TokenMix: Efficient Batched Inference for Mixture-of-Experts Models with Joint Expert Selection

Aradhana Mohan Parvathy[*†] Purdue University
Sarada Krithivasan[*†] IBM Research
Swagath Venkataramani IBM Research
Anand Raghunathan Purdue University
Viji Srinivasan IBM Research

Abstract

The mixture-of-experts (MoE) architecture has proven indispensable for scaling transformer-based artificial intelligence (AI) models. MoEs use a router network to activate specialized subnetworks, called experts, within each transformer layer on a per-token basis. Thus, MoEs increase model capacity without proportionally increasing the computational requirements of training and inference. However, a key challenge is that MoE *inference throughput scales poorly with batching*, as each input in a batch activates a different subset of experts, thereby increasing memory intensity.

To improve batched MoE inference throughput, we propose `TokenMix`, a novel technique that mixes token representations within the router across inputs in a batch to promote expert reuse. In effect, `TokenMix` encourages coordinated expert selection across inputs. `TokenMix` requires neither training nor fine-tuning and can be applied directly to off-the-shelf MoE models. We evaluate `TokenMix` across five state-of-the-art MoE models with parameter counts ranging from 16B to 235B. `TokenMix` achieves a 25–60% improvement in expert reuse, leading to a 12–28% reduction in inter-token latency with negligible or no loss in accuracy.

1 Introduction

Large language models (LLMs) based on the transformer architecture (Vaswani et al., 2017) have demonstrated remarkable performance in many generative and agentic AI applications involving text, code, and images (Grattafiori et al., 2024; OpenAI et al., 2024). The success of these models is largely attributed to their scale—hundreds of billions to trillions of parameters trained on trillions of tokens. However, scaling transformer-based LLMs poses a substantial computational challenge, necessitating new approaches to limit their compute costs.

Recent years have witnessed the advent and widespread adoption of a new class of transformer models called mixture-of-experts (MoE) transformers (Yang et al., 2024, 2025; Dai et al., 2024; Jiang et al., 2024; OpenAI et al., 2025), whose computational requirement for processing a token is decoupled from the overall model size. MoE transformers typically consist of a multitude of experts and a router network within each layer. Each input token activates a small, fixed number of experts that are best suited to process it, as determined by the router. Because expert selection significantly lowers the compute cost per token, MoE LLMs can be efficiently trained at scales considerably larger than conventional (non-MoE) models. The reduced per-token compute cost also offers the potential for superior inference efficiency. However, one key drawback of MoE architectures is that *during inference, inter-token latency and token throughput scale poorly with batching*. We investigate this challenge and address it by proposing techniques to improve batched inference throughput for MoEs.

Batching is an important technique for maximizing LLM inference throughput. Advanced batching methods, such as continuous batching, are routinely employed by AI inference servers (*e.g.,* vLLM (Kwon et al., 2023)) in production. Because LLMs are autoregressive, the token-generation (decode) phase is widely recognized as memory-bandwidth-limited (Kamath et al., 2025), leading to severe underutilization of compute resources. In non-MoE transformer architectures, batching multiple inputs increases arithmetic intensity during token generation because the model weights are reused to process all inputs. This results in a near-linear improvement in throughput with batch size, until it is limited by the size of the KV cache. However, because each token in an MoE model can potentially activate a different set of experts, batching does not guarantee a proportional increase in arithmetic intensity and therefore yields only a

[*]Equal contribution.
[†]Work done while at IBM Research.

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 4128–4140
November 6-10, 2026 ©2026 Association for Computational Linguistics

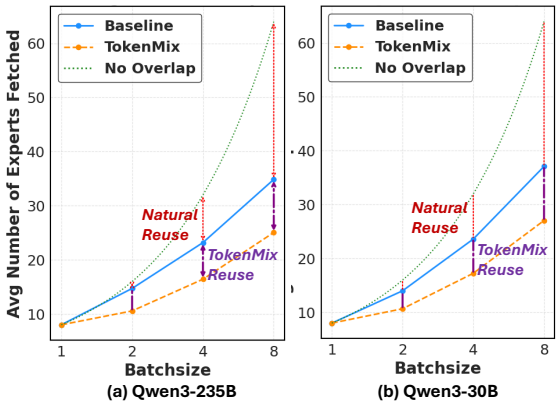


Figure 1: Average number of experts fetched in an MoE layer for Qwen3-235B and Qwen3-30B, both of which select 8 of 128 experts per token, on the GSM8K dataset with (i) no overlap between the experts selected by tokens in a batch (worst-case reuse); (ii) natural (baseline) expert overlap across tokens; and (iii) expert overlap enabled by TokenMix.

