
DOT-MoE: Differentiable Optimal Transport for MoEfication

Udbhav Bamba^{*1} Arnav Chavan^{*1} Aryamaan Thakur^{*1} Steve Teig¹ Deepak Gupta¹

Abstract

The scaling of Large Language Models (LLMs) has driven significant performance gains but created substantial challenges in inference efficiency. While Mixture of Experts (MoEs) architectures address this by decoupling model size from inference cost, training MoEs from scratch is often unstable and compute intensive. Conversion of pre-trained dense models into sparse MoEs has emerged as an alternative solution; however, existing methods typically rely on heuristic neuron clustering or random splitting to partition the Feed-Forward Network (FFN) into experts. In this work, we propose DOT-MoE, a novel framework that formulates the decomposition of dense layers as a Differentiable Optimal Transport (DOT) problem. Instead of static heuristics, we model neuron assignment as a balanced transport problem, utilizing differentiable Sinkhorn-Knopp iterations to enforce strict expert capacity constraints. Furthermore, we utilize Straight-Through Estimators (STE) to jointly learn the discrete neuron-to-expert assignment and the token-to-expert routing policy end-to-end. Extensive experiments across multiple architectures and benchmarks demonstrate that DOT-MoE significantly outperforms structured pruning, heuristic clustering, and random-split baselines, retaining 90% of the original dense model’s performance while reducing active parameters by 50%.

1. Introduction

The rapid scaling of Large Language Models (LLMs) has led to remarkable capabilities in natural language understanding and generation. However, this performance comes at a prohibitive computational cost. As model dimensions grow, the dense activation patterns of standard Transformers

(Vaswani et al., 2017), where every parameter is active for every input token, result in unsustainable inference latency and resource consumption. To address this efficiency bottleneck, Mixture-of-Experts (MoE) (Shazeer et al., 2017; Lepikhin et al., 2020; Jiang et al., 2024; Fedus et al., 2022) architectures have emerged as a promising solution. By routing tokens to a small subset of *expert* sub-networks, MoEs decouple model size from inference cost. For example, the recent Qwen3-30B-A3B (Yang et al., 2025) MoE architecture comprises a total of 30.5B parameters in the network, however, only 3.3B parameters are activated per token during inference.

Despite their inference efficiency, training MoE models from scratch is notoriously data-hungry and unstable, often requiring complex load-balancing auxiliaries (Zoph et al., 2022; Fedus et al., 2022). Consequently, a new paradigm has gained traction: MoEfication (Zhang et al., 2022), or the conversion of pre-trained dense models into sparse MoEs. This approach leverages the high-quality representations of existing dense checkpoints, transforming the Feed-Forward Network (FFN) in each block into sparse experts to reduce inference FLOPs without the cost of pre-training from scratch. Conceptually, this process can be viewed as a form of dynamic structural pruning. Unlike static pruning methods (Frantar & Alistarh, 2023; Sun et al.; Ashkboos et al., 2024; Gao et al., 2024b), which permanently remove parameters and often degrade performance by erasing *long-tail* knowledge essential for LLMs (Lele et al., 2025), MoEfication retains the full parameter set but activates it selectively. By dynamically pruning the network structure conditioned on the input, it maintains the high capacity of the dense model while achieving the efficiency of sparse execution.

The core challenge in this conversion is neuron assignment: determining how to partition the thousands of intermediate neurons in a dense FFN into discrete, independent, and functionally coherent experts. Existing approaches to dense-to-MoE conversion (Zhu et al., 2024; Qu et al., 2024; Pei et al., 2025) rely largely on heuristic strategies for neuron assignment. While effective, these methods often treat neuron assignment and router training as separate processes. They lack a unified, differentiable framework that guarantees balanced expert capacity while simultaneously optimizing for the semantic routing of tokens.

^{*}Equal contribution ¹Amazon. Correspondence to: Udbhav Bamba <udbhb@amazon.com>.

In this work, we propose DOT-MoE, a novel framework that formulates the decomposition of dense layers into experts as a Differentiable Optimal Transport (DOT) problem. We enable end-to-end learning of both the expert decomposition and the routing mechanism by using differentiable Sinkhorn-Knopp iterations (Knight, 2008; Sinkhorn & Knopp, 1967). Unlike existing methods (Pei et al., 2025; Zhu et al., 2024; Qu et al., 2024) which freeze the assignment and then train the router, DOT-MoE allows the router and the expert assignment to co-adapt. Instead of relying on static heuristics, we view neuron assignment as a balanced transport problem where neurons must be *transported* to experts under strict capacity constraints. This is achieved with Straight-Through Estimators (STEs) (Bengio et al., 2013) that allow gradients to flow through the discrete assignment decisions. This ensures that experts are not just random collections of neurons, but balanced functional units optimized specifically for the routing policy.

Our main contributions are:

1. We introduce an optimal transport framework for dense-to-MoE conversion, formulating neuron assignment as a balanced transport problem with differentiable Sinkhorn iterations.
2. We develop a dual-level assignment mechanism that jointly optimizes neuron-to-expert decomposition and token-to-expert routing via complementary straight-through estimators, enabling the router and expert structure to co-adapt unlike prior methods that treat these as separate stages.
3. We demonstrate through extensive experiments across three model families (LLaMA-2, LLaMA-3, Qwen2.5) and six benchmarks that DOT-MoE outperforms both structured pruning and existing MoEfication methods, retaining 90% of the original dense model’s performance at 50% parametric count.

2. Background and Motivation

2.1. Preliminaries: FFN Layers in Transformers

The FFN constitutes the majority of parameters in transformer-based language models, typically accounting for approximately two-thirds of total model parameters. It processes hidden states through a two-stage projection:

$$\mathbf{H} = \sigma(\mathbf{x}\mathbf{W}_{\text{gate}}) \odot (\mathbf{x}\mathbf{W}_{\text{up}}) \quad (1)$$

$$\text{FFN}(\mathbf{x}) = \mathbf{H}\mathbf{W}_{\text{down}} \quad (2)$$

where $\mathbf{x} \in \mathbb{R}^d$ is the input hidden state, $\mathbf{W}_{\text{gate}}, \mathbf{W}_{\text{up}} \in \mathbb{R}^{d \times d_{\text{in}}}$ and $\mathbf{W}_{\text{down}} \in \mathbb{R}^{d_{\text{in}} \times d}$ are weight matrices, $\sigma(\cdot)$ is an activation function (e.g., SiLU), and \odot denotes element-wise multiplication.

2.2. Existing Approaches for Efficient Inference

Reducing the computational cost of large language models while preserving their capabilities remains a central challenge. Existing approaches fall into two categories: *structured pruning*, which permanently removes model components, and *dense-to-MoE conversion*, which converts dense layers into sparse mixtures of experts.

2.2.1. STRUCTURED PRUNING

Structured pruning improves efficiency by permanently removing structured components such as neurons, channels, attention heads, or entire layers from dense LLMs (Wang et al., 2020). Representative methods such as ShortGPT (Men et al., 2025) and SliceGPT (Ashkboos et al., 2024) exploit structural redundancies via layer- or subspace-level importance criteria, achieving hardware-friendly sparsity and predictable inference speedups. More recent approaches, notably DISP-LLM (Gao et al., 2024b), substantially improve the accuracy-efficiency trade-off by enabling flexible, dimension-wise structural pruning and consistently outperform prior structured pruning baselines. However, structured pruning *irreversibly removes model capacity*, which often leads to sharp performance degradation, particularly at high compression ratios.

2.2.2. DENSE-TO-MoE CONVERSION

MoEfication (Zhang et al., 2022) introduced the paradigm of converting dense FFN layers into sparse MoE layers, preserving total model capacity while reducing per-token computation. This is achieved by partitioning the d_{ffn} intermediate neurons into E disjoint expert groups, each containing $s = d_{\text{ffn}}/E$ neurons. A learned router $\mathbf{W}_r \in \mathbb{R}^{E \times d_{\text{model}}}$ selects the top- k experts per token:

$$\mathcal{I} = \text{top-}k(\text{softmax}(\mathbf{x}\mathbf{W}_r^T), k) \quad (3)$$

This sparsification reduces computational cost from $\mathcal{O}(d_{\text{ffn}})$ to $\mathcal{O}(k \cdot d_{\text{ffn}}/E)$ per token. Existing dense-to-MoE methods differ primarily in how they assign neurons to experts:

(i) *Random Assignment*. LLaMA-MoE (Zhu et al., 2024) randomly partitions neurons into experts and relies on extensive continued pre-training to recover performance. While simple, this approach requires substantial computational resources and provides no principled basis for expert specialization.

(ii) *Weight-based Clustering*. LTE and MoEfication (Zhang et al., 2022; Zheng et al., 2024) clusters neurons based on the similarity of their projection weights \mathbf{W}_{gate} and \mathbf{W}_{up} , assuming that neurons with similar input-side weights respond to similar input patterns.

(iii) *Activation-based Clustering*. LLaMA-MoE-v2 (Qu et al., 2024) assigns neurons to experts based on impor-

tance estimates derived from activations and gradients. CMoE (Pei et al., 2025) clusters intermediate FFN activations \mathbf{H} with balanced k -means, grouping neurons by empirical co-activation patterns.

