RRD: ROUTING-AND-RESIDUAL DISTILLATION FOR RECOVERY AFTER DENSE-TO-MOE CONVERSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-Experts (MoE) models reduce per-token feed-forward network (FFN) computation by activating only a subset of experts. Dense-to-MoE conversion reuses pretrained dense weights, but sparse top-K execution can create a gap between the converted student and its dense teacher. We formulate recovery under a fixed activation budget as two linked objectives: selecting useful experts and matching layerwise FFN outputs. Cross-entropy (CE) and logit-based knowledge distillation (logit-KD) supervise next-token predictions but do not directly constrain either objective. We propose Routing-and-Residual Distillation (RRD), which derives top-K routing targets from teacher neuron activations and matches the realized sparse FFN output to the corresponding dense teacher FFN output. RRD combines these internal objectives with CE and logit-KD while retaining the converted expert partition and fixed activation budget. Across Qwen2.5-7B and Llama-2-7B, RRD achieves the highest mean zero-shot accuracy among the evaluated methods at both 25% and 50% FFN sparsity. On Qwen2.5-7B at 25% sparsity, RRD reaches 68.40% mean zero-shot accuracy after training on about 40M C4 tokens, outperforming the strongest same-topology baseline by 3.13 percentage points. At 50% sparsity, it outperforms the strongest compute-matched baseline by 2.47 percentage points under an approximately matched training-FLOP budget that includes the frozen teacher's forward pass. These results support direct supervision of expert selection and layerwise FFN outputs during sparse recovery. [1]

1 INTRODUCTION

Large language models (LLMs) continue to improve with scale, but this improvement comes with higher inference latency, memory use, and energy cost (Brown et al., 2020; Kaplan et al., 2020; Strubell et al., 2019). Mixture-of-Experts (MoE) architectures reduce active per-token feed-forward network (FFN) computation by routing each token to only a subset of experts, thereby increasing conditional capacity without incurring the full dense cost (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022). However, training an MoE from scratch still requires large-scale pretraining. Dense-to-MoE conversion offers a practical alternative by initializing sparse experts from the FFN parameters of a pretrained dense LLM (Komatsuzaki et al., 2022; Zhu et al., 2024; Qu et al., 2024; Nakamura et al., 2025; Gao et al., 2025).

Existing Dense-to-MoE methods differ in how they construct experts and initialize or learn routing. LLaMA-MoE partitions pretrained FFNs into experts (Zhu et al., 2024), while CMoE and ExpertWeaver use activation patterns to form shared and routed experts (Pei et al., 2026; Zhao et al., 2026). DOT-MoE jointly learns neuron assignment and token routing (Bamba et al., 2026). Once conversion is complete, however, sparse execution activates only a subset of the converted FFN partitions for each token and can deviate from the dense model. We focus on this post-conversion recovery stage: given a fixed expert partition and per-token activation budget, how can the sparse student recover the behavior of its dense teacher?

We formulate this recovery problem as two linked objectives. First, the router should select experts that reflect the teacher's token-specific activation pattern. Second, the resulting sparse FFN, including

[1] Code is available at `https://anonymous.4open.science/r/routing_and_residual_distillation-7226/`

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

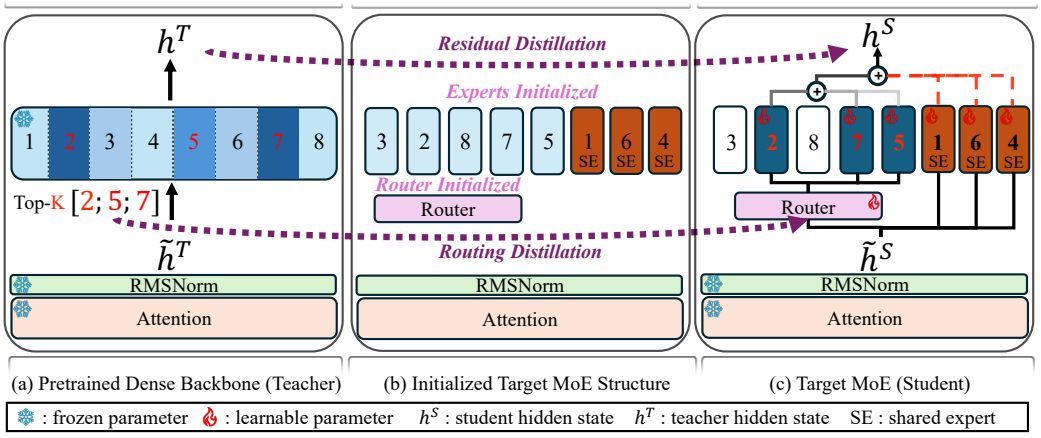


Figure 1: Overview of RRD for post-conversion MoE recovery. Routing distillation derives token-specific top-K targets from teacher activation mass. Residual distillation aligns the realized sparse FFN output, produced by the selected routed experts and any shared experts, with the dense teacher FFN output. The expert partition and activation budget remain fixed.

any shared path, should produce a layerwise output close to the dense teacher's FFN output. Standard cross-entropy (CE) training and logit-based knowledge distillation (Logit-KD) supervise next-token predictions, but they do not directly specify either internal target. The frozen dense teacher provides neuron activations and layerwise FFN outputs that can supervise both objectives directly.

We therefore propose **Routing-and-Residual Distillation (RRD)**. As illustrated in Figure 1, RRD directly supervises both recovery objectives while keeping the converted expert partition and activation budget fixed. Routing distillation uses the inherited neuron-to-expert mapping and aggregates teacher activation magnitudes within each routed group to construct token-specific top-K routing targets. Residual distillation minimizes the layerwise gap between the dense teacher FFN output and the student's realized sparse FFN output, which combines the selected routed experts with any shared path. Together with CE and Logit-KD, these objectives supervise expert selection, internal sparse computation, and next-token prediction.

We evaluate RRD on Qwen2.5-7B (Yang et al., 2024) and Llama-2-7B (Touvron et al., 2023) after approximately 40M C4 tokens at 25% and 50% FFN sparsity. RRD achieves the highest mean zero-shot accuracy among the evaluated methods across both backbones and sparsity levels. On Qwen2.5-7B, it outperforms CMoE with LoRA by 3.13 percentage points at 25% sparsity under E8S3A3 and DOT-MoE by 2.47 points at 50% sparsity under approximately matched training FLOPs that include the frozen teacher's forward pass. We further evaluate teacher-free SFT, a random-split LLaMA-MoE partition, loss ablations, and teacher–student proximity.

Our contributions are as follows. First, we formulate post-conversion Dense-to-MoE recovery under a fixed activation budget as expert selection and layerwise sparse FFN output alignment. Second, we introduce RRD, which combines routing and residual distillation with CE and Logit-KD while retaining the converted expert partition. Third, we validate RRD across two 7B backbones, two sparsity levels, token- and FLOP-matched CPT, teacher-free SFT, and an alternative expert partition, together with loss and teacher–student proximity analyses.

