RECIPEMOE: LEARNING REUSABLE BLOCK RECIPES FOR FULL-PARTICIPATION MIXTURE-OF-EXPERTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-Experts (MoE) scales capacity, but existing designs cannot set three quantities independently. **Participation** is how many experts are involved in producing a token's output, **execution** is how many are evaluated on the token, and **materialization** is how many expert-sized parameter sets must be built. Sparse routing keeps execution and materialization low, but shrinks participation: for each token, only a few experts participate. Dense output-mixing restores full participation, but its execution grows with the number of experts. Parameter-merging keeps execution at one expert, but its materialization grows with the number of routing decisions. We propose RecipeMoE, a block-conditioned MoE that decouples all three by pairing dense expert composition with sparse block execution. Its cacheable blocks come from a small learned codebook, one per entry. A lightweight hypernetwork generates block-specific composition recipes that merge each internal layer's expert pool into a composed expert. **Participation is full**, because every composed expert is synthesized from the entire pool. **Execution stays sparse**, because a router sends each token to only a few blocks. **Materialization is bounded**, because the codebook, not the input, fixes how many blocks exist. Dual-Path Residual Gating (DPRG) enables nonlinear interactions between two paths composed from the same expert pool using independent recipes. Experiments on image classification show consistent gains over representative sparse and dense MoE baselines. Additional experiments on language modeling and sequential recommendation validate its generalization beyond vision. RecipeMoE is fully deployed in a large-scale industrial generative recommendation system, serving hundreds of millions of users under a $60\,\mathrm{ms}$ latency budget, with a 2.4% relative UVCTR gain in online A/B testing. Code is available at `https://anonymous.4open.science/r/RecipeMoE-717C254/`.

1 INTRODUCTION

Mixture-of-Experts (MoE) architectures have been widely adopted in language models (Lepikhin et al., 2020; Fedus et al., 2022; Jiang et al., 2024; Dai et al., 2024), vision models (Riquelme et al., 2021; Fan et al., 2022), multimodal models (Mustafa et al., 2022; Xue et al., 2023; Li et al., 2024), and recommendation systems (Deng et al., 2025; Zhu et al., 2025). A standard MoE layer contains a pool of experts and a router that determines how the experts process each token. Existing designs follow three main strategies: *sparse routing* (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022; Dai et al., 2024), *dense output mixing* (Ma et al., 2018; Tang et al., 2020), and *parameter merging* (Muqeeth et al., 2023; Zhong et al., 2024).

Figure 1 compares MoE designs along three dimensions. **Participation** measures how many experts are involved in producing a token's output. **Execution** counts the expert transformations actually evaluated per token. **Materialization** counts the expert-sized parameter sets constructed for a batch. These dimensions remain coupled in existing approaches, resulting in the following trade-offs:

- **Sparse execution limits participation.** Sparse-routing methods bound execution by activating only a few experts for each token. The decision narrows participation: non-selected experts neither affect the token's output nor receive a learning signal from it.

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.

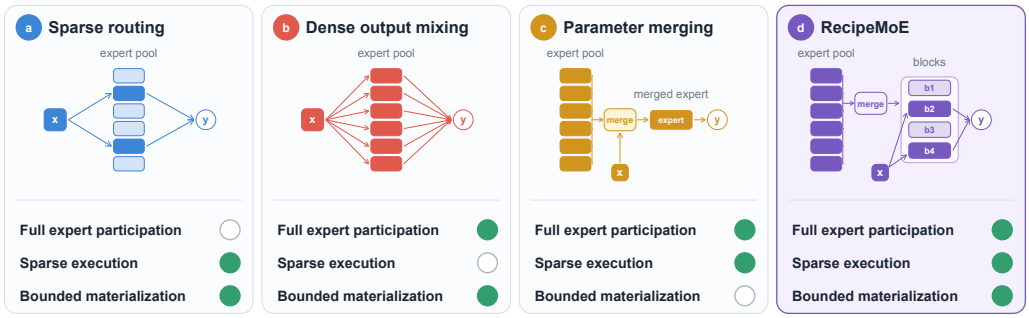

Figure 1: Comparison of MoE design strategies. Filled circles indicate that a strategy satisfies the corresponding property, while hollow circles indicate otherwise.

- **Full participation requires dense execution.** Dense output-mixing methods combine the outputs of all experts for each token. Because every expert must be evaluated, execution cost grows with the number of experts.
- **Single-expert execution increases materialization.** Parameter-merging methods combine the full expert pool into one composed expert and execute it once. Every distinct routing decision, however, needs its own copy of those composed weights, so the number of expert-sized parameter sets built at run time grows with the number of routing units.

Taken together, these trade-offs leave a central question: **Can each token draw on the full expert pool without requiring dense execution or token-specific parameter synthesis?** To address these trade-offs, we propose RecipeMoE, a block-conditioned MoE architecture that separates the construction of expert transformations from their execution on tokens. Learned composition recipes combine the full expert pool into a bounded set of reusable transformations, after which each token independently selects and executes only a few of them. RecipeMoE thereby achieves pool-wide expert participation, sparse execution, and bounded parameter materialization.

Our main contributions are summarized as follows:

- **Decoupling participation, execution, and materialization.** We propose RecipeMoE, which composes a finite set of reusable, cacheable blocks from the full expert pool and sparsely routes each token to a few blocks. This separates pool-wide participation from per-token execution and bounds parameter materialization.
- **Expressive expert composition.** We introduce Dual-Path Residual Gating (DPRG), which uses two independent recipes to compose value and gate paths from the same expert pool. The two compositions interact through multiplicative gating, enabling nonlinear interactions without enlarging the expert pool.
- **Visual evaluation and cross-domain generalization.** We demonstrate the effectiveness of RecipeMoE through extensive experiments on ImageNet-1K, supplemented by generalization tests on language modeling and sequential recommendation. We further validate its practical utility through full deployment in a large-scale industrial generative recommendation system, following a 2.4% relative UVCTR improvement in online A/B testing.

2 Related Work

Sparse Mixture-of-Experts. Sparsely gated MoEs activate only a subset of experts for each token (Shazeer et al., 2017). GShard (Lepikhin et al., 2020) and Switch Transformer (Fedus et al., 2022) use Top-2 and Top-1 routing, respectively, while V-MoE (Riquelme et al., 2021) extends sparse routing to vision. Expert Choice (Zhou et al., 2022) instead lets experts select tokens. DeepSeekMoE (Dai et al., 2024) and DeepSeek-V3 (Liu et al., 2024) refine expert organization and load balancing. ReMoE (Wang et al., 2025) and LapSum SoftMoE (Zasada et al., 2026) relax discrete routing. Dense2MoE (Zheng et al., 2025) introduces Mixture of Blocks (MoB) to selectively activate diffusion Transformer blocks. DynMoE (Guo et al., 2025) and MASS (Park & Park, 2026)

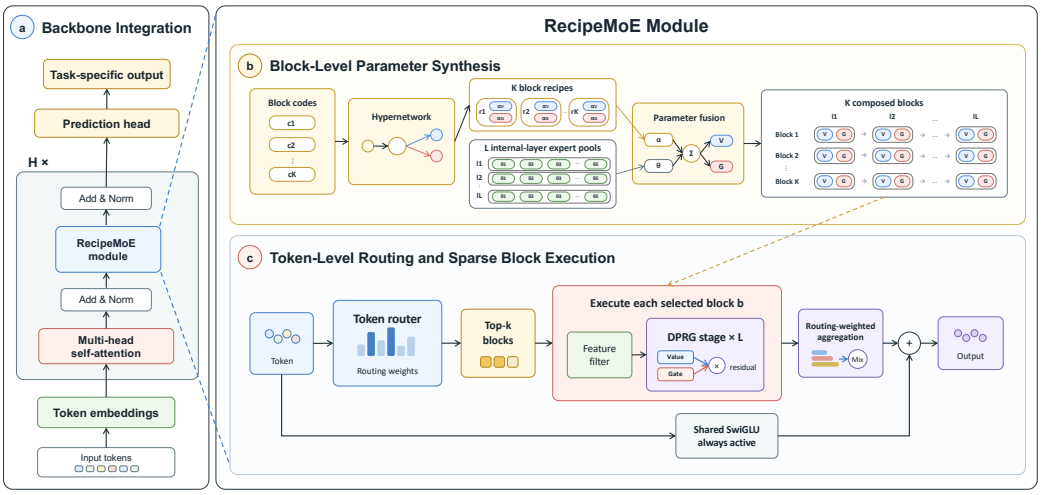

Figure 2: Overview of RecipeMoE. The left panel shows independent RecipeMoE modules integrated into a multi-layer Transformer backbone. The right panel shows block-level expert parameter synthesis and token-level Top-k block routing and execution.

adapt expert-pool sizes and routing cardinalities. MoBE (Chen et al., 2026) compresses pretrained sparse MoEs by reconstructing expert weights with expert-specific transformations and shared basis combinations. These approaches retain sparse execution, while improving routing, expert organization, or parameter efficiency.