2.3. Limitation: Optimizing Proxies Instead of Outputs

Existing approaches share a fundamental limitation: *they optimize proxies for intermediate representations while neglecting the actual output*. Consider Equation 2: the output of the FFN depends on the interaction between intermediate activations \mathbf{H} and down-projection weights \mathbf{W}_{down} . Structured pruning methods optimize layer-wise importance scores that ignore this interaction. Dense-to-MoE methods cluster based on input weights, intermediate activations, or co-activation patterns, all of which are proxies that fail to capture how each neuron ultimately contributes to the output.

To empirically validate this limitation, we conducted a controlled single-layer reconstruction analysis on LLaMA-2 and LLaMA-3 (see Appendix A). By isolating the expert construction strategy, we observe that methods relying on input-side statistics (Zhu et al., 2024) or intermediate activations (Qu et al., 2024; Pei et al., 2025) incur mean squared errors ranging from $2\times$ to over $41\times$ higher than our proposed approach. These results confirm that preserving FFN fidelity requires explicitly modeling the neuron’s contribution to the output, rather than optimizing based on proxies.

3. Method

3.1. Problem Formulation

Our goal is to convert a dense FFN layer into a sparse MoE layer with E experts, each containing s intermediate neurons such that $s \cdot E = d_{\text{ffn}}$, while activating only $k < E$ experts per token. Crucially, we aim to achieve this conversion without full model fine-tuning, enabling efficient post-hoc sparsification of pretrained dense models.

The central challenge is neuron assignment: determining which of the d_{ffn} neurons should be grouped into each expert. This problem is combinatorially intractable, with $\frac{d_{\text{ffn}}!}{(s!)^E}$ possible balanced partitions. Furthermore, assignment and routing are tightly coupled, i.e., changing which neurons belong to an expert changes what inputs should be routed to it, and vice versa. This interdependence precludes simple two-stage approaches that fix assignments before training routers.

We propose to *jointly learn* the neuron-to-expert assignment and the token-to-expert routing by formulating assignment as an optimal transport problem. The key insight is that neuron assignment can be viewed as *transporting mass* from neurons to experts: each neuron carries unit mass

to be delivered to exactly one expert, while each expert must receive exactly s units. The cost of each assignment is determined by how well the resulting MoE reconstructs the dense FFN output. This perspective maps directly to optimal transport (OT), which finds minimum-cost mass redistribution under marginal constraints.

This formulation requires three components:

1. A *differentiable* relaxation of the discrete assignment that permits gradient-based optimization.
2. *Hard capacity constraints* ensuring each expert receives exactly s neurons and each neuron is assigned to exactly one expert.
3. An *output-aware objective* that directly measures deviation from the dense FFN output, not intermediate representations.

3.2. Neuron-to-Expert Assignment via Optimal Transport

We first formalize the assignment problem using the language of optimal transport.

Definition 3.1 (Optimal Transport Problem). Given source distribution $\mathbf{r} \in \mathbb{R}_+^m$ and target distribution $\mathbf{c} \in \mathbb{R}_+^n$ with $\sum_i r_i = \sum_j c_j$, and a cost matrix $\mathbf{C} \in \mathbb{R}^{m \times n}$, the optimal transport problem seeks a transport plan \mathbf{M}^* that minimizes total transportation cost:

$$\mathbf{M}^* = \underset{\mathbf{M} \in \mathcal{U}(\mathbf{r}, \mathbf{c})}{\operatorname{argmin}} \langle \mathbf{C}, \mathbf{M} \rangle \quad (4)$$

where $\mathcal{U}(\mathbf{r}, \mathbf{c}) = \{\mathbf{M} \geq 0 : \mathbf{M}\mathbf{1}_n = \mathbf{r}, \mathbf{M}^\top \mathbf{1}_m = \mathbf{c}\}$ is the set of valid transport plans (the *transportation polytope*), and $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product.

For neuron assignment, we set $m = d_{\text{ffn}}$ (neurons) and $n = E$ (experts), with marginals $\mathbf{r} = \mathbf{1}_{d_{\text{ffn}}}$ (each neuron assigned exactly once) and $\mathbf{c} = s \cdot \mathbf{1}_E$ (each expert receives exactly s neurons). Rather than specifying a fixed cost matrix, we introduce a *learnable affinity matrix* $\mathbf{A} \in \mathbb{R}^{d_{\text{ffn}} \times E}$ where $A_{i,e}$ represents the affinity of assigning neuron i to expert e . Setting the cost as $\mathbf{C} = -\mathbf{A}$, our objective becomes:

$$\mathbf{M}^* = \underset{\mathbf{M} \in \mathcal{U}(\mathbf{r}, \mathbf{c})}{\operatorname{argmin}} \langle -\mathbf{A}, \mathbf{M} \rangle = \underset{\mathbf{M} \in \mathcal{U}(\mathbf{r}, \mathbf{c})}{\operatorname{argmax}} \langle \mathbf{A}, \mathbf{M} \rangle \quad (5)$$

This seeks the assignment that maximizes total affinity while satisfying the balance constraints.

Limitation of Standard OT. While Eq. 5 captures our desired solution, \mathbf{M}^* lies at a vertex of the transportation polytope which is a $\{0, 1\}$ -matrix where each row contains exactly one entry equal to 1 and each column sums to s . This discrete structure presents two challenges: (1) the argmin over a polytope is non-differentiable, blocking gradient flow

to the affinity matrix \mathbf{A} , and (2) solving the linear program exactly at each training step is computationally prohibitive.

Entropic Regularization. To obtain a differentiable solution, we add entropic regularization to the OT objective (Curi, 2013):

$$\mathbf{M}_\tau^* = \operatorname{argmin}_{\mathbf{M} \in \mathcal{U}(\mathbf{r}, \mathbf{c})} \langle -\mathbf{A}, \mathbf{M} \rangle - \tau H(\mathbf{M}) \quad (6)$$

where $H(\mathbf{M}) = -\sum_{i,e} M_{i,e} (\log M_{i,e} - 1)$ is the entropy of the transport plan and $\tau > 0$ is a temperature parameter. The entropy term $-\tau H(\mathbf{M})$ strictly convexifies the objective, yielding a unique solution in the *interior* of the polytope rather than at a vertex. As $\tau \rightarrow 0$, the solution approaches the unregularized optimum; as $\tau \rightarrow \infty$, it approaches the uniform plan.

The key advantage of entropic regularization is that the solution admits a closed-form factorization:

$$M_{i,e}^* = u_i \cdot \exp(A_{i,e}/\tau) \cdot v_e \quad (7)$$

where scaling vectors $\mathbf{u} \in \mathbb{R}_+^{d_{\text{in}}}$ and $\mathbf{v} \in \mathbb{R}_+^E$ can be found via the Sinkhorn-Knopp algorithm (Sinkhorn & Knopp, 1967; Knight, 2008), which performs alternating row and column normalizations that converge linearly to the unique solution satisfying the marginal constraints. We denote the resulting soft assignment as $\mathbf{M}_{\text{soft}} \in [0, 1]^{d_{\text{in}} \times E}$, computed via log-domain Sinkhorn iterations for numerical stability (see Appendix B.2.1).

3.3. Token-to-Expert Routing

In addition to assigning neurons to experts, we must learn which experts to activate for each input token. We parameterize the router as a linear projection followed by top- k selection.

For input tokens $\mathbf{X} \in \mathbb{R}^{n \times d}$, the router computes:

$$\mathbf{L} = \mathbf{X} \mathbf{W}_r^\top \in \mathbb{R}^{n \times E} \quad (8)$$

$$\mathbf{P} = \operatorname{softmax}(\mathbf{L}) \in \mathbb{R}^{n \times E} \quad (9)$$

$$\mathcal{I}_i = \operatorname{top-k}(\mathbf{P}_i, k) \quad \forall i \in \{1, \dots, n\} \quad (10)$$

where $\mathbf{W}_r \in \mathbb{R}^{E \times d}$ are learnable router weights and \mathcal{I}_i contains the indices of the k experts selected for token i .

3.4. Differentiable Assignment and Routing

With the affinity matrix \mathbf{A} governing neuron-to-expert assignment and router weights \mathbf{W}_r governing token-to-expert routing, we now describe how to jointly optimize both. Two challenges must be addressed: (1) converting the soft assignment \mathbf{M}_{soft} to discrete expert clusters, and (2) enabling gradient flow through both the discrete assignment and the top- k routing selection.

Hard Assignment via Greedy Rounding. The soft assignment \mathbf{M}_{soft} provides fractional neuron-to-expert allocations, but deployment requires discrete assignments. We convert \mathbf{M}_{soft} to a binary matrix $\mathbf{M} \in \{0, 1\}^{d_{\text{in}} \times E}$ via greedy selection: sort all entries of \mathbf{M}_{soft} in descending order, then iteratively assign neuron i to expert e if neuron i is unassigned and expert e has capacity remaining. This yields disjoint expert clusters $\mathcal{C}_1, \dots, \mathcal{C}_E$ with $|\mathcal{C}_e| = s$. A potential concern is mismatch between soft and hard assignments, e.g., when a neuron’s preferred expert has reached capacity. However, Sinkhorn already accounts for capacity constraints globally, redistributing probability mass when experts are over-demanded.