2 Related Work

Dense-to-MoE conversion and expert construction. Dense-to-MoE conversion reuses parameters of a pretrained dense model to form sparse experts. Sparse Upcycling and Drop-Upcycling adapt dense checkpoints into MoEs through continued training (Komatsuzaki et al., 2022; Nakamura et al., 2025). MoEfication partitions FFNs into functional experts and constructs routers for input-dependent expert selection (Zhang et al., 2022). LLaMA-MoE and LLaMA-MoE v2 study expert construction and post-conversion adaptation (Zhu et al., 2024; Qu et al., 2024). CMoE uses neuron activation statistics to form shared and routed experts and initialize the router (Pei et al., 2026). ToMoE uses

dynamic structural pruning during conversion (Gao et al., 2025), while ExpertWeaver uses GLU activation patterns to construct layer-adaptive shared and routed experts (Zhao et al., 2026). DOT-MoE jointly optimizes neuron assignment and token routing using differentiable optimal transport (Bamba et al., 2026). RRD takes a converted expert partition as given and focuses on recovering dense-teacher behavior under sparse execution.

Distillation for sparse MoE recovery. Knowledge distillation transfers information from teacher predictions (Hinton et al., 2015), while intermediate supervision can align internal representations (Romero et al., 2015; Sun et al., 2019; Haidar et al., 2022; Dasgupta & Cohn, 2025; Gong et al., 2025). LaDiMo uses layerwise distillation during Dense-to-MoE conversion (Kim et al., 2024), and LLaVA-MoD applies progressive distillation to MoE-based multimodal compression (Shu et al., 2024). Given a split-initialized student, RRD uses teacher neuron activations to define top-K routing targets and matches the combined output of shared and selected routed experts to the dense teacher FFN output. These internal losses are combined with CE and Logit-KD during student training.

Router training. Top-K selection is discrete, so the output loss gives no gradient to unselected experts. Our converted MoE also sums the selected experts without gate weights, so the router receives no output-loss gradient at all. Most router objectives instead regularize the routing distribution. The load-balancing loss spreads tokens evenly across experts (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022), and the router z-loss penalizes large logits for stability (Zoph et al., 2022). ReMoE controls router sparsity and load with an adaptive L_1 penalty (Wang et al., 2025). Orthogonality and variance losses counter the overly uniform routing that load balancing can induce (Guo et al., 2025). These objectives shape routing statistics but do not specify which experts each token should use. StableMoE supplies explicit targets by distilling the model's own learned routing into a lightweight router (Dai et al., 2022). MoEfication trains an activation-supervised router on sampled dense-model representations (Zhang et al., 2022), RRD instead applies teacher-derived routing supervision throughout sparse recovery. It supervises the router on the student's own hidden states, and this loss is the router's only update signal.

3 METHODOLOGY

We study recovery after converting a dense Transformer MLP into a sparsely activated mixture of experts (MoE). Sparse execution creates two linked problems: selecting routed experts for each token and reducing the gap between the resulting MoE output and the dense teacher MLP output. The MoE output combines the outputs of the selected routed experts and any shared experts. We address these problems with *Routing-and-Residual Distillation* (RRD), using a frozen dense teacher to supervise expert selection and layerwise MLP outputs.

3.1 PROBLEM SETUP AND SPLIT-INITIALIZED STUDENT

Consider a Transformer with L blocks and hidden dimension d. At converted layer ℓ, the frozen teacher has a dense MLP $f_\ell^T : \mathbb{R}^d \to \mathbb{R}^d$ with intermediate width d_f.

Let $h_{\ell,t}^T$ and $h_{\ell,t}^S$ be the teacher and student residual-stream states at token t, and let $\tilde{h}_{\ell,t}^T$ and $\tilde{h}_{\ell,t}^S$ be the corresponding states after the pre-MLP normalization. We omit ℓ and t when the context is clear.

We start from a converted MoE whose d_f intermediate neurons are partitioned into E disjoint, equal-width groups of width $d_e = d_f/E$. We combine S groups into an always-active shared module s_ℓ and use the remaining $E_\mathrm{r} = E - S$ groups as routed experts. When $S = 0$, the shared path is absent. Because the output projection is additive over disjoint intermediate groups, the teacher satisfies

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

At initialization, the S shared slices initialize s_ℓ, and the remaining E_r slices initialize $e_{\ell,i}$. The sum of these initialized modules reproduces the dense MLP when evaluated on the same input. During sparse execution, the student activates $K \le E_\mathrm{r}$ routed experts per token, along with s_ℓ when $S > 0$.

The student router $g_{\ell,\theta} : \mathbb{R}^d \to \mathbb{R}^{E_\mathrm{r}}$ produces logits that determine the top-K set:

$$\mathcal{S}_{\ell,\theta,t} = \mathrm{TopK}_K\Big(g_{\ell,\theta}(\tilde{h}^S_{\ell,t})\Big). \tag{2}$$

Here TopK_K returns the indices of the K largest components of a vector, so $\mathcal{S}_{\ell,\theta,t} \subseteq \{1,\dots,E_\mathrm{r}\}$ and $|\mathcal{S}_{\ell,\theta,t}| = K$. The routed outputs use an unweighted sum of the selected experts, as in the converted model we study:

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

For a shared-free architecture, $s_\ell \equiv 0$. Unlike Equation 1, Equation 3 uses only selected routed groups and the student's current hidden state.

3.2 Routing distillation

The split preserves a mapping between teacher intermediate neurons and routed experts. Let $\mathcal{N}_{\ell,i}$ be the teacher neuron indices assigned to routed expert i, and let

$$a^T_{\ell,j,t} = \phi\Big((W^T_{\ell,\mathrm{gate}}\tilde{h}^T_{\ell,t})_j\Big)\,(W^T_{\ell,\mathrm{up}}\tilde{h}^T_{\ell,t})_j$$

denote the activation of teacher intermediate neuron j before the output projection, where ϕ is the MLP activation function. We score each routed group by its absolute activation mass and choose the teacher-derived top-K set:

$$m^T_{\ell,i,t} = \sum_{j\in\mathcal{N}_{\ell,i}} |a^T_{\ell,j,t}|, \qquad S^\star_{\ell,t} = \mathrm{TopK}_K\Big((m^T_{\ell,i,t})_{i=1}^{E_\mathrm{r}}\Big). \tag{4}$$

This target uses the magnitudes of the teacher's intermediate activations as a proxy for useful routed-expert choices. It does not require enumerating expert subsets or assuming that activation mass identifies the output-optimal subset.

We supervise the router with top-K target cross-entropy. Let $\tilde{h}^{S,\mathrm{TF}}_{\ell,t}$ denote the student pre-MLP state in a separate router-supervision pass, in which preceding layers execute the teacher-selected routed experts and any shared experts. For this loss, we normalize router logits as $p_{\ell,\theta}(i \mid \tilde{h}^{S,\mathrm{TF}}_{\ell,t}) = \mathrm{softmax}(g_{\ell,\theta}(\tilde{h}^{S,\mathrm{TF}}_{\ell,t}))_i$:

$$\mathcal{L}_\mathrm{router} = -\mathbb{E}_{\ell,t}\left[\frac{1}{K}\sum_{i\in S^\star_{\ell,t}} \log p_{\ell,\theta}(i \mid \tilde{h}^{S,\mathrm{TF}}_{\ell,t})\right]. \tag{5}$$

The targets are computed from the frozen teacher, while the probabilities come from the student router. This loss supplies a direct learning signal for expert selection.