Dense Output Mixing. MMoE (Ma et al., 2018) and PLE (Tang et al., 2020) combine expert outputs using task-specific gates and shared/task-specific expert structures, respectively. Full participation requires evaluating all experts in the relevant pool. Soft MoE (Puigcerver et al., 2024) instead mixes tokens into slots, processes each slot with its assigned expert, and maps the expert outputs back to tokens. Its original formulation is noncausal. μMoE (Oldfield et al., 2024) uses CP and Tensor Ring factorizations to compute a dense, input-dependent mixture of experts directly in factorized form.

Parameter Merging. SMEAR (Muqeeth et al., 2023) merges the full expert pool into one composite expert. Example-level merging shares parameters across a sequence and is noncausal when conditioned on the complete sequence; token-level merging allows token-specific compositions but requires a separate full-pool merge per token. Lory (Zhong et al., 2024) reuses a preceding-segment-conditioned composition within each segment, reducing merging frequency while sharing the same composed parameters across tokens in that segment. DSFNet (Yu et al., 2025) combines factor-scenario parameter branches to construct input-conditioned networks.

Hypernetworks and Parameter Generation. Hypernetworks (Ha et al., 2016) generate target-network parameters from conditioning embeddings. HyperMoE (Zhao et al., 2024) generates a token-conditioned HyperExpert using information associated with unselected experts and executes it alongside selected experts. RecipeMoE instead generates compact composition coefficients from a finite codebook of input-independent embeddings.

3 Method

As illustrated in Figure 2, RecipeMoE integrates into a Transformer backbone (Vaswani et al., 2017) by replacing its FFN sublayer. Each RecipeMoE module composes reusable multi-layer blocks from shared expert pools and sparsely routes each token to a few blocks. A codebook defines K token-independent blocks, each containing L composed layers. For input $\mathbf{x}_t \in \mathbb{R}^d$, the module executes k selected blocks and an always-active shared SwiGLU expert.

3.1 BLOCK-LEVEL PARAMETER SYNTHESIS

A shared hypernetwork generates value- and gate-path composition recipes from each block embedding. These coefficients combine expert parameters at each internal layer to construct the block's parameters.

Block codebook and expert pools. Each module maintains a codebook $\mathcal{C} = \{\mathbf{c}_b \in \mathbb{R}^{d_c}\}_{b=1}^{K}$, with one embedding per block. Let $d_0 = d_L = d$, with d_ℓ denoting the output dimension of internal layer ℓ. For each $\ell \in \{1, \dots, L\}$, the module maintains a layer-specific pool of E experts shared across the K blocks,

$$\mathcal{P}^{(\ell)} = \{(W_e^{(\ell)}, \mathbf{b}_e^{(\ell)})\}_{e=1}^{E}, \quad W_e^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell-1}}, \quad \mathbf{b}_e^{(\ell)} \in \mathbb{R}^{d_\ell}. \tag{1}$$

The pools are independent across internal layers and backbone layers and distinct from the always-active shared expert.

Composition coefficient generation. The block hypernetwork h_ϕ is shared across the K codebook entries within one RecipeMoE module. Its trunk takes only $\mathbf{c}_b$ as input:

$$\mathbf{q}_b = \mathrm{ReLU}(\mathrm{LN}(A\mathbf{c}_b + \mathbf{a})), \tag{2}$$

where $A \in \mathbb{R}^{d_h \times d_c}$ and $\mathbf{a} \in \mathbb{R}^{d_h}$ are trainable. Two linear heads produce value and gate composition coefficients:

$$\boldsymbol{\alpha}_b^v = A_v \mathbf{q}_b + \mathbf{a}_v, \qquad \boldsymbol{\alpha}_b^g = A_g \mathbf{q}_b + \mathbf{a}_g, \qquad \boldsymbol{\alpha}_b^v, \boldsymbol{\alpha}_b^g \in \mathbb{R}^E. \tag{3}$$

Here, $A_v, A_g \in \mathbb{R}^{E \times d_h}$ and $\mathbf{a}_v, \mathbf{a}_g \in \mathbb{R}^E$ are the trainable weights and biases of the two output heads.

Expert parameter composition. The coefficients are not normalized by softmax or sigmoid. They may be negative and need not sum to one, allowing composition over the linear span of the expert parameters rather than restricting it to their convex hull. We apply a factor of $1/\sqrt{E}$ to control the scale of the composed parameters as the expert pool grows. For path $p \in \{v, g\}$, the composed parameters at every internal layer are

$$\widetilde{W}_{b,p}^{(\ell)} = \frac{1}{\sqrt{E}} \sum_{e=1}^{E} \alpha_{b,e}^p W_e^{(\ell)}, \qquad \widetilde{\mathbf{b}}_{b,p}^{(\ell)} = \frac{1}{\sqrt{E}} \sum_{e=1}^{E} \alpha_{b,e}^p \mathbf{b}_e^{(\ell)}. \tag{4}$$

Each block shares its coefficient pair across the L layer-specific pools. The resulting parameters are independent of tokens and reusable across them.

3.2 TOKEN-LEVEL ROUTING AND SPARSE BLOCK EXECUTION

Block routing. For each token, a two-layer ReLU MLP $g_{\text{block-router}} : \mathbb{R}^d \to \mathbb{R}^K$ scores the blocks, and TopK selects the k highest-scoring indices:

$$\mathbf{r}_t = g_{\text{block-router}}(\mathbf{x}_t), \qquad \mathcal{B}_t = \mathrm{TopK}(\mathbf{r}_t, k), \tag{5}$$

The routing weight of a selected block b is

$$\pi_{t,b} = \mathrm{softmax}(\mathbf{r}_t)_b, \qquad b \in \mathcal{B}_t. \tag{6}$$

Block-conditioned feature filtering. Before entering a selected block, the token representation $\mathbf{x}_t$ is filtered using that block's codebook embedding $\mathbf{c}_b$:

$$\mathbf{m}_{t,b} = \mathrm{sigmoid}\left(\mathbf{W}_f[\mathbf{x}_t \parallel \mathbf{c}_b] + \mathbf{b}_f\right), \qquad \mathbf{z}_{t,b}^{(0)} = \mathbf{x}_t \odot \mathbf{m}_{t,b}. \tag{7}$$

Here, $[\cdot \parallel \cdot]$ denotes vertical concatenation of column vectors, $\mathbf{W}_f \in \mathbb{R}^{d \times (d+d_c)}$ and $\mathbf{b}_f \in \mathbb{R}^d$ are the learnable parameters of a feature-filtering layer shared across blocks. The mask $\mathbf{m}_{t,b} \in (0,1)^d$ gives each selected block a conditioned view of the token.

Dual-Path Residual Gating. DPRG uses two independent recipes to compose value and gate parameters from the same expert pool. The resulting paths interact through residual multiplicative gating.

For internal layer ℓ and input $\mathbf{z}_{t,b}^{(\ell-1)}$, the DPRG transformation is

$$\mathbf{v}_{t,b}^{(\ell)} = \widetilde{W}_{b,v}^{(\ell)} \mathbf{z}_{t,b}^{(\ell-1)} + \widetilde{\mathbf{b}}_{b,v}^{(\ell)}, \tag{8}$$

$$\mathbf{g}_{t,b}^{(\ell)} = \mathrm{RMSNorm}\left(\widetilde{W}_{b,g}^{(\ell)} \mathbf{z}_{t,b}^{(\ell-1)} + \widetilde{\mathbf{b}}_{b,g}^{(\ell)}\right), \tag{9}$$

$$\widetilde{\mathbf{z}}_{t,b}^{(\ell)} = \mathbf{v}_{t,b}^{(\ell)} \odot \left(\mathbf{1} + \lambda\ \mathrm{SiLU}(\mathbf{g}_{t,b}^{(\ell)})\right). \tag{10}$$

For $\ell < L$, we apply LayerNorm to $\widetilde{\mathbf{z}}_{t,b}^{(\ell)}$ to obtain $\mathbf{z}_{t,b}^{(\ell)}$. The final layer uses $\mathbf{z}_{t,b}^{(L)} = \widetilde{\mathbf{z}}_{t,b}^{(L)}$ without normalization. $F_b(\mathbf{x}_t) = \mathbf{z}_{t,b}^{(L)}$ is the final block output. The learnable residual scale λ is shared across the internal layers of a RecipeMoE module. DPRG adds only a constant factor to parameter-synthesis and execution costs.

