69
w/o Routing distillation	0.1	0.0	1.0	65.29	56.83	52.02	30.38	56.30	52.16
w/o Shared experts distillation	0.1	1.0	0.0	66.38	57.30	53.24	30.72	**57.10**	52.95
RRD	0.1	1.0	1.0	**67.41**	**59.51**	**55.13**	**32.42**	56.84	**54.26**

3.2. Main results

Table 1 evaluates RRD in the 50% sparsity regime, where top-K routing leaves a larger gap between the converted MoE and the original dense MLP. Under this setting, RRD improves over both training-free conversion and standard fine-tuning baselines. On Llama-2-7B, RRD reduces C4 perplexity from 20.88 in the training-free model to 13.00 and achieves the best average downstream accuracy. On Qwen-2.5-7B, RRD reduces C4 perplexity to 23.69, outperforming training-free conversion and LoRA fine-tuning under the same 2K-sample adaptation budget. These results indicate that teacher-guided routing and residual recovery are most beneficial when sparse routing removes a larger fraction of the dense MLP computation.

3.3. Effect of each RRD loss component

Table 2 shows that each component of RRD contributes to sparse recovery. Removing cross-entropy causes the largest drop, confirming the need for task-level supervision. Removing either routing distillation or residual reconstruction also degrades performance, suggesting that language modeling alone does not fully capture the expert-selection and representation-recovery signals needed for sparse MoE recovery. The full RRD objective achieves the best average zero-shot accuracy, supporting the complementary roles of cross-entropy, teacher-guided routing, and shared-expert residual recovery.

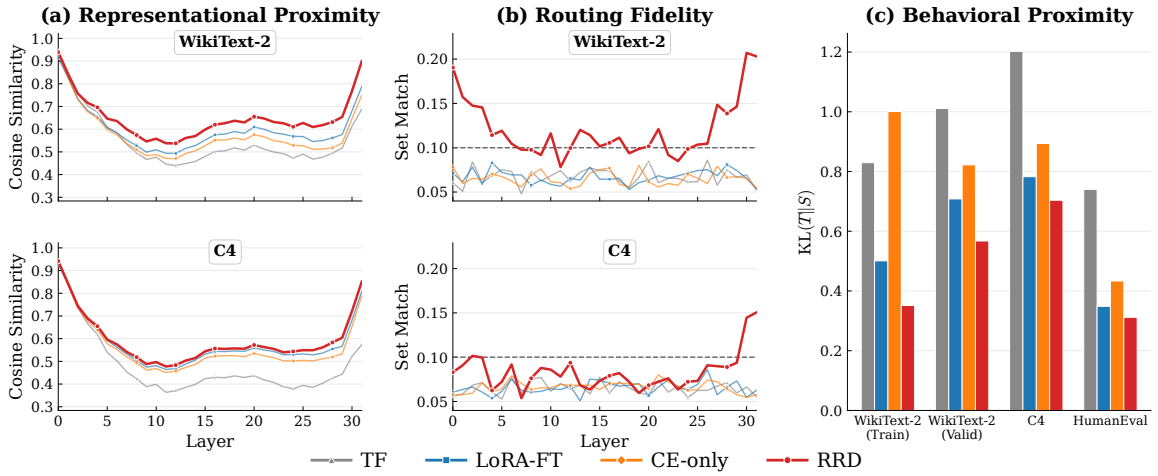

Figure 2: Multi-axis proximity analysis between CMoE-converted Llama-2-7B students and the dense teacher in E8A2S2. TF, LoRA-FT, and CE-only denote training-free conversion, LoRA fine-tuning, and cross-entropy-only fine-tuning, respectively. **(a)** Layer-wise cosine similarity between student MoE and teacher MLP outputs; **(b)** layer-wise top-K expert-set match rate against the teacher magnitude oracle, with the dashed line indicating random routing; **(c)** token-level $\mathrm{KL}(T \parallel S)$, where T and S denote teacher and student logits. RRD improves alignment between the sparse student and the dense teacher across representations, routing decisions, and output distributions.

3.4. Proximity to the dense teacher

Figure 2 explains why RRD improves high-sparsity recovery. Compared with training-free conversion and fine-tuning baselines, RRD yields higher MoE–MLP cosine similarity, higher top-K agreement with the teacher magnitude oracle, and lower token-level $\mathrm{KL}(T \parallel S)$. These trends show that RRD does not merely improve downstream scores, but also aligns the converted MoE with the dense teacher along the three axes targeted by the method: intermediate representations, routing decisions, and output behavior.