3.3 Residual distillation through MLP-output matching

At each layer and token, we define the *residual* as the output gap between the dense teacher MLP and the student's realized sparse MLP:

$$\varepsilon_{\ell,t} = f^T_\ell(\tilde{h}^T_{\ell,t}) - \mathrm{MoE}_\ell(\tilde{h}^S_{\ell,t}). \tag{6}$$

Because the teacher and student outputs are evaluated on their respective hidden states, this gap reflects omitted expert contributions, changes in active expert outputs, and differences between the teacher and student states. We minimize its layerwise root-mean-square error (RMSE), averaging over tokens and hidden coordinates within each layer and then averaging across layers:

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

1: **Input:** minibatch (x, y); frozen dense teacher; split-initialized student; teacher-neuron-to-expert mapping; top-K budget; α, β, γ
2: Run frozen teacher on x; collect intermediate activations, dense MLP outputs, and logits.
3: **for** $\ell = 1$ **to** L **do**
4: Form $S^{\star}_{\ell,t}$ from the activation mass of each routed group for every token t (Equation 4).
5: **end for**
6: Run sparse student on x; collect MoE_ℓ and final logits.
7: Compute $\mathcal{L}_{\mathrm{CE}}$, $\mathcal{L}_{\mathrm{residual}}$, and $\mathcal{L}_{\text{logit-KD}}$.
8: Accumulate gradients from these losses only with respect to the expert parameters, using weights 1, β, and γ.
9: Run a second student pass. At each layer, evaluate the router on its student pre-MLP state; execute the shared and teacher-selected routed experts on that same state without expert gradients, and pass their output to the next layer.
10: Compute $\mathcal{L}_{\mathrm{router}}$ (Equation 5); accumulate $\alpha \nabla \mathcal{L}_{\mathrm{router}}$ only for router parameters.
11: Update student expert and router parameters with their respective learning rates.

$$\mathcal{L}_{\mathrm{residual}} = \frac{1}{L} \sum_{\ell=1}^{L} \underbrace{\sqrt{\mathbb{E}_{(x,t)} \left[\frac{1}{d} \|\varepsilon_{\ell,t}\|_2^2 \right]}}_{\mathrm{RMSE}_\ell} . \tag{7}$$

3.4 Total objective and gradient routing

Let $\mathcal{L}_{\mathrm{CE}}$ be next-token cross-entropy. For teacher and student logits z_t^T, z_t^S and temperature τ, we use logit distillation on valid next-token positions:

$$\mathcal{L}_{\text{logit-KD}} = \tau^2 \mathbb{E}_t \left[D_{\mathrm{KL}} \big(\mathrm{softmax}(z_t^T/\tau) \parallel \mathrm{softmax}(z_t^S/\tau) \big) \right] . \tag{8}$$

The training objective fixes the CE coefficient to one:

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{CE}} + \alpha \mathcal{L}_{\mathrm{router}} + \beta \mathcal{L}_{\mathrm{residual}} + \gamma \mathcal{L}_{\text{logit-KD}}. \tag{9}$$

Gradients from CE, residual matching, and Logit-KD update only the shared and selected routed experts, while the routing loss updates only the router. Because the selected expert outputs are summed without router-dependent weights, the output losses do not provide gradients to the router logits through the hard top-K selection. This lets teacher activation mass guide *which* partitions are used, while layerwise and prediction losses train the sparse computation produced by those choices. Activation mass is a practical selection proxy. The residual and prediction objectives account for the remaining teacher–student gap.

Algorithm 1 summarizes the frozen teacher forward pass, the student forward pass for CE, residual matching, and Logit-KD, and the second student forward pass with teacher-forced expert selection for router supervision. Further training details are provided in Appendix B.

4 Experiments & results

Our experiments evaluate RRD in four stages. First, we compare its recovery of dense-model accuracy with that of other Dense-to-MoE methods under a limited continual pretraining (CPT) budget. We also count the frozen teacher's forward pass in matched training floating-point operations (FLOPs). Second, we apply the same teacher-free supervised fine-tuning (SFT) procedure to the CPT checkpoints to examine later adaptation. Third, we examine RRD's dependence on MoE weight initialization. Finally, we use quantitative analyses to further assess RRD's effect.

4.1 EXPERIMENTAL SETUP

We use Qwen2.5-7B and Llama-2-7B as dense teachers and sparse student backbones. E/S/A denotes the numbers of total experts, always-active shared experts, and routed experts selected per token. Feed-forward network (FFN) sparsity is the fraction of the original FFN partitions left inactive per token. At 25% sparsity, topologies with shared experts use E8S3A3, while those without shared experts use E8S0A6. At 50% sparsity, they use E8S2A2 and E8S0A4, respectively. Both topology types activate six of eight FFN partitions at 25% sparsity and four at 50%. Across the two backbones, this corresponds to roughly 81–84% and 63–69% of the full student's parameters active per token, respectively. Our main CPT runs use approximately 40M tokens from the C4 corpus. The full RRD runs take 10,240 updates with two sequences of 2,048 tokens per update, for 40M-token positions. We report each method's expert configuration, token budget, and teacher use alongside its results.

After CPT, we report mean zero-shot accuracy across five language-understanding tasks and perplexity on C4 and WikiText-2. We use a separate SFT stage to test the hypothesis that a student closer to its dense teacher after CPT provides a stronger starting point for instruction tuning. Starting from each method's final CPT checkpoint, we apply the same low-rank adaptation (LoRA) procedure to a fixed set of 10,000 examples from Ai2's Tülu-3 SFT mixture (Lambert et al., 2025). We use the same examples in the same order for every Qwen2.5-7B SFT run.

After SFT, we evaluate mean accuracy across five language-understanding tasks with task-specific shot settings, MMLU (Hendrycks et al., 2020) at five shots, and two generation tasks: GSM8K (Cobbe et al., 2021) with eight chain-of-thought examples and MBPP+ (Liu et al., 2023) pass@1. Additional experimental details are provided in Appendix C.

4.2 CONTINUAL PRETRAINING

Table 1: CPT results for Qwen2.5-7B and Llama-2-7B after a 40M-token C4 run. A 25% FFN sparsity level leaves 25% of the original FFN inactive for each token. Higher mean accuracy over five zero-shot tasks and lower C4 and WikiText-2 perplexity indicate better performance. RRD leads CMoE with LoRA by 3.13 percentage points on Qwen2.5-7B and 1.96 points on Llama-2-7B at the same E8S3A3 topology.