Output aggregation and shared expert. We combine the selected block outputs with an always-active shared SwiGLU expert S:

$$\mathbf{y}_t = \sum_{b \in \mathcal{B}_t} \pi_{t,b} F_b(\mathbf{x}_t) + S(\mathbf{x}_t). \tag{11}$$

The shared path captures common information, allowing the routed blocks to focus on transformations that benefit from token-dependent selection.

3.3 Complexity and Inference Caching

For a module processing T valid tokens, let $D = \sum_{\ell=1}^{L} d_{\ell-1} d_\ell$ denote the total number of weight parameters across the L internal layers of a single composed path. Composing all K blocks from E experts costs $O(KED)$, and executing the k selected blocks for all tokens costs $O(TkD)$. Ignoring lower-order routing operations and the constant factor from the two DPRG paths, the total uncached composition-and-routed-execution cost is $O(KED + TkD)$. The corresponding amortized per-token cost is $O(KED/T + kD)$. The composition term remains linear in E, but it is incurred once for K reusable blocks rather than once per token.

Block composition depends only on the learned block embeddings, hypernetwork, and expert parameters. Once the model parameters are fixed for inference, all composed block parameters can be constructed once and cached, removing the $O(KED)$ composition term from request-time computation. The dominant routed-block cost is then $O(TkD)$. Once the blocks are cached, inference no longer accesses the original expert pool, so increasing E alone does not increase request-time computation.

4 Visual Experiments

4.1 Experimental Setup

ImageNet-1K (Russakovsky et al., 2015) is our primary benchmark. It contains 1.28 million training images and 50,000 validation images from 1,000 classes. All methods use an eight-layer DeiT-Tiny-style backbone (Touvron et al., 2021) and are trained from scratch on the official training split. We report the mean and sample standard deviation of validation ACC@1 and ACC@5 over three random seeds. Individual runs are reported in Appendix E. We compare RecipeMoE with the dense backbone, sparse-routing MoEs, and dense-participation MoEs, with complete experimental configurations and implementation details provided in Appendix A.

4.2 Experimental Results

Main results. As shown in Table 1, RecipeMoE achieves 73.78% ACC@1 and 91.47% ACC@5. It outperforms the strongest baseline for each metric: SMEAR by 1.95 percentage points in ACC@1

Table 1: Main results on ImageNet-1K. ↑ indicates that higher values are better. Accuracy (%) is reported as mean with sample standard deviation in subscript over three seeds. Best results are bold and second-best results are underlined. Params denotes the total number of trainable parameters in the model. Activated denotes the number of trainable parameters activated as a single patch token passes through the entire model. FLOPs denotes the inference computational cost per image, measured with a batch size of 1.

Method	ACC@1 (%)↑	ACC@5 (%)↑	Params (M)	Activated (M)	FLOPs (G)
Dense	$66.47_{\pm 0.26}$	$87.64_{\pm 0.11}$	3.938	3.938	1.453
Switch Transformer	$70.23_{\pm 0.13}$	$89.49_{\pm 0.18}$	24.070	3.951	1.458
DeepSeek-V3 MoE	$71.67_{\pm 0.36}$	$90.30_{\pm 0.28}$	24.004	6.306	2.382
ReMoE	$70.17_{\pm 0.34}$	$89.70_{\pm 0.23}$	24.070	4.574	1.458
V-MoE	$71.79_{\pm 0.24}$	$\underline{90.48}_{\pm 0.20}$	24.070	5.135	1.920
Expert Choice	$71.57_{\pm 0.08}$	$90.27_{\pm 0.05}$	24.070	5.135	1.958
DynMoE	$68.79_{\pm 0.68}$	$89.20_{\pm 0.31}$	24.070	10.306	4.170
MASS	$70.53_{\pm 0.57}$	$89.76_{\pm 0.25}$	24.070	8.827	3.601
Soft MoE	$71.12_{\pm 0.14}$	$89.92_{\pm 0.10}$	24.201	23.017	1.656
SMEAR	$\underline{71.83}_{\pm 0.35}$	$90.30_{\pm 0.16}$	24.070	22.887	1.493
Lory	$69.61_{\pm 0.36}$	$88.80_{\pm 0.47}$	24.005	22.822	2.003
μMoE (CP)	$70.17_{\pm 0.46}$	$89.55_{\pm 0.19}$	24.070	22.887	9.120
μMoE (TR)	$69.76_{\pm 0.45}$	$89.34_{\pm 0.20}$	24.065	22.882	9.151
RecipeMoE	$\mathbf{73.78}_{\pm 0.09}$	$\mathbf{91.47}_{\pm 0.15}$	24.295	23.111	4.063
RecipeMoE (cached)	–	–	–	–	3.457

Table 2: Component ablations on ImageNet-1K. Accuracy (%) is reported as mean with sample standard deviation in subscript over three seeds.

Variant	ACC@1↑	ACC@5↑	Variant	ACC@1↑	ACC@5↑	Variant	ACC@1↑	ACC@5↑
Full	$\mathbf{73.78}_{\pm 0.09}$	$\mathbf{91.47}_{\pm 0.15}$	Fixed λ	$72.97_{\pm 0.19}$	$91.14_{\pm 0.09}$	w/o Shared	$73.09_{\pm 0.17}$	$91.00_{\pm 0.05}$
1-Layer	$69.13_{\pm 0.36}$	$89.01_{\pm 0.27}$	w/o Gate	$67.63_{\pm 0.12}$	$88.13_{\pm 0.06}$	w/o Filter	$73.46_{\pm 0.30}$	$91.31_{\pm 0.23}$
Direct Blocks	$73.15_{\pm 0.11}$	$91.05_{\pm 0.16}$	Tied Recipes	$72.46_{\pm 0.31}$	$90.84_{\pm 0.19}$	Softmax Coeff.	$73.30_{\pm 0.31}$	$91.22_{\pm 0.26}$

and V-MoE by 0.99 points in ACC@5. The corresponding gains over the dense backbone are 7.32 and 3.83 percentage points. RecipeMoE outperforms all evaluated baselines on both metrics in each of the three seeds. All MoE models have approximately 24M trainable parameters. RecipeMoE's 23.111M activated parameters reflect full expert participation in block composition. Execution remains sparse, with inference costs of 4.063 GFLOPs without caching and 3.457 GFLOPs with caching at batch size 1. At a similar computational scale, its ACC@1 exceeds DynMoE (4.170 GFLOPs) by 4.99 percentage points.

Ablation studies. To assess the contribution of each component, Table 2 reports ablations on ImageNet-1K. The *1-Layer* variant replaces each two-layer block with a single layer and enlarges its expert pool to preserve the total number of expert parameters. *Direct Blocks* removes the hypernetwork and expert pools and directly trains the value and gate parameters of K blocks, keeping the remaining settings unchanged to isolate the effect of parameter synthesis. *Fixed* λ makes λ nonlearnable and sets it to 1. *w/o Gate* removes the composed gate path and multiplicative modulation, reducing equation 10 to $\widetilde{\mathbf{z}}_{t,b}^{(\ell)} = \mathbf{v}_{t,b}^{(\ell)}$. *Tied Recipes* uses one coefficient head for both paths, $\boldsymbol{\alpha}_b^v = \boldsymbol{\alpha}_b^g$. *w/o Shared* removes the shared SwiGLU expert, while *w/o Filter* bypasses block-conditioned feature filtering by setting $\mathbf{z}_{t,b}^{(0)} = \mathbf{x}_t$. *Softmax Coeff.* replaces our unconstrained and scaled coefficient formulation in equation 4 with separately softmax-normalized value and gate coefficients.