Gradient Estimation via STE. Both the greedy rounding for assignment and the top- k selection for routing are non-differentiable. We employ straight-through estimators (STE) (Bengio et al., 2013) that use hard decisions in the forward pass while routing gradients through the soft counterparts in the backward pass:

$$\mathbf{M}_{\text{STE}} = \mathbf{M} + (\mathbf{M}_{\text{soft}} - \operatorname{sg}(\mathbf{M}_{\text{soft}})) \quad (11)$$

$$\mathbf{R}_{\text{STE}} = \mathbf{R} + (\mathbf{P} - \operatorname{sg}(\mathbf{P})) \quad (12)$$

where $\operatorname{sg}(\cdot)$ denotes the stop-gradient operator, $\mathbf{M} \in \{0, 1\}^{d_{\text{in}} \times E}$ is the hard neuron assignment from greedy rounding, and $\mathbf{R} \in \{0, 1\}^{n \times E}$ is the binary routing mask with $R_{i,e} = 1$ iff expert e is selected for token i . This allows end-to-end optimization: gradients from the reconstruction loss flow through \mathbf{M}_{soft} to update \mathbf{A} , and through \mathbf{P} to update \mathbf{W}_r .

3.5. Alignment Phase

During training, we simulate sparse MoE computation by masking the intermediate activations: $\hat{\mathbf{Y}} = (\mathbf{H} \odot (\mathbf{R} \mathbf{M}^\top)) \mathbf{W}_{\text{down}}$, where only the $k \cdot s$ neurons belonging to the selected experts contribute to each token’s output (see Appendix E). We jointly optimize the assignment logits \mathbf{A} and router weights \mathbf{W}_r across the network using a combination of losses aimed at preserving the residual stream: KL divergence between the pretrained dense teacher and MoE student output distributions, cross-entropy loss on the language modeling objective, and auxiliary losses for MoE training stability. Specifically, we employ router z-loss (Zoph et al., 2022) to penalize large router logits and prevent instability, and load balancing loss (Shazeer et al., 2017) to encourage uniform expert utilization and prevent expert collapse. Full details are provided in Appendix D.

Once training converges, we extract the final assignment \mathbf{M} and convert the model into a standard MoE architecture with E distinct expert FFNs, enabling efficient sparse inference. The same balanced-transport formulation extends directly to multi-head attention by grouping heads into experts; we defer the full formulation and results to Appendix G.

Table 1. Comparison of Perplexity on WikiText-2 and HellaSwag at 50% parametric budget. DOT-MoE outperforms existing structured and semi-structured methods on LLaMA-2 7B.

Method	LLaMA-2 7B	
	WikiText [PPL ↓]	HellaS. [acc-n ↑]
<i>Structured Pruning (50%)</i>		
LLM-Pruner	31.05	–
LLM Surgeon	15.38	40.3
ShortGPT	268.11	43.7
SLEB	103.38	–
K-OBDD	46.64	36.8
SliceGPT	24.82	33.0
ModeGPT	11.88	–
DISP-LLM	9.84	46.3
<i>Semi-Structured Pruning (2:4)</i>		
SparseGPT	10.17	43.3
Wanda	11.02	40.9
Pruner-Zero	10.52	54.7
DOT-MoE	7.99	53.9

4. Experiments

4.1. Experimental Setup

Models and Evaluation. We evaluate on three publicly available dense checkpoints: LLaMA-2-7B (Touvron et al., 2023), LLaMA-3-8B (Grattafiori et al., 2024), and Qwen2.5-7B (Team, 2024). All methods are evaluated using lm-evaluation-harness (Gao et al., 2024a) with benchmark’s default prompts and standard few-shot settings: ARC-Challenge (25-shot) (Clark et al., 2018), Winogrande (5-shot) (Sakaguchi et al., 2021), HellaSwag (10-shot) (Zellers et al., 2019), PIQA (0-shot) (Bisk et al., 2020), SciQ (0-shot) (Welbl et al., 2017), and BoolQ (32-shot) (Clark et al., 2019).

Implementation. DOT-MoE is implemented using PyTorch (Paszke et al., 2019) and Hugging Face Transformers (Wolf et al., 2020). We **freeze** the dense model weights during the alignment phase and only train the assignment logits \mathbf{A} and router weights \mathbf{W}_r . Unless stated otherwise, each expert contains $s = 128$ intermediate neurons, yielding $E = 148, 112$, and 86 experts per layer for Qwen2.5-7B, LLaMA-3-8B, and LLaMA-2-7B, respectively. We use top- k routing with a fixed fraction of active experts per token. For our main experiments, we target 25% FFN sparsity which translates to $k = 37, 28$ and 22 active experts for Qwen2.5-7B, LLaMA-3-8B, and LLaMA-2-7B respectively. We use AdamW with cosine learning rate decay and linear warmup. All experiments are conducted on 8xH100 GPUs. Full hyperparameters are provided in Appendix B.

Training Data. We use Dolmino-mix (OLMo et al., 2025)

for the alignment phase as well as for continuous fine-tuning. Alignment is done for 3500 steps which takes < 3 hours on 8xH100 GPUs for the LLaMA3-8B; a detailed profiling of Sinkhorn and STE overhead is provided in Appendix H. We exclusively train the assignment logits \mathbf{A} and router weights \mathbf{W}_r in the alignment phase, completely freezing the dense model weights. In order to be consistent with existing works, we further use 1.2B tokens for continuous fine-tuning of the aligned model. We sample the same data from the larger Dolmino-mix for training DOT-MoE as well as the baselines.

Baselines. We compare DOT-MoE against a broad set of structured pruning, semi-structured pruning and dense-to-MoE conversion methods. Structured pruning baselines include LLM-Pruner (Ma et al., 2023), LLM Surgeon (Ouderaa et al., 2023), ShortGPT (Men et al., 2025), SLEB (Song et al., 2024), K-OBDD, SliceGPT (Ashkboos et al., 2024), ModeGPT (Lin et al., 2024) and DISP-LLM (Gao et al., 2024b). Semi-structured pruning baselines include Wanda (Sun et al.), SparseGPT (Frantar & Alistarh, 2023) and Pruner-Zero (Dong et al., 2024). Dense-to-MoE baselines include CMoE (Pei et al., 2025), LLaMA-MoE (Zhu et al., 2024) and LLaMA-MoE-v2 (Qu et al., 2024). All baselines are evaluated under comparable sparsity. We justify baseline selection and provide additional comparisons in Appendix I.

4.2. Main Results

Comparison with Pruning Methods. Table 1 compares DOT-MoE against structured and semi-structured pruning methods on LLaMA-2 7B at 50% parametric budget. DOT-MoE achieves the lowest perplexity (7.99) among all existing methods, outperforming the state-of-the-art DISP-LLM (9.84) by a substantial margin. DOT-MoE is also competitive with semi-structured pruning methods which have a greater degree of freedom to achieve any target sparsity.

This advantage extends to downstream tasks. As shown in Table 3, DOT-MoE demonstrates superior knowledge retention compared to pruning baselines across all three model families. On Qwen2.5-7B, DOT-MoE outperforms DISP-LLM (72.3% vs 66.7% average accuracy), confirming that dense-to-MoE conversion is more effective than pruning, as it preserves total model capacity while activating only a subset of parameters per token.

Comparison with Dense-to-MoE Methods. We compare DOT-MoE with existing dense-to-MoE conversion methods in Table 3. A key distinction is that methods like LLaMA-MoE and CMoE first permanently assign neurons to experts, then train a randomly initialized router on this fixed partition. This requires extensive fine-tuning to recover performance. In contrast, DOT-MoE jointly learns neuron assignment and routing during the alignment phase, allowing them to co-

Table 2. Fine-tuning performance comparison on common-sense reasoning benchmarks. #FT Tokens denotes the fine-tuning data budget after conversion. Dense rows report the original pretraining budget (marked with *) for reference; no fine-tuning is applied to dense models. DOT-MoE models recover performance with fewer tokens and bridge the gap to dense counterparts.