Qwen2.5-7B	Sparsity	Topology	PIQA	WinoGrande	ARC-E	ARC-C	HellaSwag	Avg. Acc.	C4 PPL ↓	WT2 PPL ↓
LLaMA-MoE + CE	25%	E8S0A6	73.83	60.54	62.96	39.76	61.88	59.79	17.75	15.77
CMoE + LoRA	25%	E8S3A3	75.03	65.82	69.95	45.82	69.71	65.27	14.63	11.45
ExpertWeaver	25%	E8S3A3	73.50	62.98	59.81	38.74	62.26	59.46	17.97	15.97
DOT-MoE	25%	E8S0A6	74.48	65.43	66.37	44.71	68.27	63.85	16.69	13.85
CMoE + RRD	25%	E8S3A3	**76.61**	**69.61**	**73.86**	**49.32**	**72.62**	**68.40**	**13.62**	**9.45**
Llama-2-7B	Sparsity	Topology	PIQA	WinoGrande	ARC-E	ARC-C	HellaSwag	Avg. Acc.	C4 PPL ↓	WT2 PPL ↓
LLaMA-MoE + CE	25%	E8S0A6	72.42	61.09	57.03	31.91	62.28	56.94	10.65	13.32
CMoE + LoRA	25%	E8S3A3	76.50	64.33	65.61	38.23	71.23	63.18	8.53	**10.30**
ExpertWeaver	25%	E8S3A3	73.99	63.14	60.27	35.49	63.86	59.35	10.05	12.02
DOT-MoE	25%	E8S0A6	74.10	61.17	60.82	35.15	66.66	59.58	9.50	10.97
CMoE + RRD	25%	E8S3A3	**77.48**	**66.61**	**69.32**	**40.27**	**72.01**	**65.14**	**8.26**	10.32

Table 1 reports CPT results at 25% FFN sparsity. With all four loss terms, RRD reaches 68.40% mean zero-shot accuracy and 13.62 C4 perplexity on Qwen2.5-7B. CMoE with LoRA reaches 65.27% mean accuracy, ExpertWeaver reaches 59.46%, and DOT-MoE reaches 63.85%. On Llama-2-7B, RRD reaches 65.14%, compared with 63.18% for CMoE with LoRA and 59.35% for ExpertWeaver. These numbers compare each method's full conversion and training procedure. We examine individual RRD losses in the ablations below.

Table 2 extends the comparison to 50% FFN sparsity. On Qwen2.5-7B, RRD reaches 59.71% mean zero-shot accuracy and 16.90 C4 perplexity, compared with 55.09% and 21.13 for CMoE with LoRA at the same E8S2A2 topology. On Llama-2-7B, RRD reaches 58.12% mean accuracy and 10.10 C4 perplexity, compared with 55.04% and 11.27 for CMoE with LoRA under the same topology.

Table 2: CPT results for Qwen2.5-7B and Llama-2-7B at 50% FFN sparsity after a 40M-token C4 run. In the Llama-2-7B E8S2A2 comparison, RRD reaches 58.12% mean accuracy, 3.08 percentage points above CMoE + LoRA. The accuracy gain persists at this lower active FFN capacity.

Qwen2.5-7B	Sparsity	Topology	PIQA	WinoGrande	ARC-E	ARC-C	HellaSwag	Avg. Acc.	C4 PPL ↓	WT2 PPL ↓
LLaMA-MoE + CE	50%	E8S0A4	62.95	54.78	45.75	26.02	40.58	46.02	77.88	95.82
CMoE + LoRA	50%	E8S2A2	66.65	57.54	59.05	37.46	54.74	55.09	21.13	16.28
ExpertWeaver	50%	E8S2A2	68.55	59.04	52.78	30.80	53.54	52.94	21.60	19.59
DOT-MoE	50%	E8S0A4	69.86	60.62	56.94	35.92	59.88	56.64	20.24	18.25
CMoE + RRD	50%	E8S2A2	**73.61**	**63.38**	**60.94**	**38.65**	**61.96**	**59.71**	**16.90**	**13.05**
Llama-2-7B	Sparsity	Topology	PIQA	WinoGrande	ARC-E	ARC-C	HellaSwag	Avg. Acc.	C4 PPL ↓	WT2 PPL ↓
LLaMA-MoE + CE	50%	E8S0A4	67.30	55.96	47.10	26.96	52.42	49.95	13.22	17.20
CMoE + LoRA	50%	E8S2A2	70.95	61.40	51.81	30.20	60.84	55.04	11.27	14.66
ExpertWeaver	50%	E8S2A2	70.40	58.09	52.78	30.20	57.02	53.70	11.60	14.45
DOT-MoE	50%	E8S0A4	69.31	59.75	51.35	28.75	58.34	53.50	11.81	14.79
CMoE + RRD	50%	E8S2A2	**73.45**	**61.80**	**58.38**	**33.02**	**63.97**	**58.12**	**10.10**	**13.25**

Table 3: Qwen2.5-7B at 50% FFN sparsity under approximately 3,370 PFLOPs of training compute, including the frozen dense teacher's forward pass where used. E8S2A2 includes shared experts and E8S0A4 has no shared experts. Methods train on different numbers of C4 tokens within this compute budget. RRD reaches 59.71% mean accuracy over five zero-shot tasks on 40M tokens, 2.47 percentage points above DOT-MoE on 59.06M tokens. Higher mean accuracy and lower C4 perplexity indicate better performance. RRD retains an accuracy lead after counting the teacher's forward-pass compute. The budget excludes conversion, calibration, evaluation, and method search.

Qwen2.5-7B	Sparsity	Topology	Train FLOPs (PFLOPs)	Train tokens (M)	Update steps	Avg. Accuracy	C4 PPL ↓
ExpertWeaver	50%	E8S2A2	3.37K	89.74M	21,910	54.52	21.04
DOT-MoE	50%	E8S0A4	3.37K	59.06M	14,418	57.24	20.06
CMoE + RRD	50%	E8S2A2	3.37K	41.94M	10,240	**59.71**	**16.90**

4.3 Performance under a matched training budget

RRD uses a frozen teacher during CPT, so we compare methods under approximately the same training compute budget, counting the teacher's forward pass. Table 3 reports Qwen2.5-7B results at 50% FFN sparsity under about 3,370 peta floating-point operations (PFLOPs). RRD reaches 59.71% mean zero-shot accuracy and 16.90 C4 perplexity after 40M C4 input tokens. DOT-MoE reaches 57.24% and 20.06 after 59.06M tokens, while ExpertWeaver reaches 54.52% and 21.04 after 89.74M tokens. RRD leads DOT-MoE, the strongest comparison method in this table, by 2.47 percentage points while training on fewer tokens.

These results show that RRD's accuracy gain persists when we account for the teacher during training. The reported budget covers algorithmic training operations. It excludes conversion, calibration, evaluation, and method search. The later SFT stage uses response-only LoRA, so it incurs no teacher forward cost.

4.4 Supervised fine-tuning

Table 4 compares Qwen2.5-7B SFT at 25% and 50% FFN sparsity. Each sparse run starts from the final CPT checkpoint for its method and sparsity setting. We then train every method with the same teacher-free, response-only LoRA procedure. This design tests whether a stronger CPT starting point also supports better instruction tuning without further distillation. For each fixed CPT checkpoint, we report the mean and sample standard deviation across three SFT seeds. We evaluate the mixed-shot five-task average, MMLU, GSM8K, and MBPP+ after SFT.