Removing the gate path causes the largest performance drop, followed by collapsing each two-layer block into a parameter-matched single layer. The full model outperforms Direct Blocks and

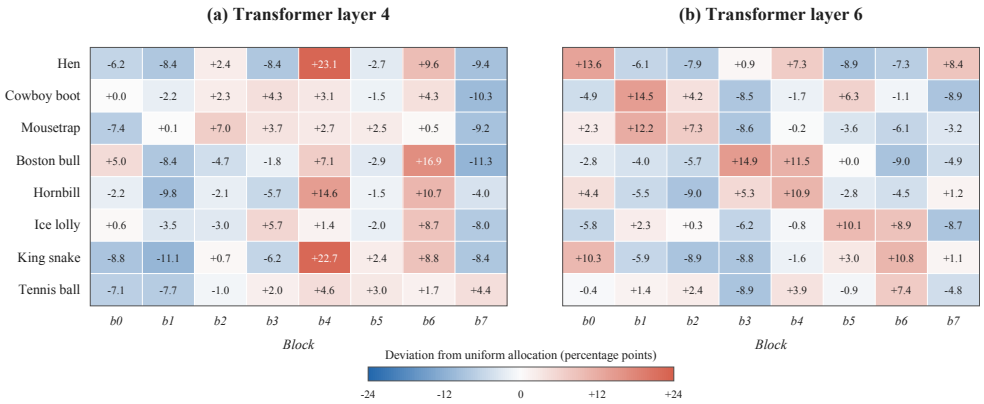

Figure 3: Block routing patterns across classes and layers on ImageNet-1K. Each cell shows the token-assignment proportion for a class and block minus the uniform allocation of 12.5%, in percentage points. Positive (red) and negative (blue) values indicate above- and below-uniform use of a block, respectively. Each Top-k selection is counted as one token–block assignment.

Tied Recipes by 0.63 and 1.32 percentage points in ACC@1, respectively, supporting parameter synthesis and independent value/gate recipes. The shared expert and block-conditioned feature filter both improve mean accuracy. Learning λ and using unconstrained composition coefficients also outperform fixing λ and softmax-normalizing the coefficients, respectively. The full model achieves higher ACC@1 than every ablation variant across all three seeds (Appendix E).

Sensitivity to architectural hyperparameters. On ImageNet-1K, increasing the codebook size K or expert-pool size E yields diminishing gains at larger values. Selecting more blocks improves accuracy, while $k = 2$ balances accuracy and per-token computation. A depth of $L = 2$ performs best in our evaluation. Full results are provided in Appendix B.

4.3 Analysis

Routing analysis. On ImageNet-1K, we compute the proportion of patch-token routing assignments received by each block within each ground-truth class. Figure 3 shows deviations from uniform routing across the eight blocks (12.5% per block) for the same eight representative classes at layers 4 and 6. Different image classes favor different blocks within the same layer, and these preferences often change with depth. For example, at layer 4, hen favors b_4, whereas Boston bull favors b_6. At layer 6, their preferred blocks shift to b_0 and b_3, respectively. These patterns show that routing is both class-conditioned and depth-dependent.

Composition analysis. Figure 4 shows value- and gate-path composition coefficients on ImageNet-1K. Within each layer, blocks assign different positive and negative weights to the same ordered experts, demonstrating distinct composition recipes without coefficient collapse. The patterns also differ across layers and between the two DPRG paths. Recipe differentiation increases with depth. At layers 0, 2, 4, and 6, mean pairwise cosine similarity between block recipes decreases from 0.796 to 0.079, 0.019, and 0.010 for the value path, and from 0.726 to 0.126, 0.083, and 0.043 for the gate path. The shallowest layer thus largely shares a composition pattern across blocks, while deeper layers learn more diverse recipes.

Inference cost and caching. Figure 5 shows how the inference memory and FLOPs of cached and uncached RecipeMoE change as the number of experts increases in the ImageNet-1K setup. Without caching, peak inference memory increases from 54.57 to 627.83 MB and inference cost increases from 3.495 to 8.305 GFLOPs as the expert-pool size E grows from 1 to 128. With caching, they remain constant at 104.33 MB and 3.457 GFLOPs, respectively. Caching introduces a fixed memory cost for the precomposed blocks, so it uses more memory for small expert pools, but becomes more memory-efficient from 16 experts onward. These results show that caching decouples request-time

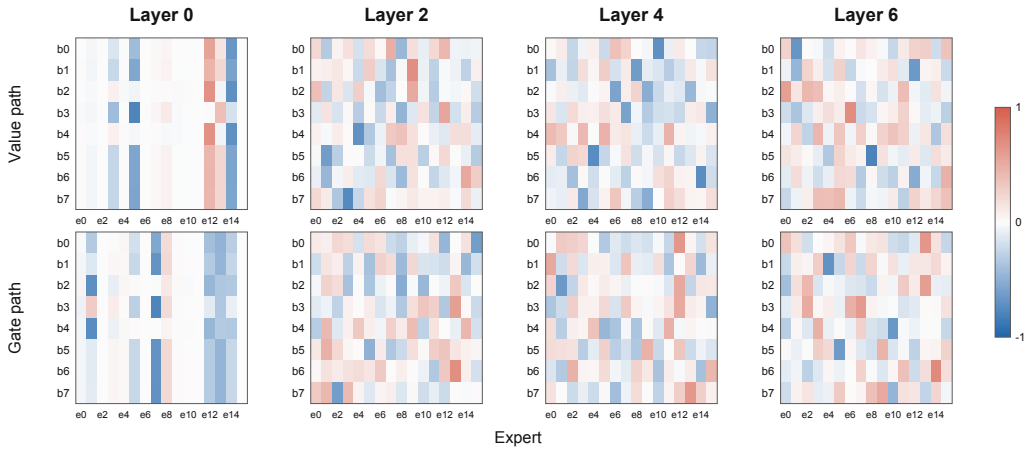

Figure 4: Expert-composition coefficients across blocks and layers on ImageNet-1K. Within each heatmap, rows represent blocks and columns represent experts. Each row is ℓ_2-normalized for visualization only. Red and blue indicate positive and negative coefficients, respectively.

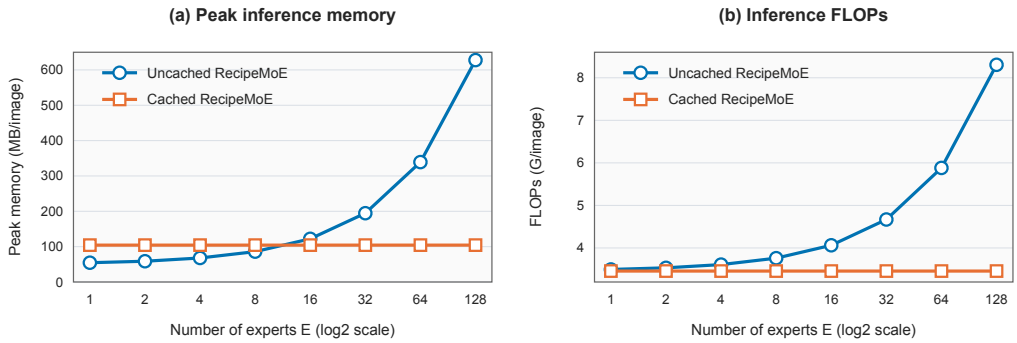

Figure 5: Peak inference memory and inference FLOPs versus expert count E for cached and uncached RecipeMoE using the ImageNet-1K configuration (FP32, batch size 1). Cached models discard the original expert pools. Peak memory excludes one-time cache construction. The horizontal axis uses a $\log_2$ scale.

memory and computation from the size of the expert pool. During training, all images in a minibatch share the same composed blocks. The composition is therefore performed once per batch, and its cost is amortized across the images in that batch.

Expert contributions. Removing any examined expert reduces ImageNet-1K ACC@1, indicating that RecipeMoE makes effective use of the full expert pool. The earliest MoE layer relies more heavily on a few experts, whereas later layers distribute useful contributions more evenly. Full results are provided in Appendix C.

5 Generalization to Language Modeling and Recommendation

Tasks and datasets. We use MiniPile (Kaddour, 2023) and IntTravel (Yan et al., 2026) to test whether the same architecture generalizes to language modeling and sequential recommendation. MiniPile uses an 18-layer Llama-style causal Transformer (Touvron et al., 2023), while IntTravel uses a causal Transformer. MiniPile is a 6-GB subset of the deduplicated Pile. We evaluate causal language modeling on MiniPile using token-level cross-entropy and perplexity. IntTravel contains 162.8 million users, 7.3 million POIs, and 4.13 billion interactions. We use its "Where" task to predict the destination POI of the next journey and report HR@1, HR@5, and NDCG@5.