4. Related work

RRD relates to dense-to-MoE conversion, MoE upcycling, and intermediate knowledge distillation. Prior work reuses dense checkpoints through continued training, MLP-to-expert decomposition, pruning-based conversion, or activation-aware initialization [7, 12, 15–18, 26, 27], while distillation methods align final predictions or intermediate representations [5, 8–10, 19, 23]. RRD differs by targeting post-conversion recovery under sparse top-K routing, supervising both expert selection and residual reconstruction from dense-model activations.

5. Conclusion

We presented RRD, a dense-to-MoE recovery framework that decomposes distillation into routing supervision for expert selection and residual supervision for shared experts. Experiments show that this design brings sparse MoE students closer to the dense teacher across routing, representations, and outputs, and that such proximity leads to stronger high-sparsity recovery.

6. Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (RS-2024-00457216).

References

[1] Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari Do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*, 2024.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[3] 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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, 2024.

[4] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

[5] Sayantan Dasgupta and Trevor Cohn. Improving language model distillation through hidden state matching. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

[7] Shangqian Gao, Ting Hua, Reza Shirkavand, Chi-Heng Lin, Zhen Tang, Zhengao Li, Longge Yuan, Fangyi Li, Zeyu Zhang, Alireza Ganjdanesh, et al. Tomoe: Converting dense large language models to mixture-of-experts through dynamic structural pruning. *arXiv preprint arXiv:2501.15316*, 2025.

[8] Guoqiang Gong, Jiaxing Wang, Jin Xu, Deping Xiang, Zicheng Zhang, Leqi Shen, Yifeng Zhang, JunhuaShu JunhuaShu, ZhaolongXing ZhaolongXing, Zhen Chen, et al. Beyond logits: Aligning feature dynamics for effective knowledge distillation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23067–23077, 2025.

[9] Md Akmal Haidar, Nithin Anchuri, Mehdi Rezagholizadeh, Abbas Ghaddar, Philippe Langlais, and Pascal Poupart. Rail-kd: Random intermediate layer mapping for knowledge distillation. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1389–1400, 2022.

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[11] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[12] 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. *arXiv preprint arXiv:2212.05055*, 2022.

[13] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

[14] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[15] Taishi Nakamura, Takuya Akiba, Kazuki Fujii, Yusuke Oda, Rio Yokota, and Jun Suzuki. Drop-upcycling: Training sparse mixture of experts with partial re-initialization. *arXiv preprint arXiv:2502.19261*, 2025.

[16] Zehua Pei, Hui-Ling Zhen, Lancheng Zou, Xianzhi Yu, Wulong Liu, Sinno Jialin Pan, Mingxuan Yuan, and Bei Yu. Analytical ffn-to-moe restructuring via activation pattern analysis. *arXiv preprint arXiv:2502.04416*, 2026.

[17] Zihan Qiu, Zeyu Huang, and Jie Fu. Unlocking emergent modularity in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2638–2660, 2024.

[18] Xiaoye Qu, Daize Dong, Xuyang Hu, Tong Zhu, Weigao Sun, and Yu Cheng. Llama-moe v2: Exploring sparsity of llama from perspective of mixture-of-experts with post-training. *arXiv preprint arXiv:2411.15708*, 2024.

[19] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. arxiv 2014. *arXiv preprint arXiv:1412.6550*, 2014.

[20] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[21] Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. In *International Conference on Machine Learning*, pages 46136–46155. PMLR, 2024.

[22] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 3645–3650, 2019.

[23] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*, 2019.

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

[25] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[26] Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Moefication: Transformer feed-forward layers are mixtures of experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 877–890, 2022.

[27] 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. *arXiv preprint arXiv:2406.16554*, 2024.