At 25% sparsity, CMoE + RRD achieves the highest mean score among sparse models on every reported SFT metric, including 70.48% five-task average accuracy and 59.67% GSM8K accuracy. At 50% sparsity, it again leads on the five-task average, MMLU, GSM8K, and all five individual language-understanding tasks. ExpertWeaver scores higher on MBPP+ at 50% sparsity (18.96%

Table 4: Qwen2.5-7B SFT results at 25% and 50% FFN sparsity. Each sparse model starts from its method-specific CPT checkpoint and uses the same teacher-free, response-only LoRA procedure. Numbers in parentheses after task names indicate the number of few-shot examples. Avg. Acc. averages PIQA, WinoGrande (WG), ARC-E, ARC-C, and HellaSwag (HS). MMLU uses five shots, and GSM8K-CoT uses eight CoT examples. MBPP+ reports pass@1. Values are means and sample standard deviations across three SFT seeds for each fixed checkpoint. Higher is better for all metrics.

Qwen2.5-7B	Sp.	Topology	PIQA (0)	WG (5)	ARC-E (0)	ARC-C (25)	HS (10)	Avg. Accuracy	MMLU (5)	GSM8K CoT (8)	MBPP+ (pass@1)
Dense	0%	-	$79.92_{\pm 0.22}$	$74.48_{\pm 0.37}$	$79.64_{\pm 1.14}$	$63.23_{\pm 0.37}$	$79.43_{\pm 0.13}$	$75.34_{\pm 0.19}$	$73.71_{\pm 0.29}$	$77.84_{\pm 2.31}$	$67.99_{\pm 1.83}$
CMoE + LoRA	25%	E8S3A3	$75.79_{\pm 0.96}$	$67.38_{\pm 1.61}$	$74.73_{\pm 0.38}$	$52.02_{\pm 0.79}$	$71.40_{\pm 0.14}$	$68.26_{\pm 0.43}$	$61.28_{\pm 0.48}$	$54.94_{\pm 2.17}$	$36.16_{\pm 1.91}$
ExpertWeaver	25%	E8S3A3	$72.36_{\pm 0.09}$	$63.48_{\pm 0.32}$	$66.13_{\pm 0.30}$	$44.08_{\pm 0.13}$	$64.35_{\pm 0.06}$	$62.08_{\pm 0.10}$	$55.37_{\pm 0.69}$	$40.49_{\pm 0.93}$	$37.83_{\pm 0.26}$
DOT-MoE	25%	E8S0A6	$74.32_{\pm 0.41}$	$66.59_{\pm 1.05}$	$70.30_{\pm 0.95}$	$51.22_{\pm 0.21}$	$69.54_{\pm 0.05}$	$66.39_{\pm 0.36}$	$59.09_{\pm 0.16}$	$49.03_{\pm 1.92}$	$40.48_{\pm 0.26}$
CMoE + RRD	25%	E8S3A3	**77.49**$_{\pm 0.46}$	**70.24**$_{\pm 0.75}$	**77.27**$_{\pm 0.04}$	**54.04**$_{\pm 0.43}$	**73.35**$_{\pm 0.12}$	**70.48**$_{\pm 0.26}$	**63.77**$_{\pm 0.38}$	**59.67**$_{\pm 2.97}$	**41.71**$_{\pm 1.62}$
Qwen2.5-7B	Sp.	Topology	PIQA (0)	WG (5)	ARC-E (0)	ARC-C (25)	HS (10)	Avg. Accuracy	MMLU (5)	GSM8K CoT (8)	MBPP+ (pass@1)
CMoE + LoRA	50%	E8S2A2	$66.87_{\pm 0.14}$	$58.51_{\pm 1.23}$	$59.15_{\pm 0.89}$	$38.08_{\pm 0.42}$	$56.13_{\pm 0.19}$	$55.75_{\pm 0.28}$	$43.24_{\pm 1.06}$	$27.17_{\pm 1.53}$	$15.26_{\pm 1.70}$
ExpertWeaver	50%	E8S2A2	$68.81_{\pm 0.03}$	$60.30_{\pm 1.01}$	$56.36_{\pm 1.80}$	$36.38_{\pm 0.40}$	$55.06_{\pm 0.18}$	$55.38_{\pm 0.25}$	$47.52_{\pm 0.32}$	$11.62_{\pm 0.97}$	**18.96**$_{\pm 0.55}$
DOT-MoE	50%	E8S0A4	$69.13_{\pm 0.17}$	$60.12_{\pm 0.37}$	$59.43_{\pm 0.68}$	$40.02_{\pm 1.87}$	$61.80_{\pm 0.11}$	$58.10_{\pm 0.55}$	$46.16_{\pm 0.47}$	$17.74_{\pm 1.65}$	$14.02_{\pm 3.30}$
CMoE + RRD	50%	E8S2A2	**73.34**$_{\pm 0.71}$	**64.27**$_{\pm 0.33}$	**66.72**$_{\pm 0.64}$	**41.35**$_{\pm 0.69}$	**62.87**$_{\pm 0.27}$	**61.71**$_{\pm 0.30}$	**49.61**$_{\pm 0.98}$	**32.07**$_{\pm 2.50}$	$15.96_{\pm 2.28}$

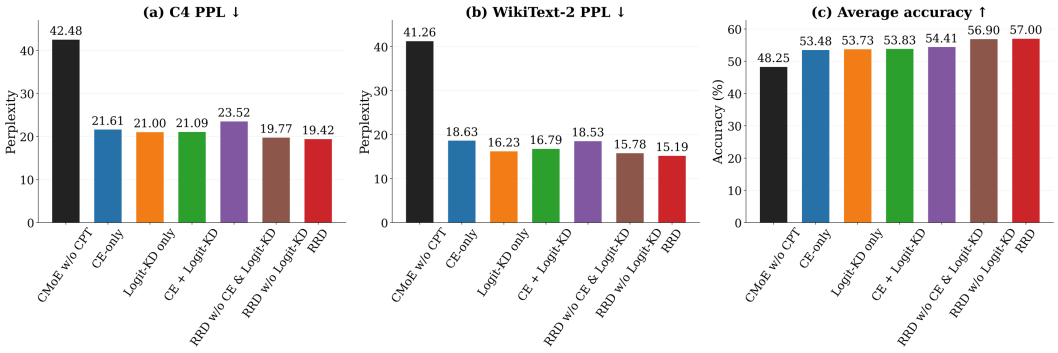


Figure 2: Ablation results for Qwen2.5-7B (E8S2A2). The panels report (a) C4 perplexity, (b) WikiText-2 perplexity, and (c) mean zero-shot accuracy across five benchmarks. All trained variants use the same 4M-token C4 training budget. RRD performs best on all three metrics, achieving 19.42 C4 PPL, 15.19 WikiText-2 PPL, and 57.00% mean accuracy.

versus 15.96%). Together, these results are consistent with our hypothesis that teacher-guided internal alignment during CPT benefits subsequent instruction tuning, even though SFT itself uses no teacher.