Method	Active Params	#FT Tokens	BoolQ [32, acc]	SciQ [0, acc]	PIQA [0, acc]	WinoG. [5, acc]	ARC-C [25, acc-n]	HellaS. [10, acc-n]	Avg.
<i>LLaMA-2 7B</i>									
Dense	6.74B	2T*	82.0	94.0	78.1	74.3	52.5	78.9	76.6
LLaMA-MoE (E8A2) [†]	3.49B	1.2B	37.8	20.0	49.7	50.1	25.8	26.2	34.9
LLaMA-MoE-v2 (E8A2) [†]	3.49B	1.2B	51.3	67.0	56.6	52.9	25.7	35.1	48.1
CMoE (E8S1A1) [†]	3.49B	1.2B	55.0	77.5	57.1	54.1	27.6	38.8	51.7
DOT-MoE	3.49B	1.2B	72.5	94.3	69.3	62.5	40.9	60.2	66.6
<i>LLaMA-3 8B</i>									
Dense	8.03B	15T*	83.2	96.2	79.6	77.3	58.3	82.1	79.4
CMoE (E8A1S1)	3.80B	1.2B	71.1	94.4	69.5	59.5	38.2	55.3	64.7
DOT-MoE	3.80B	1.2B	75.0	94.2	70.2	63.8	42.4	61.1	67.8
LLaMA-MoE-v2 (E8A2) [‡]	3.80B	7B	74.6	94.5	69.3	60.5	42.8	59.0	66.8
LLaMA-MoE-v2 (E8A1S1) [‡]	3.80B	7B	76.9	92.8	67.9	58.4	40.2	53.7	65.0
DOT-MoE	3.80B	7B	75.4	96.2	73.3	66.1	49.1	66.0	71.0
<i>Qwen2.5 7B</i>									
Dense	7.62B	18T*	88.0	96.6	78.6	76.2	64.1	80.3	80.6
CMoE (E8A1S1)	3.34B	1.2B	69.4	93.8	67.6	59.0	37.8	57.2	64.1
DOT-MoE	3.34B	1.2B	75.2	95.0	70.4	61.3	42.8	58.4	67.2
DOT-MoE	4.76B	1.2B	81.3	94.6	73.1	68.1	49.8	67.3	73.4

[†]Results from (Pei et al., 2025). [‡]Results from (Qu et al., 2024).

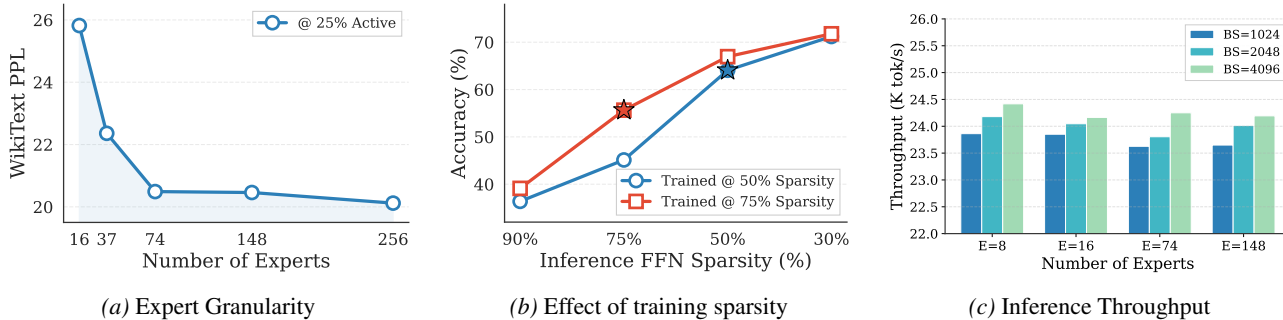


Figure 1. Ablation results for DOT-MoE. (a) Increasing expert granularity improves performance until saturation. (b) Training with higher FFN sparsity yields robust expert representations that generalize better to extreme sparsity regimes at inference time. (c) Inference throughput remains stable across expert granularities when active parameters are held constant.

adapt. This enables a stronger *zero-shot* transfer without training the model weights at all. As shown in Table 3, DOT-MoE achieves 61.5% average accuracy on LLaMA-2 7B, substantially outperforming CMoE (44.5%). A similar trend can be seen for other models. These results demonstrate that our output-aware expert construction optimizes the trade-off between sparsity and reconstruction fidelity more effectively than activation-based clustering.

Impact of Fine-tuning. While DOT-MoE is effective out-of-the-box, we investigate the impact of continuous fine-tuning. As shown in Table 2, fine-tuning DOT-MoE on 1.2B

tokens boosts LLaMA-2 7B accuracy from 61.5% to 66.6%, widening the gap against CMoE (51.7%) and LLaMA-MoE-v2 (48.1%). On LLaMA-3 8B, DOT-MoE achieves 67.8% with 1.2B tokens and improves to 71.0% when scaled to 7B tokens, outperforming LLaMA-MoE-v2 (66.8%) trained on the exact similar data split. This scaling behavior confirms that DOT-MoE provides a superior initialization for sparse models that continues to benefit from additional training data. It is worth noting that DOT-MoE substantially closes the gap between Dense and Sparse model with as few as 1.2B tokens of training. On Qwen2.5-7B DOT-MoE achieves an average accuracy of 73.4% whereas the dense

Table 3. Zero-shot performance comparison on standard common-sense reasoning benchmarks. DOT-MoE consistently outperforms existing structured pruning and dense-to-MoE conversion methods across multiple model families.

Method	Active Params	BoolQ [32, acc]	SciQ [0, acc]	PIQA [0, acc]	WinoG. [5, acc]	ARC-C [25, acc-n]	HellaSwag [10, acc-n]	Avg.
<i>LLaMA-2 7B</i>								
SliceGPT	3.73B	38.3	73.4	55.2	52.8	24.5	33.0	46.2
DISP-LLM	3.57B	60.0	83.5	65.2	57.0	29.4	49.5	57.4
CMoE (E8A1S1)	3.49B	46.1	65.3	52.8	48.7	23.8	30.1	44.5
DOT-MoE	3.49B	66.4	89.6	67.2	57.9	34.0	53.9	61.5
<i>LLaMA-3 8B</i>								
SliceGPT	3.90B	37.8	62.4	53.3	49.6	22.1	27.6	42.1
DISP-LLM	3.84B	51.0	84.5	60.6	53.9	26.3	40.1	52.7
CMoE (E8A1S1)	3.80B	41.0	51.1	53.9	51.0	25.3	28.6	41.8
DOT-MoE	3.80B	62.3	90.9	65.0	56.0	34.1	50.1	59.8
<i>Qwen2.5 7B</i>								
ShortGPT	4.82B	42.2	55.5	59.4	51.7	27.5	35.9	45.4
SliceGPT	4.75B	38.0	80.6	56.5	53.5	22.8	33.5	47.5
DISP-LLM	4.91B	65.7	94.1	70.2	62.8	45.9	61.3	66.7
CMoE (E8A2S2)	4.76B	60.3	87.0	62.8	51.5	28.4	42.7	55.5
DOT-MoE	4.76B	80.1	95.0	74.2	66.2	50.3	67.9	72.3
DISP-LLM	3.67B	46.8	84.8	63.5	57.2	35.1	46.4	55.6
CMoE (E8A1S1)	3.34B	43.1	41.8	54.2	50.4	25.0	27.1	40.3
DOT-MoE	3.34B	64.3	93.9	68.5	58.7	39.6	55.0	63.3

pre-trained model has an average accuracy of 80.6%. DOT-MoE also scales to larger models: on Qwen2.5-32B it improves over CMoE by +34.3 points, and maintains consistent gains across context lengths up to 32K tokens (Appendix K). It would be interesting to study the scaling behavior of dense-to-MoE models in the ~100B token regime but we leave that to future work due to compute constraints.

4.3. Ablation Studies

4.3.1. EXPERT GRANULARITY

In Figure 1a, we investigate the effect of expert granularity on performance by varying the total number of experts ($E \in \{16, 37, 74, 148, 256\}$) in DOT-MoE on Qwen2.5 7B. One might argue that DOT-MoE exhibits better performance due to a higher number of experts. However prior Dense-to-MoE methods (Pei et al., 2025; Qu et al., 2024) observed that increasing expert count from 8 to 16 leads to minimal improvement or degraded performance due to increased routing complexity, whereas DOT-MoE maintains stable performance at much higher expert counts. To test this, we trained a CMoE model with Qwen2.5-7B backbone. We set the total number of experts to 37 and active experts to 9; and observed a >5K WikiText perplexity for CMoE. In case of DOT-MoE, increasing the number of experts initially improves performance, but gains saturate as granularity becomes excessively fine, consistent with observations in

OLMoE (Muennighoff et al., 2025). To control for the granularity confound, we also run DOT-MoE at CMoE’s own setting ($E=8, \text{top-}k=2$) and still observe consistent gains across three model families (Appendix J).

Observation 1: Routing benefits from fine-grained experts up to a point beyond which additional experts provide limited returns.

4.3.2. EFFECT OF EXPERT GRANULARITY ON INFERENCE SPEED

A natural concern with fine-grained experts is inference overhead: does increasing the number of experts slow down generation? To investigate, we benchmark inference throughput using vLLM’s fused MoE kernels (Kwon et al., 2023) across four expert configurations ($E \in \{8, 16, 74, 148\}$) while holding active parameters constant at 25% of the FFN.