Table 3: Generalization results on MiniPile and IntTravel. ↑ indicates that higher values are better, and ↓ indicates that lower values are better. Results are means with sample standard deviations shown as subscripts over three seeds. Best results per column are bold and second-best results are underlined.

	MiniPile		IntTravel		
Method	Loss↓	PPL↓	HR@1 (%)↑	HR@5 (%)↑	NDCG@5 (%)↑
Dense	$2.8168_{\pm 0.0036}$	$16.7230_{\pm 0.0607}$	$67.50_{\pm 0.02}$	$86.57_{\pm 0.02}$	$77.87_{\pm 0.02}$
Switch Transformer	$2.7164_{\pm 0.0019}$	$15.1265_{\pm 0.0292}$	$68.01_{\pm 0.10}$	$86.66_{\pm 0.04}$	$78.15_{\pm 0.08}$
DeepSeek-V3 MoE	$2.7078_{\pm 0.0066}$	$14.9968_{\pm 0.0993}$	$68.13_{\pm 0.07}$	$86.77_{\pm 0.03}$	$78.24_{\pm 0.05}$
ReMoE	$2.7346_{\pm 0.0010}$	$15.4034_{\pm 0.0152}$	$68.33_{\pm 0.01}$	$\underline{86.85}_{\pm 0.01}$	$78.36_{\pm 0.01}$
DynMoE	$2.7115_{\pm 0.0084}$	$15.0520_{\pm 0.1268}$	$68.16_{\pm 0.04}$	$86.79_{\pm 0.03}$	$78.26_{\pm 0.01}$
MASS	$2.7513_{\pm 0.0100}$	$15.6636_{\pm 0.1559}$	$68.22_{\pm 0.04}$	$86.81_{\pm 0.02}$	$78.30_{\pm 0.02}$
Soft MoE (causal)	$2.7129_{\pm 0.0035}$	$15.0723_{\pm 0.0534}$	$68.29_{\pm 0.04}$	$86.81_{\pm 0.02}$	$78.33_{\pm 0.02}$
SMEAR (token-level)	$2.7133_{\pm 0.0029}$	$15.0789_{\pm 0.0442}$	$68.27_{\pm 0.03}$	$86.79_{\pm 0.03}$	$78.31_{\pm 0.04}$
Lory	$2.7595_{\pm 0.0086}$	$15.7925_{\pm 0.1367}$	$68.12_{\pm 0.01}$	$86.73_{\pm 0.04}$	$78.21_{\pm 0.02}$
μMoE (CP)	$\underline{2.7075}_{\pm 0.0024}$	$\underline{14.9925}_{\pm 0.0366}$	$68.35_{\pm 0.03}$	$86.84_{\pm 0.02}$	$78.37_{\pm 0.02}$
μMoE (TR)	$2.7186_{\pm 0.0021}$	$15.1590_{\pm 0.0311}$	$\underline{68.38}_{\pm 0.02}$	$\underline{86.85}_{\pm 0.01}$	$\underline{78.38}_{\pm 0.03}$
RecipeMoE	$\mathbf{2.6789}_{\pm 0.0011}$	$\mathbf{14.5696}_{\pm 0.0161}$	$\mathbf{68.52}_{\pm 0.01}$	$\mathbf{86.92}_{\pm 0.01}$	$\mathbf{78.50}_{\pm 0.03}$

Experimental results. Table 3 compares RecipeMoE with the dense backbone, sparse-routing methods, and dense-participation methods on MiniPile and IntTravel. Results are reported as means with sample standard deviations across three seeds. Complete model, training, routing, and baseline configurations are provided in Appendix A. On MiniPile, RecipeMoE reduces mean PPL by 2.8% relative to the strongest baseline, μMoE (CP), and by 12.9% relative to the dense backbone. On IntTravel, RecipeMoE consistently outperforms all available baselines across the three seeds on HR@1, HR@5, and NDCG@5. Per-seed results for both tasks and a detailed analysis of IntTravel are provided in Appendix E. Component ablations on MiniPile and IntTravel support the effectiveness of the main design choices beyond vision. Detailed results and analysis are provided in Appendix D.

Online evaluation and deployment. We evaluate cached RecipeMoE in an industrial generative POI recommendation service within a map app. The model predicts the Top-10 POIs a user is likely to visit, which determine the app's initial map viewport and zoom level. The service requires a response within $60\,\mathrm{ms}$. A one-week online A/B experiment compared the existing production model without an MoE module against cached RecipeMoE. The experiment handled approximately 5,000 queries per second. Running on PPUs, cached RecipeMoE achieved $19\,\mathrm{ms}$ average and $38\,\mathrm{ms}$ P99 latency, with a 2.4% relative UVCTR improvement. It was subsequently fully deployed to serve production traffic.

6 Conclusion

We presented RecipeMoE, which decouples full expert participation from sparse token-level execution. Learned recipes compose a finite set of reusable blocks from the entire expert pool, and each token executes only a few blocks selected by Top-k routing. RecipeMoE further introduces DPRG, which uses independent value and gate recipes to compose the same expert pool, enabling nonlinear interactions through multiplicative gating. RecipeMoE achieves higher ACC@1 and ACC@5 than all evaluated sparse-routing and dense-participation MoE baselines on ImageNet-1K. Expert-removal and routing analyses show effective use of the expert pool and class- and depth-dependent block specialization. Results on MiniPile and IntTravel support its applicability to language modeling and sequential recommendation. Caching removes block synthesis from request-time computation, and deployment in a large-scale generative recommendation system achieves low latency and improves online engagement.

AI USE STATEMENT

We have not used generative AI tools for formulating the core research idea or designing the RecipeMoE architecture. We used generative AI tools to assist with code implementation, including code generation, debugging, and optimization. We also used these tools for translation, language editing, and grammar checking. They further assisted with qualitative analysis, interpretation of experimental results and figures, and summarization of conclusions. We also used AI assistance for LaTeX formatting and adjustments to figure-generation scripts. AI-assisted analyses and written claims were checked against the manuscript and available experimental data, and LaTeX compilation and visual inspection were used to verify formatting and figure presentation. We have reviewed all AI-assisted work and take responsibility for the final content of this paper, including its technical claims, experimental results, and AI-assisted code, text, and artifacts.

ETHICS STATEMENT

This work studies a general-purpose model architecture and evaluates its use in visual recognition, language modeling, and POI recommendation. User behavior and location-related data in recommendation raise privacy concerns. The paper reports aggregate evaluation results and does not disclose individual user records or trajectories. Our use of the datasets and deployment of the model comply with applicable data-access restrictions and privacy requirements. Recommendation systems may reinforce exposure biases. The observed UVCTR improvement does not establish fairness across users or POIs.

REPRODUCIBILITY STATEMENT

The source code of RecipeMoE is publicly available at `https://anonymous.4open.science/r/RecipeMoE-717C254/`. Appendix A provides the model configurations, optimization settings, data preprocessing, random seeds, and hardware used in our experiments. The method section specifies parameter composition, routing, and block execution, while the supplementary experiments report architectural sensitivity and component ablations. These details support reproduction of the offline evaluations. Online A/B results depend on the production environment and live traffic.

REFERENCES

Xiaodong Chen, Mingming Ha, Zhenzhong Lan, Jing Zhang, and Jianguo Li. Mobe: Mixture-of-basis-experts for compressing moe-based llms. In *International Conference on Learning Representations*, volume 2026, pp. 40010–40028, 2026.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1280–1297, 2024.

Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment. *arXiv preprint arXiv:2502.18965*, 2025.

Zhiwen Fan, Rishov Sarkar, Ziyu Jiang, Tianlong Chen, Kai Zou, Yu Cheng, Cong Hao, Zhangyang Wang, et al. M^3vit: Mixture-of-experts vision transformer for efficient multi-task learning with model-accelerator co-design. *Advances in Neural Information Processing Systems*, 35:28441–28457, 2022.

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.

Yongxin Guo, Zhenglin Cheng, Xiaoying Tang, Zhaopeng Tu, and Tao Lin. Dynamic mixture of experts: An auto-tuning approach for efficient transformer models. In *International Conference on Learning Representations*, volume 2025, pp. 79643–79672, 2025.