4.5 Recovery with a random-split MoE

CMoE uses activation-based expert partitioning with an always-active shared path (Pei et al., 2026), whereas the tested LLaMA-MoE configuration randomly partitions FFN neurons into eight experts without a shared path (Zhu et al., 2024). Table 5 shows that the random split starts from lower accuracy and higher perplexity, consistent with frequently active neurons being omitted by sparse routing. We apply RRD using the CMoE training hyperparameters without retuning. RRD improves recovery over CE in this setting: LLaMA-MoE reaches 57.18% mean accuracy with RRD, versus 46.02% with CE. Although LLaMA-MoE starts at 36.18% mean accuracy versus CMoE's 48.25%, LLaMA-MoE with RRD exceeds CMoE with LoRA in mean accuracy (57.18% versus 55.09%) and lowers both C4 and WikiText-2 perplexity (19.20 versus 21.13 and 15.28 versus 16.28, respectively). These results suggest that teacher-guided internal alignment transfers across the two tested conversion procedures and topologies.

Table 5: Qwen2.5-7B at 50% sparsity: CMoE and random-split LLaMA-MoE before and after CPT.

Qwen2.5-7B	Sp.	Topology	Avg. Acc.	C4 PPL ↓	WT2 PPL ↓
LLaMA-MoE w/o CPT	50%	E8S0A4	36.18	4.65K	8.14K
LLaMA-MoE + CE	50%	E8S0A4	46.02	77.88	95.82
LLaMA-MoE + RRD	50%	E8S0A4	**57.18**	**19.20**	**15.28**
CMoE w/o CPT	50%	E8S2A2	48.25	42.48	41.26
CMoE + LoRA	50%	E8S2A2	55.09	21.13	16.28
CMoE + RRD	50%	E8S2A2	**59.71**	**16.90**	**13.05**

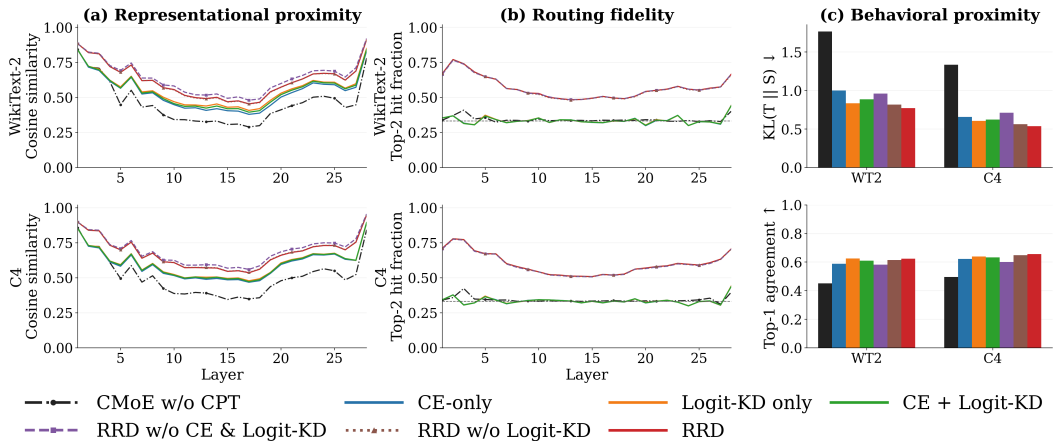


Figure 3: Teacher–student proximity across representation, routing, and behavior. We compare seven Qwen2.5-7B E8S2A2 CMoE variants with the same dense teacher on fixed WikiText-2 and C4 inputs. CMoE w/o CPT is the step-0 conversion, while the other variants are trained on 4M C4 tokens. (a) Layerwise cosine similarity between student MoE and teacher dense MLP outputs. (b) Layerwise top-2 expert hit fraction against a partition-conditioned teacher activation-mass oracle. The dashed line marks the random hit rate of $2/6$. This metric counts matched experts, not exact set matches. (c) Full-vocabulary token-level $\mathrm{KL}(\mathrm{T} \parallel \mathrm{S})$ and teacher–student top-1 agreement. RRD yields the lowest KL on both datasets, whereas RRD w/o CE & Logit-KD has the highest mean MLP cosine.

4.6 Ablation and Teacher–Student Proximity Analyses

We use a separate Qwen2.5-7B E8S2A2 experiment with approximately 4M C4 tokens to examine how structural losses complement CE and Logit-KD. Figure 2 compares the CMoE conversion before CPT with six variants trained on the same token budget. RRD reaches 57.00% mean zero-shot accuracy and C4 and WikiText-2 perplexities of 19.42 and 15.19. Compared with CE + Logit-KD, which reaches 53.83%, 21.09, and 16.79 on the same measures, adding routing and residual distillation together improves accuracy by 3.17 percentage points and reduces both perplexities. Within RRD, Logit-KD raises mean accuracy from 56.90% to 57.00% and lowers C4 and WikiText-2 perplexity from 19.77 to 19.42 and from 15.78 to 15.19.

Figure 3 compares the same variants with their dense teacher on fixed WikiText-2 and C4 inputs. The RRD variants attain top-2 expert hit fractions above the random rate of two out of six across layers, while the CE-only and Logit-KD-only variants stay close to that rate. RRD without CE and Logit-KD has the highest mean cosine similarity between student MoE and teacher MLP outputs on both datasets. RRD has the lowest full-vocabulary $\mathrm{KL}(\mathrm{T} \parallel \mathrm{S})$ between teacher and student next-token distributions on both datasets. The structural-loss-only variant also trails RRD in C4 perplexity (23.52 versus 19.42) and mean accuracy (54.41% versus 57.00%). The highest MLP-output cosine similarity does not coincide with the best perplexity or accuracy. RRD performs best on these predictive measures in this experiment.

5 Conclusion

We address Dense-to-MoE recovery through expert selection and layerwise FFN output alignment, which CE and Logit-KD do not directly supervise. Routing-and-Residual Distillation (RRD) adds these teacher-derived objectives while retaining the converted expert partition and capacity. After about 40M C4 tokens, RRD leads evaluated methods in mean zero-shot accuracy on Qwen2.5-7B and Llama-2-7B at 25% and 50% FFN sparsity. On Qwen2.5-7B at 50% sparsity, it exceeds DOT-MoE by 2.47 percentage points under approximately equal training FLOPs, including teacher forward passes. Ablation study shows that adding routing and residual distillation to CE and Logit-KD further improves recovery. After identical teacher-free SFT, RRD's Qwen2.5-7B checkpoints also lead the compared sparse models in five-task average accuracy, MMLU, and GSM8K at both sparsity levels. These results show that RRD helps recover accuracy lost during conversion, with benefits that persist through instruction tuning.

AI USE STATEMENT

We used generative AI tools for proofreading, checking references, and assisting with the planning and monitoring of computational resources for experiments. The authors verified the final manuscript and references and take responsibility for the paper.