Figure 1c shows throughput (tokens/sec) across batch sizes. Crucially, throughput remains stable as expert count increases. This is because vLLM’s fused MoE implementation batches all expert computations into a small number of large GEMMs rather than executing each expert separately. The fused weights are stored as $\mathbf{W}_{\text{fused}} = [\mathbf{W}_1, \dots, \mathbf{W}_E]$ along the expert dimension, and token reordering enables a single large matrix multiplication regardless of expert count. Since the total fused intermediate dimension ($E \times s$) and ac-

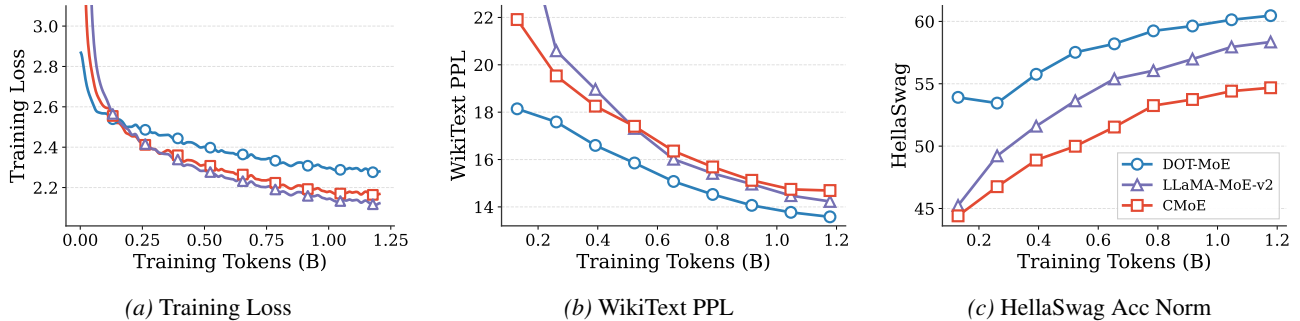


Figure 2. Effect of initialization on training dynamics. DOT-MoE starts with substantially lower training loss and WikiText perplexity, maintaining this advantage throughout fine-tuning. This translates to consistently higher downstream accuracy on HellaSwag.

tive neurons per token ($k \times s$) remain constant, the GEMM sizes and thus throughput are largely unaffected by expert granularity.

Observation 2: Fine-grained experts incur no throughput penalty with fused MoE kernels when active parameters are held constant.

4.3.3. EFFECT OF TRAINING SPARSITY.

A key advantage of our method is the ability to dynamically adjust the number of active experts at inference time, enabling flexible compute-accuracy trade-offs without alignment or fine-tuning. We investigate how the sparsity level used during training affects the model’s generalization across different inference-time sparsity configurations. We train two Qwen2.5-7B models with different FFN sparsity levels (50% and 75%) and evaluate both across a range of inference sparsities (30%, 50%, 75%, 90%). Figure 1b reports the average accuracy across the same six benchmarks reported in Table 3 as a function of FFN sparsity at inference time. Individual benchmark results are present in the Appendix Table 8. We observe an interesting behavior - models aligned with a higher sparsity level yields more robust expert representations across inference sparsities. For example, the model trained at 75% sparsity consistently outperforms the 50% trained model across varying inference sparsities. This behavior can be explained by our reconstruction objective. When trained with fewer active experts, the model learns to encode information more efficiently within each expert, resulting in more compact and discriminative representations.

Observation 3: Training at higher sparsity yields experts that generalize better across varying inference sparsities.

4.3.4. EFFECT OF INITIALIZATION ON TRAINING DYNAMICS

We investigate how different expert construction strategies affect fine-tuning dynamics by comparing DOT-MoE,

CMoE, and LLaMA-MoE-v2 on LLaMA-3 8B. All methods are trained on the same data split at 25% FFN sparsity. Figure 2 shows training loss, WikiText perplexity, and zero-shot HellaSwag normalized accuracy as a function of training tokens.

DOT-MoE exhibits a clear advantage at initialization, starting with substantially lower training loss compared to CMoE and LLaMA-MoE-v2. While all methods reduce training loss over time, CMoE and LLaMA-MoE-v2 exhibit signs of overfitting: despite achieving lower training loss, they consistently show higher WikiText perplexity and lower HellaSwag accuracy compared to DOT-MoE. In contrast, DOT-MoE continues to improve on both validation perplexity and downstream task performance throughout training, demonstrating that its expert construction leads to more distinct and generalizable representations.

These results show that output-aware expert construction provides not only a better starting point but also a more robust learning potential. The quality of the initial expert assignment directly impacts generalization, validating our approach of jointly optimizing neuron assignment and routing during the alignment phase.

Observation 4: Output-aware initialization achieves superior training generalization whereas heuristic methods exhibit overfitting.

5. Future Work and Conclusion

In this work, we introduced DOT-MoE, a novel framework that formulates the conversion of dense FFNs into sparse MoEs as a Differentiable Optimal Transport problem. By jointly learning the neuron-to-expert assignment and the routing policy via Straight-Through Estimators, we achieve a superior trade-off between sparsity and performance compared to heuristic clustering or structured pruning methods. We further show that the same balanced-transport formulation generalizes to multi-head attention, yielding substantial gains when applied to attention-head assignment. Several

promising directions remain for future research. First, while we currently initialize the affinity matrix \mathbf{A} randomly, exploring data-driven initializations such as leveraging weight correlations or pre-computed activation statistics could accelerate Sinkhorn convergence and yield tighter clusters. Second, we plan to investigate the hard pruning of experts that exhibit consistently low utilization during training. Permanently removing these experts could reduce the model’s memory footprint beyond just inference FLOPs, bridging the gap between MoEfication and model compression.

Acknowledgements

We thank Aditi Raghunathan and Sankalp Dayal for valuable feedback on experimental design and ablation studies.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning by improving the inference efficiency of Large Language Models. By reducing the computational cost required for deployment, our method contributes to lowering the energy consumption and carbon footprint of foundation models. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

- Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. GQA: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.
- Ashkboos, S., Croci, M. L., Nascimento, M. G. d., Hoefler, T., and Hensman, J. SliceGPT: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.
- Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.
- Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.
- Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.
- Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.
- Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transportation distances, 2013. URL <https://arxiv.org/abs/1306.0895>.
- Dong, P., Li, L., Tang, Z., Liu, X., Pan, X., Wang, Q., and Chu, X. Pruner-zero: Evolving symbolic pruning metric from scratch for large language models. In *International Conference on Machine Learning*, pp. 11346–11374. PMLR, 2024.
- Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.
- Frantar, E. and Alistarh, D. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pp. 10323–10337. PMLR, 2023.
- Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac’h, A., Li, H., McDonnell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. The language model evaluation harness, 07 2024a. URL <https://zenodo.org/records/12608602>.

- Gao, S., Lin, C.-H., Hua, T., Tang, Z., Shen, Y., Jin, H., and Hsu, Y.-C. Disp-llm: Dimension-independent structural pruning for large language models. *Advances in Neural Information Processing Systems*, 37:72219–72244, 2024b.
- Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., Casas, D. d. l., Hanna, E. B., Bressand, F., et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.
- Knight, P. A. The sinkhorn–knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 30(1):261–275, 2008.
- Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.
- Lele, N., Chavan, A., Thakur, A., and Gupta, D. Rethinking the value of training-free structured pruning of LLMs. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL <https://openreview.net/forum?id=7KkytYyhMv>.
- Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.
- Lin, C.-H., Gao, S., Smith, J. S., Patel, A., Tuli, S., Shen, Y., Jin, H., and Hsu, Y.-C. Modegpt: Modular decomposition for large language model compression. *arXiv preprint arXiv:2408.09632*, 2024.
- Ma, X., Fang, G., and Wang, X. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.
- Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008.
- Men, X., Xu, M., Zhang, Q., Yuan, Q., Wang, B., Lin, H., Lu, Y., Han, X., and Chen, W. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, 2025.
- Muennighoff, N., Soldaini, L., Groeneveld, D., Lo, K., Morrison, J., Min, S., Shi, W., Walsh, P., Tafjord, O., Lambert, N., Gu, Y., Arora, S., Bhagia, A., Schwenk, D., Wadden, D., Wettig, A., Hui, B., Dettmers, T., Kiela, D., Farhadi, A., Smith, N. A., Koh, P. W., Singh, A., and Hajishirzi, H. Olmoe: Open mixture-of-experts language models, 2025. URL <https://arxiv.org/abs/2409.02060>.
- OLMo, T., Walsh, P., Soldaini, L., Groeneveld, D., Lo, K., Arora, S., Bhagia, A., Gu, Y., Huang, S., Jordan, M., Lambert, N., Schwenk, D., Tafjord, O., Anderson, T., Atkinson, D., Brahman, F., Clark, C., Dasigi, P., Dziri, N., Ettinger, A., Guerquin, M., Heineman, D., Ivson, H., Koh, P. W., Liu, J., Malik, S., Merrill, W., Miranda, L. J. V., Morrison, J., Murray, T., Nam, C., Poznanski, J., Pyatkin, V., Rangapur, A., Schmitz, M., Skjongsberg, S., Wadden, D., Wilhelm, C., Wilson, M., Zettlemoyer, L., Farhadi, A., Smith, N. A., and Hajishirzi, H. 2 olmo 2 furious, 2025. URL <https://arxiv.org/abs/2501.00656>.
- Ouderaa, T. F., Nagel, M., Van Baalen, M., Asano, Y. M., and Blankevoort, T. The llm surgeon. *arXiv preprint arXiv:2312.17244*, 2023.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- Pei, Z., Zou, L., Zhen, H.-L., Yu, X., Liu, W., Pan, S. J., Yuan, M., and Yu, B. Cmoe: Converting mixture-of-experts from dense to accelerate llm inference, 2025. URL <https://arxiv.org/abs/2502.04416>.
- Qu, X., Dong, D., Hu, X., Zhu, T., Sun, W., and Cheng, Y. Llama-moe v2: Exploring sparsity of llama from perspective of mixture-of-experts with post-training. *ArXiv*, abs/2411.15708, 2024. URL <https://api.semanticscholar.org/CorpusID:274234365>.
- Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL <https://doi.org/10.1145/3474381>.
- Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. The sparsely-gated mixture-of-experts layer. *Outrageously large neural networks*, 2, 2017.
- Sinkhorn, R. and Knopp, P. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