David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks, 2016. URL https://arxiv.org/abs/1609.09106.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Jean Kaddour. The minipile challenge for data-efficient language models. *arXiv preprint arXiv:2304.08442*, 2023.

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.

Jiachen Li, Xinyao Wang, Sijie Zhu, Chia-Wen Kuo, Lu Xu, Fan Chen, Jitesh Jain, Humphrey Shi, and Longyin Wen. Cumo: Scaling multimodal llm with co-upcycled mixture-of-experts. *Advances in Neural Information Processing Systems*, 37:131224–131246, 2024.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1930–1939, 2018.

Mohammed Muqeeth, Haokun Liu, and Colin Raffel. Soft merging of experts with adaptive routing. *arXiv preprint arXiv:2306.03745*, 2023.

Basil Mustafa, Carlos Riquelme, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. Multimodal contrastive learning with limoe: the language-image mixture of experts. *Advances in Neural Information Processing Systems*, 35:9564–9576, 2022.

James Oldfield, Markos Georgopoulos, Grigorios G Chrysos, Christos Tzelepis, Yannis Panagakis, Mihalis A Nicolaou, Jiankang Deng, and Ioannis Patras. Multilinear mixture of experts: Scalable expert specialization through factorization. *Advances in Neural Information Processing Systems*, 37:53022–53063, 2024.

Sumin Park and Noseong Park. How many experts are enough? towards optimal semantic specialization for mixture-of-experts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 24792–24800, 2026.

Joan Puigcerver, Carlos Riquelme Ruiz, Basil Mustafa, and Neil Houlsby. From sparse to soft mixtures of experts. In *International Conference on Learning Representations*, volume 2024, pp. 28435–28445, 2024.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in neural information processing systems*, 34:8583–8595, 2021.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

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.

Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Proceedings of the 14th ACM conference on recommender systems*, pp. 269–278, 2020.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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.

Zeyue Xue, Guanglu Song, Qiushan Guo, Boxiao Liu, Zhuofan Zong, Yu Liu, and Ping Luo. Raphael: Text-to-image generation via large mixture of diffusion paths. *Advances in Neural Information Processing Systems*, 36:41693–41706, 2023.

Huimin Yan, Longfei Xu, Junjie Sun, Zheng Liu, Wei Luo, Kaikui Liu, and Xiangxiang Chu. Inttravel: A real-world dataset and generative framework for integrated multi-task travel recommendation. *arXiv preprint arXiv:2602.11664*, 2026.

Jiahao Yu, Yihai Duan, Longfei Xu, Chao Chen, Shuliang Liu, Kaikui Liu, Fan Yang, Xiangxiang Chu, and Ning Guo. Dsfnet: Learning disentangled scenario factorization for multi-scenario route ranking. In *Companion Proceedings of the ACM on Web Conference 2025*, pp. 567–576, 2025.

Mikołaj Zasada, Łukasz Struski, Jacek Tabor, and Marcin Kurdziel. Softmoe: Soft differentiable routing for mixture-of-experts in llms. *arXiv preprint arXiv:2606.17952*, 2026.

Hao Zhao, Zihan Qiu, Huijia Wu, Zili Wang, Zhaofeng He, and Jie Fu. Hypermoe: Towards better mixture of experts via transferring among experts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10605–10618, 2024.

Youwei Zheng, Yuxi Ren, Xin Xia, Xuefeng Xiao, and Xiaohua Xie. Dense2moe: Restructuring diffusion transformer to moe for efficient text-to-image generation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18661–18670. IEEE, 2025.

Zexuan Zhong, Mengzhou Xia, Danqi Chen, and Mike Lewis. Lory: Fully differentiable mixture-of-experts for autoregressive language model pre-training. *arXiv preprint arXiv:2405.03133*, 2024.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

Jie Zhu, Zhifang Fan, Xiaoxie Zhu, Yuchen Jiang, Hangyu Wang, Xintian Han, Haoran Ding, Xinmin Wang, Wenlin Zhao, Zhen Gong, et al. Rankmixer: Scaling up ranking models in industrial recommenders. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 6309–6316, 2025.

Table 4: Dataset-specific model and training configurations.

Configuration	ImageNet-1K	MiniPile	IntTravel
Layers H	8	18	8
Hidden size d	192	768	96
Attention $n_{\text{heads}} \times d_{\text{head}}$	3×64	12×64	1×96
Router hidden size d_r	64	64	64
Experts E	16	8	32
Initial residual scale λ_0	1	1	0
Batch size per device	256	8	64
Input size	224^2 ($p = 16$)	1024	126
Training length	200 epochs	1 epoch	610K steps

A Detailed Experimental Configurations

A.1 Common Experimental Protocol

All experiments use eight PPUs, each with 96 GB of memory. Within each dataset, all methods share the backbone, input processing, task head (except on IntTravel, where the prediction head also uses the corresponding MoE module), optimization schedule, seed set, and expert initialization; only the FFN or MoE module differs. For the main comparisons, we randomly select three seeds for each dataset. ImageNet-1K uses seeds 214797, 531770, and 635451. MiniPile uses seeds 1, 145306, and 145849. IntTravel uses seeds 42, 4981213, and 211231244.

Table 4 summarizes the model and training configurations. E denotes the number of experts per internal layer in RecipeMoE, excluding the shared expert. Baseline expert counts follow the parameter-matching rules in Appendix A.3. Input size denotes image resolution or maximum sequence length, and p denotes patch size.

A.2 Dataset-Specific Configurations

ImageNet-1K. We use a DeiT-Tiny-style backbone (Touvron et al., 2021) with eight Transformer layers. MoE modules replace the FFNs in alternating layers (0, 2, 4, and 6) for patch tokens, while the class token uses a separate dense FFN. We use AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$ and weight decay 0.05. Router parameters, biases, and normalization parameters are exempt from weight decay. The base learning rate is 5×10^{-4} at a reference global batch size of 512 and scales linearly with the global batch size. With 256 images per device across eight devices, the peak learning rate is 2×10^{-3}. The learning rate warms up from 10^{-6} over five epochs and then follows cosine decay to 10^{-5}. Standard dropout, attention dropout, patch dropout, and position dropout are disabled, while stochastic depth uses a maximum DropPath rate of 0.1. Training images undergo a random resized crop to 224×224 using bicubic interpolation, with scale $[0.08, 1.0]$ and aspect ratio $[0.75, 1.33]$. We apply random horizontal flipping with probability 0.5 and RandAugment with two operations per image, magnitude 9, and magnitude standard deviation 0.5. Pixel-mode random erasing is applied with probability 0.25, using one erased region. Each batch uses either Mixup ($\alpha = 0.8$) or CutMix ($\alpha = 1.0$) with equal probability, with label smoothing of 0.1. At evaluation, images are resized with bicubic interpolation to a shorter side of 256 pixels, center-cropped to 224×224, and normalized using ImageNet mean and standard deviation.

MiniPile. We use a Llama-style causal Transformer (Touvron et al., 2023) and replace the FFN in every layer with an MoE module. We train for one epoch on 1.523 billion tokens and evaluate on a 15.0-million-token test split. Text is tokenized with a byte-level BPE vocabulary of 32,000. Within each split, documents are separated by EOS tokens and concatenated into sequences of 1,024 tokens, with the final incomplete sequence discarded. Accumulating gradients over four microbatches of eight sequences per device gives an effective batch size of 256 sequences across eight devices. We

use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$ and weight decay 0.1. The learning rate warms up to 4×10^{-4} over the first 10% of optimizer steps and then decays cosinusoidally to 4×10^{-5}.

IntTravel. We follow the original IntTravel experimental protocol (Yan et al., 2026) for dataset splits, candidate construction with negative sampling, and evaluation. We use a causal Transformer for sequential recommendation and replace every FFN and the prediction head with MoE modules. We use AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$, weight decay 10^{-6}, and a constant learning rate of 10^{-4}.