Appendix

Appendix A. RRD algorithm

Algorithm 1: Routing-and-Residual Distillation (RRD) training step

Data: minibatch (x, y), frozen dense teacher $\{f_\ell^T\}_{\ell=1}^L$, split-initialized experts $\{e_{\ell,i}\}$, routers $\{g_{\ell,\theta}\}$, shared experts $\{s_\ell\}$, top-K, weights α, β, and γ

Set oracle targets $\{S_{\ell,t}^\star\}_t \leftarrow \arg\operatorname{topK}_{i\in\{1,\dots,E\}} \|f_{\ell,i}^T(\tilde{h}_{\ell,t}^T)\|_2$ for all token positions t;

for $\ell = 1$ **to** L **do**

  Compute teacher slice outputs $\{f_{\ell,i}^T(\tilde{h}_\ell^T)\}_{i=1}^E$;

  Compute dense teacher output $f_\ell^T(\tilde{h}_\ell^T) = \sum_{i=1}^E f_{\ell,i}^T(\tilde{h}_\ell^T)$;

  Set oracle target $S_\ell^\star \leftarrow \operatorname{topK}_i \|f_{\ell,i}^T(\tilde{h}_\ell^T)\|_2$;

end

Run the student forward and obtain $\{\tilde{h}_\ell^S, \mathrm{MoE}_\ell, s_\ell\}_{\ell=1}^L$ and the final logits;

for $\ell = 1$ **to** L **do**

  Compute routed-path loss $\mathcal{L}_{\mathrm{router},\ell}$ using $S_\ell^\star$, $p_{\ell,\theta}$, and the non-shared MoE path;

  Compute shared loss from $s_\ell(\mathrm{sg}[\tilde{h}_\ell^S])$ and $\mathrm{sg}\Big[f_\ell^T(\tilde{h}_\ell^T) - \mathrm{MoE}_\ell(\tilde{h}_\ell^S)\Big]$;

end

Compute language-modeling loss $\mathcal{L}_{\mathrm{CE}}$ on the final logits;

$\mathcal{L}_{\mathrm{total}} \leftarrow \alpha\mathcal{L}_{\mathrm{CE}} + \beta\mathcal{L}_{\mathrm{router}} + \gamma\mathcal{L}_{\mathrm{shared}}$;

Update the student parameters θ using $\nabla\mathcal{L}_{\mathrm{total}}$;

Appendix B. Experimental Setup

Models and conversion. We evaluate on Llama-2-7B [24] and Qwen-2.5-7B [25]. For each model, we follow the CMoE [16] dense-to-MoE conversion setting. CMoE analytically restructures each dense MLP into shared experts and routed experts using activation statistics. Frequently activated neurons are assigned to shared experts, while the remaining neurons are grouped into routed experts based on activation-pattern similarity. The router is initialized from representative neurons of the routed experts rather than from random weights. Following CMoE, we use 8 or 64 WikiText-2 [14] samples as the calibration set for conversion.

Sparsity settings. We build eight-expert sparse MoE models with the activation-pattern partitioning of CMoE [16]. Our primary high-sparsity setting in the main text is E8A2S2, where each layer has two shared experts and activates two routed experts per token, corresponding to 50% sparsity. We additionally evaluate E8A3S3, a milder 25% sparsity setting, in Appendix C.2. We denote the number of shared experts by n_s and the number of activated routed experts by n_a.

Training recipe. All fine-tuned methods use the same data budget as CMoE: 2,048 samples from WikiText-2 raw text. We use sequence length 2048, one training epoch, Adam, effective batch size 2, and a constant learning-rate schedule. LoRA follows the original CMoE recipe and applies adapters to both attention and MLP modules while also training the router with a load-balancing objective.

CE-only and RRD update only the router and experts, while keeping attention and embeddings frozen. RRD does not use an additional router load-balancing loss.

Hyperparameter settings. For LoRA, we follow CMoE and use `lora_extra_lr`=0.001, `lora_r`=8, `lora_alpha`=32, and `lora_dropout`=0.1. We tune the base learning rate over $\{3 \times 10^{-5}, 10^{-4}\}$ for all methods, and additionally evaluate 5.95×10^{-5} for LoRA following the original paper. For RRD, we search $\alpha_{\text{task}} \in \{0.1, 0.3, 0.7\}$, $\alpha_{\text{shared}} \in \{1.0, 3.0, 5.0, 10.0\}$, and $\alpha_{\text{router}} \in \{0.1, 0.5, 1.0\}$.

Baselines. We follow the CMoE benchmark setting and compare against both structured pruning and dense-to-MoE conversion methods. Dense is the original unsparsified model. SliceGPT [1] and SLEB [21] are structured pruning baselines that remove redundant model components from the pretrained dense model. LLaMA-MoE [27] and LLaMA-MoE-v2 [18] convert dense MLPs into routed experts through parameter partitioning and post-training adaptation. EMoE [17] builds experts using parameter-based neuron clustering. CMoE [16] separates frequently activated neurons from input-dependent neurons and forms shared and routed experts from the original dense MLP. RRD is applied to the CMoE-converted architecture.