- Song, J., Oh, K., Kim, T., Kim, H., Kim, Y., and Kim, J.-J. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. *arXiv preprint arXiv:2402.09025*, 2024.
- Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*.
- Team, Q. Qwen2.5: A party of foundation models, September 2024. URL <https://qwenlm.github.io/blog/qwen2.5/>.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and finetuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- Wang, Z., Wohlwend, J., and Lei, T. Structured pruning of large language models. In *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)*, pp. 6151–6162, 2020.
- Welbl, J., Liu, N. F., and Gardner, M. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*, 2017.
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.
- Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report, 2025. URL <https://arxiv.org/abs/2505.09388>.
- Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.
- Zhang, Z., Lin, Y., Liu, Z., Li, P., Sun, M., and Zhou, J. MoEfication: Transformer feed-forward layers are mixtures of experts. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 877–890, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.71. URL <https://aclanthology.org/2022.findings-acl.71/>.
- Zheng, H., Bai, X., Liu, X., Mao, Z. M., Chen, B., Lai, F., and Prakash, A. Learn to be efficient: build structured sparsity in large language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems, NIPS ’24*, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.
- Zhu, T., Qu, X., Dong, D., Ruan, J., Tong, J., He, C., and Cheng, Y. LLaMA-MoE: Building mixture-of-experts from LLaMA with continual pre-training. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15913–15923, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.890. URL <https://aclanthology.org/2024.emnlp-main.890/>.
- Zoph, B., Bello, I., Kumar, S., Du, N., Huang, Y., Dean, J., Shazeer, N., and Fedus, W. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

Appendix

A. The Assignment-Routing Gap

The central claim of our method is that existing dense-to-MoE approaches optimize inadequate proxies for the FFN output, whereas DOT-MoE is output-aware. We provide two complementary pieces of evidence: a single-layer reconstruction analysis that isolates the assignment strategy under a fixed evaluation protocol, and a full-pipeline ablation that isolates it from training-recipe effects.

A.1. Single-Layer Reconstruction

We conduct a controlled single-layer analysis that directly measures reconstruction fidelity under different expert assignment strategies. For each method, we compute the mean squared error (MSE) between the output of the original dense FFN and its sparse MoE approximation, isolating the effect of expert construction and routing.

We analyze layer 31 of both LLaMA-2-7B and LLaMA-3-8B, partitioning the FFN intermediate dimension into experts of size $D = 128$ and applying top- $k = 10$ routing. Expert assignments are constructed using calibration data from the WikiText training split, and reconstruction error is evaluated on the WikiText-2 test set. We compare four assignment strategies: LLaMA-MoE v1, LLaMA-MoE v2, CMoE, and our proposed DOT-MoE. All methods share identical expert granularity and routing configurations within each model.

Table 4 shows a consistent performance gap across both LLaMA-2 and LLaMA-3. On LLaMA-2, LLaMA-MoE v1 incurs over $35\times$ higher reconstruction error than DOT-MoE, LLaMA-MoE v2 nearly $9\times$, and CMoE more than $2\times$. The same pattern holds for LLaMA-3, where random and proxy-based assignments yield substantially higher error, while DOT-MoE consistently achieves the lowest MSE. These results indicate that clustering neurons based on input-side statistics or intermediate activations is insufficient, and that preserving the FFN output requires explicitly modeling each neuron’s interaction with the down-projection and residual stream.

Table 4. Single-layer reconstruction loss (MSE) on layer 31 for LLaMA-2 and LLaMA-3. All models use $D = 128$ neurons per expert and top- $k = 10$ routing. Calibration is performed on WikiText training data, and evaluation is conducted on WikiText-2. Lower is better.

Method	LLaMA-2		LLaMA-3	
	Error	Rel.	Error	Rel.
LLaMA-MoE v1	10.62	+36x	2.91	+41.6x
LLaMA-MoE v2	2.92	+9.9x	0.43	+6.1x
CMoE	0.61	+2.1x	0.15	+2.1x
DOT-MoE	0.29	—	0.07	—

A.2. Output-Aware Assignment Ablation

To further isolate the contribution of the OT-based assignment from training-recipe effects, we run CMoE and DOT-MoE under an identical fine-tuning pipeline on Qwen2.5-7B at the same expert granularity ($E=8$, top- $k=2$), with the same data and training steps. Table 5 reports cosine similarity and MSE of hidden representations before and after the LM head against the dense teacher.

Table 5. Representation similarity to the dense teacher on Qwen2.5-7B ($E=8$, top- $k=2$) under an identical fine-tuning pipeline. DOT-MoE preserves the dense residual-stream and logit geometry substantially better than CMoE.

Method	Cos Sim (pre-LM)	MSE (pre-LM)	Cos Sim (post-LM)	MSE (post-LM)
CMoE	0.55	21.1	0.49	16.6
DOT-MoE	0.84	8.2	0.71	10.5

Combined with the single-layer reconstruction analysis above and the training-dynamics comparison (Figure 2), this confirms that the improvements come from the OT-based assignment itself and not from a stronger training recipe.

B. Implementation Details

B.1. Hyperparameters

Tables 6 and 7 summarize the hyperparameters used for MoEfication and supervised fine-tuning, respectively. All hyperparameters are kept consistent across model families unless otherwise noted.

Table 6. Hyperparameters for MoEfication.

Hyperparameter	Symbol	Value
Expert size	s	128
Sinkhorn iterations	N	50
Sinkhorn temperature	τ	0.1
Learning rate	η	5×10^{-4}
Weight decay	λ	10^{-4}
LR schedule	—	Cosine
Warmup ratio	—	0.2
Max gradient norm	—	1.0
Batch size	—	64
Sequence length	—	2048
KL loss weight	w_{kl}	2.0
CE loss weight	w_{ce}	1.0
Z-loss weight	w_z	10^{-3}
Load balancing weight	w_{lb}	10^{-2}

Table 7. Hyperparameters for supervised fine-tuning.

Hyperparameter	Value
Learning rate	5×10^{-5}
Warmup ratio	0.1
LR schedule	Cosine
Batch size	128
Training tokens	1.2B
Max gradient norm	1.0
Load balancing weight	10^{-4}

B.2. Numerical Stability

We implement several techniques to ensure numerical stability during training:

B.2.1. LOG-DOMAIN IMPLEMENTATION.

For numerical stability, we implement Sinkhorn iterations in log-space (Algorithm 1). This avoids underflow when τ is small and enables stable computation even with thousands of neurons and experts.

Algorithm 1 Log-Domain Sinkhorn for Balanced Assignment

Require: Assignment logits $\mathbf{A} \in \mathbb{R}^{d_{\text{in}} \times E}$, temperature τ , iterations N

Ensure: Soft Assignment Matrix $\mathbf{M}_{\text{soft}} \in [0, 1]^{d_{\text{in}} \times E}$

- 1: $\mathbf{K} \leftarrow \mathbf{A} / \tau$
 - 2: $\mathbf{u} \leftarrow \mathbf{0}, \quad \mathbf{v} \leftarrow \log(s) \cdot \mathbf{1}_E$
 - 3: **for** $t = 1$ to N **do**
 - 4: $\mathbf{u} \leftarrow -\log\text{sumexp}(\mathbf{K} + \mathbf{v}, \text{dim} = 1)$
 - 5: $\mathbf{v} \leftarrow \log(s \cdot \mathbf{1}_E) - \log\text{sumexp}(\mathbf{K} + \mathbf{u}, \text{dim} = 0)$
 - 6: **end for**
 - 7: $\mathbf{M}_{\text{soft}} \leftarrow \exp(\mathbf{K} + \mathbf{u} + \mathbf{v})$
-

B.2.2. ASSIGNMENT LOGITS PRECISION.

The assignment logits \mathbf{A} are maintained in FP32 regardless of model dtype to ensure stable Sinkhorn iterations. Router weights \mathbf{W}_r use the model’s native dtype.

B.2.3. TEMPERATURE ANNEALING.

We linearly anneal the Sinkhorn temperature from $\tau_{\text{start}} = 1.0$ to $\tau_{\text{end}} = 0.1$ during the warmup phase. Higher temperatures early in training allow exploration of the assignment space; lower temperatures sharpen assignments as training progresses. Validation always uses τ_{end} .