A.3 MoE Configurations and Fair Comparison

Expert-parameter matching. We use the parameter count of the backbone's original Transformer FFN, which is replaced by the MoE module, as the target for a single expert. Each internal RecipeMoE layer retains a separate pool of E experts, and each expert contains one weight matrix. For parameter accounting, we pair same-index experts from the two internal layers and match their combined parameter count to this target. For the baselines, one complete FFN or SwiGLU expert is matched to the same target by counting all of its weight matrices. This matching sets the input–intermediate–output dimensions of each two-layer RecipeMoE block to $d \to 2844 \to d$ on MiniPile and $d \to 4d \to d$ on ImageNet-1K and IntTravel. We also match the always-active shared SwiGLU expert to the same parameter target by adjusting its intermediate width. To keep the total expert parameter count comparable across methods, a baseline with a shared path uses E routed experts and one shared expert, whereas a baseline without one uses $E+1$ routed experts. For the CP and Tensor Ring variants of μMoE, we choose the factorization ranks that make the total trainable parameter count of the full model as close as possible to that of RecipeMoE. Specifically, we solve the CP rank R under this parameter budget; for Tensor Ring, we fix $R_1 = R_2 = 4$ and solve R_3.

Baseline configurations. Switch Transformer follows its original Top-1 routing rule. Other Top-k token-choice baselines use Top-2 routing when their original configurations do not specify k. Expert Choice retains expert-to-token routing with a capacity factor of 2. μMoE uses $\mathrm{entmax}_{1.5}$ routing, and MASS uses Top-p routing with $p = 0.5$.

On ImageNet-1K, Soft MoE uses as many slots as patch tokens. On MiniPile and IntTravel, we make Soft MoE causal by constructing each slot only from the current and preceding tokens. We use one slot per expert to limit computation. SMEAR uses example-level composition for images and token-level composition for autoregressive tasks. Lory uses a segment size of 16.

For DynMoE and MASS, the initial expert-pool sizes on IntTravel, ImageNet-1K, and MiniPile are 16, 8, and 4, respectively. Their maximum pool sizes match the expert counts used by the other methods. We create all expert parameters in advance up to these maxima. DynMoE initializes each expert's threshold to correspond to a selection probability of 0.5 and uses a routing temperature of 1.0. Every 300 optimizer steps, it removes unused experts and opens a new expert when some tokens activate no expert. MASS detects semantic drift over the previous 200 optimizer steps and expands the pool using significance and similarity thresholds of 0.01 and 0.001. Its redundancy regularizer has a coefficient of 0.01.

RecipeMoE configuration. Each module contains $K = 8$ codebook blocks and routes each token to $k = 2$ blocks. The block-embedding dimension, router hidden dimension, and hypernetwork hidden dimension are 32, 64, and 16, respectively. Each block contains two internal DPRG layers. These layers use equally sized, layer-specific expert pools and share the same pair of value and gate composition coefficients. An always-active SwiGLU expert provides the shared path. We initialize the expert matrices with PyTorch's default `nn.Linear` scheme and set all expert biases to zero. This small initial weight scale limits perturbations from the composed transformations early in training.

B Sensitivity to Architectural Hyperparameters

We vary the number of candidate blocks K, the number of selected blocks k, the expert-pool size E at each internal layer, and the block depth L on ImageNet-1K. We change one hyperparameter

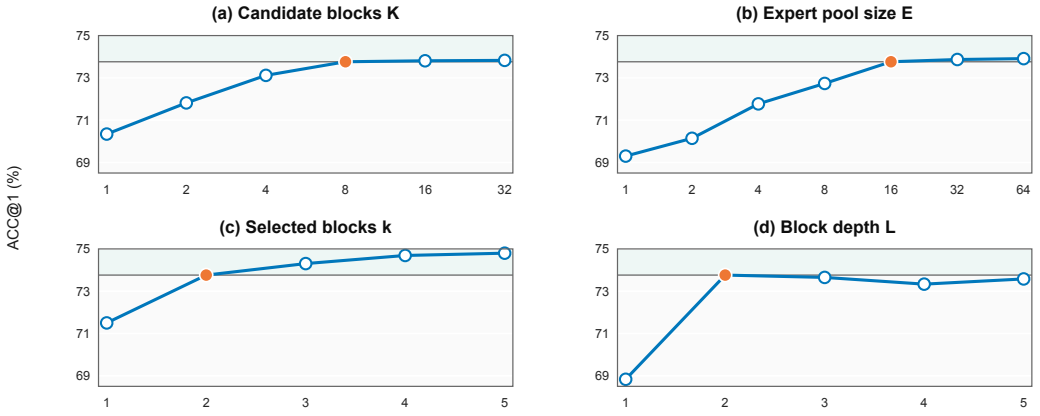

Figure 6: Sensitivity of RecipeMoE to architectural hyperparameters on ImageNet-1K. Each panel reports ACC@1 while varying one hyperparameter. Results are from single runs. Higher values are better. All panels use the same vertical scale. Orange points indicate the default configuration $(K, E, k, L) = (8, 16, 2, 2)$.

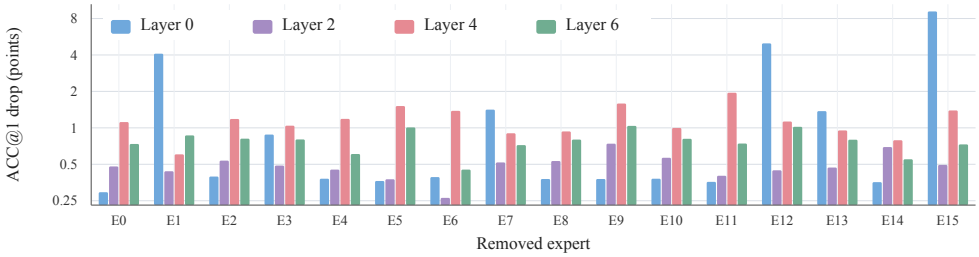

Figure 7: ACC@1 drop on ImageNet-1K after removing individual experts from one MoE layer at a time. All results are evaluated using a single trained checkpoint. The vertical axis uses a logarithmic scale.

at a time from the default configuration $(K, E, k, L) = (8, 16, 2, 2)$. For the $K = 1$ setting, we set $k = 1$. Figure 6 reports the resulting ACC@1 values.

Increasing either K or E improves accuracy, but further gains are small beyond their default values. Expanding K from 8 to 32 increases ACC@1 accuracy by only 0.07 percentage points, and expanding E from 16 to 64 increases it by only 0.15 percentage points. Increasing k consistently improves ACC@1. We use $k = 2$ to obtain a substantial gain over $k = 1$ while limiting per-token computation. For block depth, $L = 2$ achieves the highest ACC@1.

C Expert Contributions

Although every block is constructed from all experts, this design alone does not show that every expert makes a meaningful contribution to model performance. We therefore evaluate the contribution of each expert on ImageNet-1K by removing it from all block compositions in one MoE layer at a time. We examine all 16 experts in Layers 0, 2, 4, and 6, producing 64 removal settings in total, and evaluate the original checkpoint without retraining. After removing an expert, we replace $1/\sqrt{E}$ with $1/\sqrt{E-1}$ as an approximate scale adjustment for the reduced expert count.

Figure 7 shows that removing any of the examined experts reduces ACC@1. The mean drops at Layers 0, 2, 4, and 6 are 1.60, 0.49, 1.17, and 0.78 percentage points, respectively. Contributions are highly uneven in Layer 0, where removing E1, E12, or E15 lowers accuracy by 4.10, 4.99, and 9.15 percentage points, while the effects of most other experts are much smaller. The removal effects

Table 5: Component ablations on MiniPile and IntTravel. Results are from single runs.

	MiniPile		IntTravel		
Variant	Loss↓	PPL↓	HR@1 (%)↑	HR@5 (%)↑	NDCG@5 (%)↑
RecipeMoE	2.6802	14.5878	68.52	86.92	78.50
1-Layer	2.7174	15.1410	68.24	86.80	78.28
Fixed λ	2.6938	14.7871	68.44	86.85	78.27
w/o Gate	2.7660	15.8947	68.20	86.76	78.28
Tied Recipes	2.7138	15.0861	68.41	86.87	78.44
w/o Shared	2.7281	15.3035	68.46	86.89	78.44
w/o Filter	2.6829	14.6277	68.47	86.87	78.46
Softmax Coeff.	2.7071	14.9864	68.35	86.86	78.39

Table 6: Per-seed accuracy on ImageNet-1K (%). Values are rounded to three decimal places. Best results in each column are bold and second-best results are underlined.