ETHICS STATEMENT

This work uses publicly available pretrained models and datasets and does not collect new data from human participants. The converted models may inherit biases and harmful generation behavior from their source models and data. We do not claim improvements in model safety.

REPRODUCIBILITY STATEMENT

The method and training procedure are described in the main text and appendix. The appendix provides the data, conversion, optimization, and evaluation settings. An anonymous code repository is linked in the paper.

REFERENCES

Udbhav Bamba, Arnav Chavan, Aryamaan Thakur, Steven Teig, and Deepak Gupta. DOT-MoE: Differentiable optimal transport for MoEfication. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=WrXyXq11Yc.

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

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 (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

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*, abs/1803.05457, 2018.

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.

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. StableMoE: Stable routing strategy for mixture of experts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7085–7095, 2022.

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

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.

Shangqian Gao, Ting Hua, Reza Shirkavand, Chi-Heng Lin, Zheng 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.

Guoqiang Gong, Jiaxing Wang, Jin Xu, Deping Xiang, Zicheng Zhang, Leqi Shen, Yifeng Zhang, Junhua Shu, Zhaolong Xing, Zhen Chen, Pengzhang Liu, and Ke Zhang. Beyond logits: Aligning feature dynamics for effective knowledge distillation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23067–23077, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1125. URL https://aclanthology.org/2025.acl-long.1125/.

Hongcan Guo, Haolang Lu, Guoshun Nan, Bolun Chu, Jialin Zhuang, Yuan Yang, Wenhao Che, Xinye Cao, Sicong Leng, Qimei Cui, and Xudong Jiang. Advancing expert specialization for Better MoE. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 48767–48809. Curran Associates, Inc., 2025. doi: 10.52202/085713-1629. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/4598de7d243d528e38eb0c5d8155fb52-Paper-Conference.pdf.

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*, pp. 1389–1400, 2022.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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

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.

Sungyoon Kim, Youngjun Kim, Kihyo Moon, and Minsung Jang. LaDiMo: Layer-wise distillation inspired MoEfier. *arXiv preprint arXiv:2408.04278*, 2024.

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.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=i1uGbfHHpH.

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.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

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.

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. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th*

Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 4777–4789, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.218. URL https://aclanthology.org/2026.acl-long.218/.

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.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets. In *International Conference on Learning Representations (ICLR)*, 2015.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019.

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.

Fangxun Shu, Yue Liao, Le Zhuo, Chenning Xu, Lei Zhang, Guanghao Zhang, Haonan Shi, Long Chen, Tao Zhong, Wanggui He, et al. LLaVA-MoD: Making LLaVA tiny via MoE knowledge distillation. *arXiv preprint arXiv:2408.15881*, 2024.

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*, pp. 3645–3650, 2019.

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

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.

Ziteng Wang, Jun Zhu, and Jianfei Chen. ReMoE: Fully differentiable mixture-of-experts with ReLU routing. In *International Conference on Learning Representations*, volume 2025, pp. 59486–59507, 2025.

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, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

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

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*, pp. 877–890, 2022.

Ziyu Zhao, Tong Zhu, Xin Yu, Zhi Zhang, Tiantian Fan, Jinluan Yang, Kun Kuang, Zhongyu Wei, Fei Wu, and Yu Cheng. ExpertWeaver: Unlocking the inherent MoE in dense LLMs with GLU activation patterns. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=cRs6HUUUay.

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.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. ST-MoE: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

A LIMITATIONS AND FUTURE WORK

Limitations. Our experiments show consistent gains for RRD within the evaluated settings, but the study remains limited in scale and replication. We evaluate two 7B backbones, two FFN sparsity levels, and main recovery runs of approximately 40M C4 tokens. The compared methods use different expert-construction, calibration, and recovery procedures, so the main tables compare complete dense-to-MoE pipelines rather than isolated training objectives. The controlled loss and teacher–student analyses more directly examine the RRD objectives, but use a shorter 4M-token run. Most CPT results rely on one converted checkpoint, while the SFT seeds vary only downstream adaptation from fixed Qwen2.5-7B CPT checkpoints.

Future work. Future work should evaluate independent conversion and CPT seeds, topology- and calibration-matched baselines, longer training, larger backbones, and additional data domains. Repeating the routing and residual ablations at the main training budget would test whether their contributions remain stable. Broader teacher-free SFT experiments would further examine the transfer of CPT recovery to later adaptation.

B RRD TRAINING PROCEDURE

For each minibatch, a frozen teacher forward pass provides intermediate activations, dense FFN outputs, and logits. We derive the layerwise top-K routing targets from the teacher activations. An ordinary student forward pass then supplies the sparse FFN outputs and logits for $\mathcal{L}_{\mathrm{CE}}$, $\mathcal{L}_{\mathrm{residual}}$, and $\mathcal{L}_{\text{logit-KD}}$. Gradients from these losses are accumulated only for the shared and routed expert parameters.

To compute $\mathcal{L}_{\mathrm{router}}$, we run the student again with teacher-forced expert selection. At each layer, the router and experts receive that pass's student pre-FFN state $\tilde{h}_{\ell,t}^{S,\mathrm{TF}}$. The teacher-derived indices $S_{\ell,t}^{\star}$ determine which routed experts execute, and the combined output of the shared module and the teacher-selected routed experts is passed to the next layer without expert gradients. Teacher hidden states and FFN outputs are not substituted into this student pass. We accumulate router gradients only for router parameters and then make one optimizer update.

For the main 40M-token C4 runs, we use $(\alpha, \beta, \gamma) = (1, 2, 1)$ and $\tau = 1$. Adam8bit uses a constant learning rate of 3×10^{-5} for all shared and routed experts. Router parameters use 5×10^{-4} in the early and late layers and 10^{-3} in the middle third of layers.

C EXPERIMENTAL SETUP

Models, conversion, and sparsity. We use the pretrained Qwen2.5-7B (Yang et al., 2024) and Llama-2-7B (Touvron et al., 2023) checkpoints as dense references and as the sources of sparse students. Each converted feed-forward network (FFN) contains eight equal-width expert partitions. E/S/A denotes the numbers of total experts, shared experts active for every token, and routed experts selected per token, with $A = K$. At 25% FFN sparsity, E8S3A3 has three shared and three selected routed experts, while E8S0A6 selects six routed experts without a shared path. At 50% sparsity, E8S2A2 and E8S0A4 activate four of eight partitions per token. Sparsity refers to inactive FFN partitions, rather than inactive parameters across the whole model. Appendix Table 6 reports the full-model counts.

The main Routing-and-Residual Distillation (RRD) experiments start from a CMoE partition (Pei et al., 2026). CMoE uses activation statistics to assign frequently active neurons to shared experts, groups the remaining neurons into routed experts, and initializes its router from representative neurons. We also apply RRD to an independent random LLaMA-MoE partition (Zhu et al., 2024) with no shared experts to test another initialization. CMoE and ExpertWeaver use 64 fixed sequences of 2,048 tokens from C4 for conversion, depending on the final run. The random LLaMA-MoE split uses seed 0 without activation calibration. DOT-MoE (Bamba et al., 2026) learns its assignment within the training budget. We keep separate calibration outside the training token budget and record its corpus and checkpoint in each run manifest. The main comparisons therefore evaluate complete conversion and training procedures; they do not assume identical calibration across methods.