C. Effect of Training Sparsity

Table 8 presents the complete per-benchmark results for the training sparsity ablation discussed in Section 4. We train

two Qwen2.5-7B models at 50% and 75% FFN sparsity, then evaluate each across four inference sparsity levels.

Table 8. Detailed benchmark performance under different training and inference sparsity configurations on Qwen2.5-7B. Models trained at higher sparsity (75%) generalize better to extreme sparsity regimes at inference time, while both converge at low sparsity (30%).

Inference Sparsity	ARC-e [acc-n]	ARC-c [acc-n]	PIQA [acc-n]	WinoG. [acc]	HellaS. [acc-n]	Avg.
<i>Trained at 50% FFN Sparsity</i>						
90%	26.64	25.26	52.07	49.80	28.13	36.38
75%	42.85	26.62	61.70	53.28	41.20	45.13
50%	70.50	45.65	73.72	63.38	67.11	64.07
30%	79.08	54.27	78.62	68.90	75.16	71.21
<i>Trained at 75% FFN Sparsity</i>						
90%	34.81	21.93	56.04	50.28	32.56	39.12
75%	59.97	35.24	68.82	59.98	54.21	55.64
50%	72.77	48.55	76.22	66.77	70.45	66.95
30%	80.35	53.92	79.27	69.53	75.83	71.78

D. Training Objective

We jointly optimize the affinity matrix \mathbf{A} and router weights \mathbf{W}_r to minimize the discrepancy between dense and sparse outputs. Given a sequence of T tokens, let $\mathbf{z}_t^{\text{dense}}, \mathbf{z}_t^{\text{MoE}} \in \mathbb{R}^V$ denote the output logits from the dense teacher and MoE student at position t , respectively.

KL Divergence Loss. We distill the dense model’s output distribution into the MoE student. This loss directly minimizes the distribution gap between teacher and student outputs.

Cross-Entropy Loss. We additionally train on the standard language modeling objective. This ensures the MoE model maintains language modeling capability on the training distribution.

Router Z-Loss. Following Zoph et al. (2022), we penalize large router logits to improve numerical stability:

$$\mathcal{L}_z = \frac{1}{T} \sum_{t=1}^T \left(\log \sum_{e=1}^E \exp(L_{t,e}) \right)^2 \quad (13)$$

where $L_{t,e}$ is the router logit (Equation 8) for expert e at token position t . This loss prevents the router from producing extremely large logits that can cause overflow in the softmax computation.

Load Balancing Loss. To prevent expert collapse and encourage uniform utilization, we include the auxiliary balancing loss from Shazeer et al. (2017):

$$\mathcal{L}_{\text{bal}} = E \cdot \sum_{e=1}^E f_e \cdot \bar{p}_e \quad (14)$$

where $f_e = |\{i : e \in \mathcal{I}_i\}|/n$ is the fraction of tokens routed to expert e , and $\bar{p}_e = \frac{1}{n} \sum_{i=1}^n P_{i,e}$ is the average routing probability for expert e . The product $f_e \cdot \bar{p}_e$ is minimized when experts are utilized uniformly.

Total Objective. The complete training loss combines all components:

$$\mathcal{L} = w_{\text{kl}} \mathcal{L}_{\text{KL}} + w_{\text{ce}} \mathcal{L}_{\text{CE}} + w_z \mathcal{L}_z + w_{\text{bal}} \mathcal{L}_{\text{bal}} \quad (15)$$

where $w_{\text{kl}}, w_{\text{ce}}, w_z, w_{\text{bal}}$ are weighting hyperparameters (see Table 6).

E. Sparse MoE Computation

During the alignment phase, we simulate sparse MoE computation without materializing separate expert weights. Given the neuron assignment \mathbf{M} and routing mask \mathbf{R} , we compute the sparse MoE output by masking the intermediate activations:

$$\hat{\mathbf{Y}} = (\mathbf{H} \odot (\mathbf{R}\mathbf{M}^\top)) \mathbf{W}_{\text{down}} \quad (16)$$

where $\mathbf{H} \in \mathbb{R}^{n \times d_{\text{fin}}}$ is the intermediate activation from Eq. 1 and \odot denotes element-wise multiplication.

The matrix product $\mathbf{R}\mathbf{M}^\top \in \{0, 1\}^{n \times d_{\text{fin}}}$ composes two levels of selection:

- $\mathbf{R} \in \{0, 1\}^{n \times E}$: which experts are active for each token
- $\mathbf{M}^\top \in \{0, 1\}^{E \times d_{\text{fin}}}$: which neurons belong to each expert
- $\mathbf{R}\mathbf{M}^\top \in \{0, 1\}^{n \times d_{\text{fin}}}$: which neurons are active for each token

Since each token activates k experts and each expert contains s neurons, only $k \cdot s$ out of d_{fin} neurons contribute to each token’s output. This masking-based formulation enables efficient training on the original dense weights while simulating sparse computation.

After alignment training converges, we extract the final binary assignment \mathbf{M} and use it to partition the dense FFN weights into E separate expert modules. For each expert e , we slice the corresponding rows from $\mathbf{W}_{\text{gate}}, \mathbf{W}_{\text{up}}$, and columns from \mathbf{W}_{down} according to the neurons in cluster $\mathcal{C}_e = \{i : M_{i,e} = 1\}$. The resulting model is a standard MoE architecture compatible with existing sparse inference frameworks.

F. Expert Specialization and Utilization

Expert Specialization. To understand the learned behavior of our converted MoE model, we visualize expert output activations using t-SNE (Maaten & Hinton, 2008). We collect

activation vectors from the expert outputs across a diverse set of input samples and project them into two dimensions. Figure 3 shows the resulting visualization for layer 9, where each color represents a different expert. The visualization reveals clear clustering structure, indicating that experts learn to specialize in processing distinct types of inputs. Activations from the same expert tend to cluster together in the embedding space, forming well-separated regions. The clear separation between expert clusters suggests that our method successfully partitions the representation space, with minimal redundancy across experts.

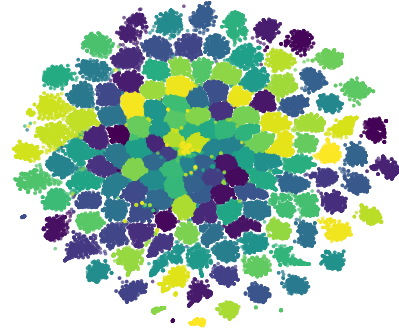


Figure 3. t-SNE visualization of expert output activations at layer 9 for Qwen2.5-7B. Each color represents a different expert. The clear clustering indicates that experts learn distinct, well-separated representations.

Expert Utilization. To analyze routing behavior after MoEfication, we collect expert token allocation statistics across all transformer layers on the WikiText-2 dataset for Qwen2.5-7B with 50% sparsity. Figure 4 visualizes the proportion of tokens routed to each expert at every layer. Overall, expert routing remains well balanced across most layers, with no evidence of severe expert collapse.

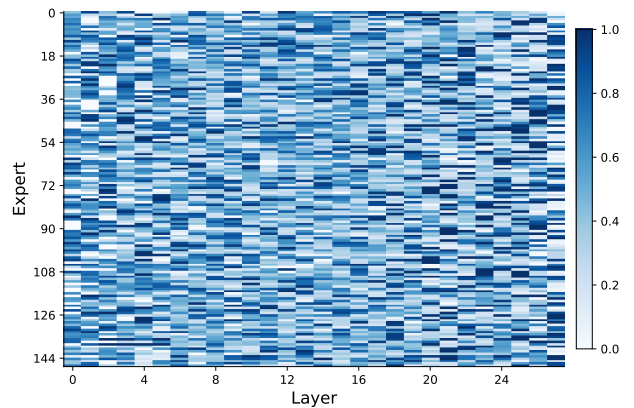


Figure 4. Expert token allocation across transformer layers for Qwen2.5-7B with 50% sparsity on the WikiText-2 dataset.

Table 9. Comparison with LTE on LLaMA-2-7B ($E=86$) at matched FFN sparsity. DOT-MoE outperforms LTE by +6.0 points on average while keeping per-token compute constant through top- k routing.

Method	Active FFN	BoolQ	SciQ	PIQA	WinoG.	ARC-C	HellaS.	Avg.
LTE ($\eta=3.0$)	29%	68.9	82.5	59.5	62.7	33.9	56.2	60.6
DOT-MoE	25%	72.5	94.3	69.3	62.5	40.9	60.2	66.6

Table 10. Attention MoEfication on Qwen2.5-7B at 50% attention sparsity. OT-based head assignment outperforms random assignment by +17.9 points on average.