Evaluation and compute. We report zero-shot downstream accuracy on PIQA, WinoGrande, ARC-Easy, ARC-Challenge, and HellaSwag. For the high-sparsity setting, we also report perplexity on WikiText-2 and C4. All runs use a single 95 GiB GPU with bfloat16 mixed precision in PyTorch.

Appendix C. Additional Experimental Results

C.1. Routing signals after dense-to-MoE conversion

Table 3: Effect of routing signals after dense-to-MoE conversion. Magnitude-based routing selects experts using dense MLP activation magnitudes. Compared with random routing, activation-based expert selection substantially improves performance, and teacher representations provide stronger routing signals than student representations.

Router	Input	Sparsity	ARC-E	HellaSwag.	PIQA	Avg. Accuracy
Random	h^S	50%	26.60	25.79	50.76	34.38
Magnitude-based	h^S	50%	34.30	32.87	57.34	41.50
Magnitude-based	h^T	50%	**37.88**	**34.79**	**57.73**	**43.47**

To motivate routing distillation, we study dense-to-MoE conversion on Qwen3-1.7B without post-conversion training. Each dense MLP is evenly split into eight experts, and four experts are activated per token. We compare random routing with magnitude-based routing, which selects experts by dense MLP activation magnitudes, and evaluate whether the routing signal is computed from the student representation h^S or the teacher representation h^T. Table 3 shows that magnitude-based routing substantially improves average accuracy over random routing, from 34.38 to 41.50 with h^S and 43.47 with h^T. These results indicate that dense activations provide useful expert-selection signals and that teacher-side representations yield stronger routing targets, motivating RRD's teacher-guided routing objective.

C.2. Results under 25% sparsity

Table 4: Zero-shot downstream performance on Llama-2-7B and Qwen-2.5-7B under the standard 25% sparsity setting. All sparsified models are fine-tuned with 2K WikiText-2 samples, while the dense baseline is not fine-tuned. Converted MoE models use 25% sparsity, and structured pruning baselines use 20% reduction. RRD denotes our method applied to the CMoE-converted architecture. Baseline results are from CMoE [16].

Method	Sparsity	PIQA	WinoGrande	ARC-E	ARC-C	HellaSwag	Avg. Accuracy
Llama-2 7B							
Dense	0%	78.78	69.06	74.58	46.16	76.00	68.92
SliceGPT	20%	65.71	62.88	59.76	33.21	51.34	54.58
SLEB	20%	73.13	58.98	57.90	33.02	62.47	57.10
LLaMA-MoE	25%	49.35	50.28	54.04	26.37	25.77	41.16
LLaMA-MoE-v2	25%	63.55	59.35	63.77	34.81	54.89	55.27
EMoE	25%	72.47	64.48	58.63	35.75	60.80	58.43
CMoE + LoRA-FT	25%	**74.34**	*65.77*	**67.09**	40.35	69.36	*63.38*
CMoE + RRD	25%	*72.91*	**66.22**	*67.05*	**41.30**	**70.93**	**63.68**
Qwen-2.5-7B							
Dense	0%	79.82	73.16	77.36	51.02	78.86	72.04
SliceGPT	20%	66.19	*66.51*	61.88	36.69	53.21	56.90
SLEB	20%	74.95	61.76	59.95	35.80	64.41	59.37
LLaMA-MoE	25%	49.63	53.21	57.05	28.64	25.65	42.84
LLaMA-MoE-v2	25%	64.25	62.71	65.77	37.59	56.06	57.28
EMoE	25%	73.98	65.41	60.63	38.48	62.71	60.24
CMoE + LoRA-FT	25%	75.93	**69.36**	*70.59*	*43.86*	**72.21**	**66.39**
CMoE + RRD	25%	73.99	66.06	**72.52**	**47.53**	*69.36*	*65.89*

Table 4 reports results under the standard E8A3S3 setting, where each converted MoE layer retains 75% of the dense MLP computation. In this relatively mild sparsity regime, CMoE initialization already preserves much of the dense model's behavior. RRD remains competitive with strong dense-to-MoE baselines, slightly improving average accuracy on Llama-2-7B and maintaining comparable performance on Qwen-2.5-7B. These results complement the high-sparsity experiments in the main text, where the benefit of routing and residual distillation becomes more pronounced.