	Seed 214797		Seed 531770		Seed 635451	
Method	ACC@1↑	ACC@5↑	ACC@1↑	ACC@5↑	ACC@1↑	ACC@5↑
Dense	66.246	87.582	66.412	87.566	66.748	87.758
Switch Transformer	70.120	89.292	70.374	89.636	70.184	89.550
DeepSeek-V3 MoE	71.366	90.060	71.568	90.246	<u>72.064</u>	<u>90.604</u>
ReMoE	69.786	89.444	70.272	89.774	70.450	89.888
V-MoE	71.892	<u>90.568</u>	71.970	<u>90.620</u>	71.514	90.244
Expert Choice	71.550	90.328	71.500	90.230	71.650	90.248
DynMoE	68.356	89.036	68.446	89.012	69.582	89.556
MASS	71.178	90.050	70.312	89.652	70.102	89.592
Soft MoE	71.270	89.882	71.110	89.838	70.994	90.036
SMEAR	<u>71.922</u>	90.288	<u>72.128</u>	90.470	71.450	90.144
Lory	69.366	88.852	70.018	89.238	69.434	88.300
μMoE (CP)	70.300	89.662	69.660	89.330	70.562	89.664
μMoE (TR)	69.332	89.148	70.230	89.556	69.724	89.320
RecipeMoE	**73.890**	**91.640**	**73.724**	**91.348**	**73.738**	**91.414**

become more balanced in later layers, ranging from 0.26 to 0.74 percentage points in Layer 2, 0.60 to 1.95 percentage points in Layer 4, and 0.45 to 1.04 percentage points in Layer 6. This suggests that the earliest MoE layer relies strongly on a few experts, whereas later layers distribute useful contributions more evenly across their pools. Even the smallest decrease is 0.26 percentage points, indicating that every expert contributes to the model. Together, the results support the conclusion that the model makes effective use of the full expert pool throughout its MoE layers.

D Component Ablations on Language Modeling and Recommendation

Table 5 reports component ablations on both MiniPile and IntTravel. We evaluate Direct Blocks only on ImageNet-1K, where the parameter budgets are comparable. Removing the gate path causes the largest PPL increase on MiniPile and the largest drops in HR@1 and HR@5 on IntTravel among the evaluated ablations. The parameter-matched 1-Layer variant also underperforms the full model, showing that the two-layer block structure provides benefits beyond expert parameter count. Tied Recipes increases MiniPile PPL from 14.5878 to 15.0861 and reduces IntTravel HR@1, HR@5, and NDCG@5 by 0.11, 0.05, and 0.06 percentage points, respectively, supporting independent value

Table 7: Per-seed results on MiniPile. Values are rounded to four decimal places. Lower values are better; best results in each column are bold and second-best results are underlined.

	Seed 1		Seed 145306		Seed 145849	
Method	Loss↓	PPL↓	Loss↓	PPL↓	Loss↓	PPL↓
Dense	2.8131	16.6621	2.8168	16.7231	2.8204	16.7836
Switch Transformer	2.7156	15.1135	2.7187	15.1600	2.7151	15.1061
DeepSeek-V3 MoE	2.7119	15.0580	<u>2.7002</u>	<u>14.8822</u>	2.7114	15.0500
ReMoE	2.7346	15.4034	2.7356	15.4186	2.7336	15.3882
DynMoE	2.7108	15.0406	2.7035	14.9312	2.7202	15.1840
MASS	2.7608	15.8128	2.7409	15.5017	2.7521	15.6762
Soft MoE (causal)	2.7112	15.0466	2.7169	15.1337	2.7105	15.0367
SMEAR (token-level)	2.7165	15.1277	2.7108	15.0417	2.7125	15.0673
Lory	2.7519	15.6721	2.7689	15.9411	2.7577	15.7643
μMoE (CP)	<u>2.7101</u>	<u>15.0306</u>	2.7073	14.9891	<u>2.7052</u>	<u>14.9576</u>
μMoE (TR)	2.7171	15.1359	2.7178	15.1466	2.7209	15.1943
RecipeMoE	**2.6802**	**14.5878**	**2.6781**	**14.5572**	**2.6785**	**14.5638**

Table 8: Per-seed results on IntTravel (%). Best results in each column are bold and second-best results are underlined.

	Seed 42			Seed 4981213			Seed 211231244		
Method	HR@1 (%)↑	HR@5 (%)↑	NDCG@5 (%)↑	HR@1 (%)↑	HR@5 (%)↑	NDCG@5 (%)↑	HR@1 (%)↑	HR@5 (%)↑	NDCG@5 (%)↑
Dense	67.48	86.59	77.85	67.52	86.56	77.88	67.49	86.56	77.87
Switch Transformer	67.98	86.67	78.12	68.12	86.69	78.24	67.93	86.61	78.10
DeepSeek-V3 MoE	68.05	86.74	78.18	68.19	86.79	78.28	68.16	86.79	78.26
ReMoE	68.33	<u>86.85</u>	78.36	68.34	86.84	78.36	68.33	<u>86.86</u>	78.37
DynMoE	68.18	86.77	78.26	68.19	86.78	78.27	68.12	86.82	78.26
MASS	68.22	86.79	78.29	68.18	86.83	78.29	68.26	86.82	78.32
Soft MoE (causal)	68.32	86.83	78.35	68.31	86.80	78.34	68.25	86.79	78.31
SMEAR (token-level)	68.25	86.76	78.28	68.31	86.81	78.35	68.26	86.81	78.31
Lory	68.11	86.77	78.23	68.13	86.70	78.20	68.11	86.71	78.20
μMoE (CP)	68.35	86.82	<u>78.37</u>	68.33	<u>86.85</u>	78.36	68.38	<u>86.86</u>	78.39
μMoE (TR)	<u>68.37</u>	<u>86.85</u>	78.35	<u>68.36</u>	86.84	<u>78.38</u>	<u>68.40</u>	<u>86.86</u>	<u>78.40</u>
RecipeMoE	**68.52**	**86.92**	**78.50**	**68.53**	**86.92**	**78.53**	**68.52**	**86.91**	**78.47**

and gate recipes across both tasks. The shared expert yields smaller but consistent gains. Fixing λ and removing feature filtering hurt both autoregressive tasks, while the unconstrained and scaled coefficients consistently outperform softmax-normalized composition. These results confirm that the main design choices remain effective in language modeling and sequential recommendation.

E Per-Seed Results

Tables 6, 7, and 8 report the individual runs for RecipeMoE and the baselines on ImageNet-1K, MiniPile, and IntTravel, respectively. These runs are used to compute the means and sample standard deviations in Tables 1 and 3. Table 9 provides the individual ImageNet-1K ablation runs summarized in Table 2.

Table 8 shows that RecipeMoE ranks first on all three metrics for each seed. Relative to the strongest available baseline for each metric and seed, its HR@1 gains are 0.15, 0.17, and 0.12 percentage points for seeds 42, 4981213, and 211231244, respectively. The corresponding gains are 0.07, 0.07, and 0.05 percentage points on HR@5, and 0.13, 0.15, and 0.07 percentage points on NDCG@5.

Table 9: Per-seed component ablations on ImageNet-1K (%).

	Seed 214797		Seed 531770		Seed 635451	
Variant	ACC@1↑	ACC@5↑	ACC@1↑	ACC@5↑	ACC@1↑	ACC@5↑
Full	73.890	91.640	73.724	91.348	73.738	91.414
1-Layer	68.900	88.892	69.544	89.318	68.958	88.830
Direct Blocks	73.022	91.130	73.230	91.158	73.202	90.860
Fixed λ	72.864	91.104	73.186	91.240	72.848	91.064
w/o Gate	67.710	88.184	67.490	88.062	67.680	88.134
Tied Recipes	72.698	90.858	72.106	90.636	72.584	91.020
w/o Shared	73.270	91.034	73.076	90.942	72.926	91.034
w/o Filter	73.134	91.074	73.536	91.536	73.722	91.328
Softmax Coeff.	73.232	91.116	73.032	91.032	73.640	91.516

Thus, the higher mean performance is supported by improvements in every run rather than a single favorable seed.

On ImageNet-1K, the full model outperforms every ablation variant in ACC@1 across all three seeds (Table 9). We examine four ablations with less pronounced effects than removing the gate path or reducing block depth to assess the consistency of their gains across seeds. The ACC@1 gains are 0.620–0.812 percentage points over w/o Shared, 0.016–0.756 over w/o Filter, 0.538–1.026 over Fixed λ, and 0.098–0.692 over Softmax Coeff. ACC@5 improves on average in all four comparisons, though not consistently across seeds for w/o Filter and Softmax Coeff.