Method	BoolQ	SciQ	PIQA	WinoG.	ARC-C	HellaS.	Avg.
Random assign. + router	45.1	60.5	56.8	51.8	27.7	35.2	46.2
DOT-MoE (Attn)	60.1	87.2	74.2	56.0	44.4	62.6	64.1

G. Extension to Attention Layers

DOT-MoE extends to multi-head attention by treating heads as the units to be grouped into experts, mirroring the FFN setting where neurons are grouped. Given N_h query heads of dimension d_h , we form $E_{\text{attn}} = N_h/s_h$ experts of s_h heads and activate k_{attn} per token via a separate router $\mathbf{W}_r^{\text{attn}}$. A learnable affinity matrix $\mathbf{A}_{\text{attn}} \in \mathbb{R}^{N_h \times E_{\text{attn}}}$ with marginals $(\mathbf{1}_{N_h}, s_h \mathbf{1}_{E_{\text{attn}}})$ is optimized via the same log-domain Sinkhorn iterations and discretized through the same STE as Eq. 11. During training, all heads are computed with frozen dense weights; sparsity is realized by masking the concatenated head outputs before the output projection \mathbf{W}_O , yielding an expression identical in form to the FFN reconstruction (Eq. 16). For Grouped Query Attention (Ainslie et al., 2023), assignment operates at query-head granularity and all KV heads are computed unconditionally; sparsity therefore lives in the Q and O projections, which dominate attention parameters.

To validate the formulation, we evaluate attention-only DOT-MoE on Qwen2.5-7B at 50% attention sparsity ($N_h=28, E_{\text{attn}}=14, s_h=2, k_{\text{attn}}=7$) against a random head-assignment baseline with a trained router under the same training recipe. Table 10 shows that OT-based assignment outperforms the baseline by +17.9 points on average (64.1 vs. 46.2). Two points are worth noting. First, the search space is far smaller than for the FFN (28 heads vs. 18,944 neurons), so attention assignment converges faster and with less gradient variance; the attention router itself matches the FFN router architecturally and adds negligible compute. Second, because attention parameters are roughly one third of FFN parameters, attention-only MoEfication yields modest overall compression; combining it with FFN MoEfication (joint MLP+Attention) is a direct extension.

H. Training Overhead

We profile DOT-MoE’s alignment phase on $8 \times \text{H100}$ to quantify the cost of Sinkhorn iterations and straight-through

estimation relative to a standard dense forward/backward pass. Sinkhorn iterations account for only $\sim 2\%$ of the total forward-and-backward time. All DOT-MoE-specific operations combined add $\sim 15\%$ overhead per training step over a standard dense training step; most of this overhead comes from hard-assignment matrix construction rather than from Sinkhorn itself. We currently run the greedy rounding on CPU and incur CPU-to-GPU transfer overhead; a dedicated GPU kernel (e.g., using a parallel bucketing or priority-queue primitive) would remove most of this cost. Crucially, this overhead is incurred *only during the alignment phase*: once alignment converges, the extracted MoE model is a standard fused-expert architecture with no Sinkhorn or STE at inference time.

I. Additional Dense-to-MoE Baselines

I.1. Comparison with LTE

LTE (Zheng et al., 2024) controls sparsity through a scalar η and activates a variable number of experts per token via sigmoid thresholding. To enable a fair comparison, we evaluate DOT-MoE at matching FFN sparsity on LLaMA-2-7B with $E=86$ experts (Table 9).

At *lower* FFN sparsity (25% vs. 29%), DOT-MoE still outperforms LTE by +6.0 points on average. Beyond accuracy, LTE’s sigmoid routing activates a variable number of experts per token, leading to unpredictable per-token compute; DOT-MoE uses softmax top- k routing, which keeps compute per token constant and is compatible with standard fused-MoE serving kernels.

I.2. Positioning Among Dense-to-MoE Methods

To clarify why our primary baselines are CMoE and LLaMA-MoE(-v2), we compare the structural settings of dense-to-MoE methods in Table 11. Methods that use ReLU-specific activation patterns (MoEfication, DejaVu), inflate total parameters (Read-ME), or use variable-compute rout-

Table 11. Structural settings of dense-to-MoE conversion methods. Our primary baselines match DOT-MoE’s problem setting: parameter-preserving, activation-agnostic, fixed-active-params per token, and softmax top- k routing compatible with standard MoE serving frameworks.

Method	Param. Preserv.	Act. Agnostic	Fixed Compute	Routing	Expert Assignment	MoE Serve
MoEfication	✓	× (ReLU)	✓	Top- k	Weight clustering	✓
DejaVu	✓	× (ReLU)	×	Per-input pred.	Activation pred.	×
LLaMA-MoE	✓	✓	✓	Top- k	Random partition	✓
LLaMA-MoE-v2	✓	✓	✓	Top- k	Act./grad. heuristic	✓
CMoE	✓	✓	✓	Top- k	Co-activation clust.	✓
LTE	✓	✓	×	Sigmoid+thresh.	k -means + router	×
Read-ME	× (2.4×)	✓	✓	Pre-gating	Activation-based	✓
DOT-MoE	✓	✓	✓	Top-k	Learned (OT)	✓

Table 12. Controlled same-granularity comparison ($E=8$, top- $k=2$, 1.2B fine-tuning tokens). At CMoE’s own granularity, DOT-MoE consistently outperforms CMoE across all three architectures.

Model	Method	BoolQ	SciQ	PIQA	WinoG.	ARC-C	HellaS.	Avg.
Qwen2.5-7B	CMoE	69.4	93.8	67.6	59.0	37.8	57.2	64.1
	DOT-MoE	75.3	94.6	69.3	63.7	45.8	57.6	67.7
LLaMA-2-7B	CMoE	55.0	77.5	57.1	54.1	27.6	38.8	51.7
	DOT-MoE	73.0	93.2	70.5	61.3	40.2	58.2	66.0
LLaMA-3-8B	CMoE	71.1	94.4	69.5	59.5	38.2	55.3	64.7
	DOT-MoE	73.8	94.6	70.5	62.7	40.4	60.3	67.0

ing (DejaVu, LTE) address a different problem setting than ours. Our baselines share the parameter-preserving, fixed-active-params, softmax top- k setting.

Read-ME is excluded as a primary baseline because it inflates parameters by 2.4× (7B dense to 17B MoE), placing it in the upcycling rather than parameter-preserving category. MoEfication and DejaVu are excluded because they were designed for ReLU-based encoder architectures and do not transfer to SwiGLU decoder LLMs.

J. Controlled Same-Granularity Comparison

A natural concern in dense-to-MoE comparisons is that different expert granularities can confound the contribution of the assignment strategy. To address this, we run DOT-MoE at CMoE’s own granularity ($E=8$, top- $k=2$) with 1.2B fine-tuning tokens across three model families. Table 12 reports the controlled comparison.

At CMoE’s own granularity, DOT-MoE outperforms CMoE by +3.6 on Qwen2.5-7B, +14.3 on LLaMA-2-7B, and +2.3 on LLaMA-3-8B. Moreover, DOT-MoE at $E=8$ nearly matches DOT-MoE at $E=148$ on the same active-parameter budget (e.g., 67.7 vs. 67.2 on Qwen2.5-7B), indicating that DOT-MoE’s advantage is not an artifact of finer expert granularity. We also verified that CMoE at DOT-MoE’s default

granularity ($E=148$) achieves 64.1 avg on Qwen2.5-7B versus DOT-MoE’s 67.2, so the conclusion holds in both directions.

K. Scalability

K.1. Scaling to 32B Parameters

DOT-MoE’s alignment phase only trains assignment logits and router weights (under 2% of model parameters), so the method itself is not the bottleneck at scale; the frozen dense model’s forward pass dominates cost. To verify scalability, we evaluate DOT-MoE on Qwen2.5-32B at 25% active parameters (Table 13).

Table 13. Scalability to Qwen2.5-32B at 25% active parameters. Zero-shot performance on common-sense reasoning benchmarks.

Method	BoolQ	SciQ	PIQA	WinoG.	ARC-C	HellaS.	Avg.
CMoE	43.5	34.1	52.8	49.6	23.7	28.8	38.8
DOT-MoE	81.0	95.4	74.8	67.6	49.1	70.6	73.1

At 32B parameters, DOT-MoE improves the benchmark average by +34.3 points (73.1 vs. 38.8) over CMoE, confirming that the OT-based assignment holds up as model scale increases.

K.2. Robustness to Sequence Length

The neuron-to-expert assignment and router operate per token and are independent of sequence length, so DOT-MoE applies directly to longer contexts. We evaluate WikiText-2 word perplexity on Qwen2.5-7B (with 1.2B fine-tuning tokens) using rolling log-likelihood with varying maximum context windows. The document-level WikiText¹ split is used; documents exceeding the context window are split into rolling windows with sliding overlap.

Table 14. WikiText-2 word perplexity on Qwen2.5-7B (with 1.2B fine-tuning tokens) at different maximum sequence lengths. DOT-MoE’s advantage over CMoE is consistent across context windows from 2K to 32K.

Method	2048	4096	8192	32768
Dense (Qwen2.5-7B)	9.50	9.03	8.78	8.73
CMoE (E8S1A1)	18.83	17.62	16.90	16.84
DOT-MoE (E8A2)	16.47	15.48	14.91	14.84

DOT-MoE maintains a consistent ~ 2 PPL improvement over CMoE across all context lengths up to 32K tokens, confirming that per-token routing is robust to long contexts.

¹EleutherAI/wikitext_document_level