Table 6: Active and total inference parameters (billions) for Qwen2.5-7B and Llama-2-7B at 25% and 50% feed-forward network sparsity. E/S/A denotes the numbers of total, shared, and selected routed experts.

Qwen2.5-7B	Sparsity	Topology	Active parameters (B)	Total parameters (B)	Active / Total
LLaMA-MoE + CE	25%	E8S0A6	6.19	7.62	81.27%
CMoE + LoRA	25%	E8S3A3	6.22	7.65	81.25%
ExpertWeaver	25%	E8S3A3	6.19	7.62	81.28%
DOT-MoE	25%	E8S0A6	6.19	7.62	81.28%
LLaMA-MoE + RRD	25%	E8S0A6	6.19	7.62	81.27%
CMoE + RRD	25%	E8S3A3	6.29	7.72	81.53%
Qwen2.5-7B	Sparsity	Topology	Active parameters (B)	Total parameters (B)	Active / Total
LLaMA-MoE + CE	50%	E8S0A4	4.77	7.62	62.56%
CMoE + LoRA	50%	E8S2A2	4.78	7.65	62.52%
ExpertWeaver	50%	E8S2A2	4.76	7.62	62.56%
DOT-MoE	50%	E8S0A4	4.76	7.62	62.56%
LLaMA-MoE + RRD	50%	E8S0A4	4.77	7.62	62.56%
CMoE + RRD	50%	E8S2A2	4.87	7.72	63.06%
Llama-2-7B	Sparsity	Topology	Active parameters (B)	Total parameters (B)	Active / Total
LLaMA-MoE + CE	25%	E8S0A6	5.66	6.74	83.93%
CMoE + LoRA	25%	E8S3A3	5.68	6.78	83.91%
ExpertWeaver	25%	E8S3A3	5.66	6.74	83.94%
DOT-MoE	25%	E8S0A6	5.66	6.74	83.94%
LLaMA-MoE + RRD	25%	E8S0A6	5.66	6.74	83.93%
CMoE + RRD	25%	E8S3A3	5.79	6.87	84.26%
Llama-2-7B	Sparsity	Topology	Active parameters (B)	Total parameters (B)	Active / Total
LLaMA-MoE + CE	50%	E8S0A4	4.58	6.74	67.89%
CMoE + LoRA	50%	E8S2A2	4.60	6.78	67.83%
ExpertWeaver	50%	E8S2A2	4.57	6.74	67.89%
DOT-MoE	50%	E8S0A4	4.58	6.74	67.89%
LLaMA-MoE + RRD	50%	E8S0A4	4.58	6.74	67.89%
CMoE + RRD	50%	E8S2A2	4.71	6.87	68.52%

Continual pretraining data and budget. Within each backbone, our main continual pretraining (CPT) methods use a fixed, ordered stream from C4. A 40M-token run takes 10,240 optimizer updates with two sequences of 2,048 input tokens per update and no gradient accumulation, totaling 41,943,040 input token positions. A separate Qwen2.5-7B E8S2A2 ablation uses 1,024 updates and approximately 4M C4 input token positions. The conversion calibration samples sit outside these CPT budgets.

Compared methods and CPT optimization. The dense reference retains the pretrained model without sparse conversion or CPT. LLaMA-MoE uses the independent random FFN partition and trains its routed experts and router with cross-entropy (CE) (Zhu et al., 2024). For CMoE + LoRA, we first perform CMoE conversion and router initialization, then adapt the converted MoE with LoRA on attention and expert projections while training its router and scaling parameters (Pei et al., 2026). As in CMoE, this post-conversion stage uses LoRA-based recovery rather than full fine-tuning. ExpertWeaver forms shared and routed experts from activation patterns before CPT (Zhao et al., 2026). DOT-MoE learns neuron assignment and routing during alignment, exports the sparse MoE, and continues fitting it (Bamba et al., 2026).

CMoE + LoRA uses rank 8, scaling factor 32, dropout 0.1, a 5.95×10^{-5} LoRA learning rate, and a separate 10^{-3} rate for router and scaling parameters. The final Qwen2.5-7B E8S2A2 LoRA run uses full-precision Adam and gradient clipping at 1. ExpertWeaver uses a 5×10^{-5} peak learning rate with cosine decay and a load-balancing term. DOT-MoE uses alignment for 25% of its C4 token budget and sparse fitting for the remaining 75%. We adapt these methods to the eight-expert configurations and shared C4 budget; their data, training scale, and some optimization settings therefore differ from the original papers. These comparisons evaluate the implemented conversion-and-training procedures, not reproduced scores from the original papers.

Supervised fine-tuning. For supervised fine-tuning (SFT), we use Qwen2.5-7B at 25% and 50% FFN sparsity and start each sparse run from the exact final CPT checkpoint for its method and sparsity setting. We train a fresh LoRA adapter for one epoch on the same fixed 10,000-example subset of the Tülu-3 SFT mixture (Lambert et al., 2025), with Qwen2.5-7B tokenization and a 2,048-token limit. For every method, including DOT-MoE, the SFT objective consists solely of response-only cross-entropy on assistant tokens. The common LoRA recipe uses rank 8, scaling factor 32, dropout 0.1, and learning rate 5.95×10^{-5}. SFT uses no dense teacher, routing distillation, residual matching, or Logit-KD. We vary the SFT seed while holding the source CPT checkpoint fixed. The three seeds measure SFT variability from that checkpoint, not independent conversion or CPT runs. We report the mean and sample standard deviation across the three SFT seeds.

Evaluation and compute accounting. After CPT, we measure perplexity on 256 C4 test windows and model-specific WikiText-2 validation-and-test windows: 265 for Qwen2.5-7B and 292 for Llama-2-7B. We report zero-shot accuracy on PIQA (Bisk et al., 2020), ARC-Easy (ARC-E), ARC-Challenge (ARC-C) (Clark et al., 2018), HellaSwag (Zellers et al., 2019), and WinoGrande (Sakaguchi et al., 2019). Their arithmetic mean is the CPT five-task score. After SFT, we use PIQA and ARC-Easy at zero shots, WinoGrande at five shots, ARC-Challenge at 25 shots, and HellaSwag at ten shots. We also evaluate Massive Multitask Language Understanding (MMLU) at five shots, GSM8K with eight chain-of-thought demonstrations, and MBPP+ pass@1. We keep the CPT zero-shot average separate from the SFT mixed-shot average.

For the approximately matched training-compute comparison, we count each method's training operations and include the frozen teacher's forward pass when a method uses one. The budget excludes conversion, calibration, evaluation, and method search. Methods can therefore consume different numbers of C4 tokens under the same approximate floating-point operation budget. We report token counts and teacher use alongside the compute comparison.

Computational resources. Each CPT run used one NVIDIA RTX PRO 6000 GPU, and each SFT run used one NVIDIA RTX A6000 GPU.