modest improvement in throughput. Figure 1(a) illustrates this behavior for Qwen3-235B, in which each token selects 8 of 128 experts. As seen, the number of experts engaged increases considerably with batch size, growing from 8 experts at batch size=1 to ∼35 experts at batch size=8. This increase in the number of experts has a key adverse effect. Engaging more experts proportionally increases the inter-token latency (ITL) [1] when processing the batch, negating the expected benefits from batching. It is therefore imperative to improve expert reuse across a batch of tokens to achieve high-performance inference for MoE models.

Proposed approach: We modify the MoE router network to perform *joint expert selection across a batch*, with the objective of minimizing the number of experts activated during batched inference. To select experts for a given input token, the router computes an affinity score for each expert and selects the top-k experts with the highest scores. Our technique, called TokenMix, introduces carefully crafted perturbations to the representations of different inputs in a batch to increase the overlap in expert choices across inputs. TokenMix promotes expert reuse without affecting generation quality. For example, in Figure 1(a), TokenMix lowers the number of experts engaged from 35 to 25 for batch size=8 in the Qwen3-235B model.

TokenMix incurs negligible runtime and storage overhead because router computations account for a very small fraction of the overall LLM computation. Importantly, TokenMix requires *no training or fine-tuning* of model weights. In our experiments, TokenMix is applied directly to off-the-shelf MoE models using the original model weights for both the experts and the router.

Finally, TokenMix is complementary to prior efforts targeting efficient MoE inference (Bai et al., 2025; Dong et al., 2025; Skliar et al., 2025; He et al., 2025; Lu et al., 2024; Li et al., 2024). While techniques such as expert pruning (Bai et al., 2025; Dong et al., 2025; He et al., 2025; Lu et al., 2024) and expert merging (Li et al., 2024) reduce the number of experts activated for a single input, the challenge of poor scaling with batch size remains. Cache-Conditional MoE (Skliar et al., 2025) proposes a strategy to improve temporal cache hit rates in main memory across consecutive inference executions, which is orthogonal to the challenge we address. To the best of our knowledge, TokenMix is the first approach to leverage joint expert selection across inputs within a batch to improve MoE inference throughput. In Section 5, we study the impact of applying TokenMix in conjunction with prior expert-pruning techniques and Cache-Conditional MoE.

In summary, the paper makes the following contributions:

- We identify a key bottleneck in batched MoE inference: inference throughput scales poorly with batching because different tokens activate different subsets of experts.
- We propose TokenMix, a novel approach that adapts the router to jointly select experts across inputs in a batch, thereby improving expert reuse. TokenMix requires no fine-tuning of model weights.
- We evaluate TokenMix across five state-of-the-art MoE models, with parameter counts ranging from 16B to 235B, and on tasks spanning multiple domains. Across all benchmarks, TokenMix achieves a 25–60% improvement in expert reuse, resulting in a 12–28% reduction in inter-token latency (ITL), with negligible or no loss in accuracy.

2 Background and Motivation

In this section, we provide a brief background on Mixture-of-Experts (MoE) models and outline the

[1] Inter-token latency is the time required to generate a token during decoding, i.e., the elapsed time between the generation of two consecutive tokens during the decode phase of a large language model.

key phases of large language model (LLM) inference along with their deployment characteristics.

2.1 Mixture-of-Experts Models

Mixture-of-Experts (MoE) models replace the feed-forward network (FFN) in each transformer block with a set of expert FFNs and a lightweight router parameterized by $W \in \mathbb{R}^{d \times N}$, where d is the hidden dimension and N is the number of experts. At inference time, for an input token $X \in \mathbb{R}^{1 \times d}$, the router first computes the logits L using Equation 1. The router then applies a softmax to L to obtain expert scores S, which are used to select the top-k experts. Finally, the MoE layer output is obtained by aggregating the selected experts' outputs, weighted by their routing scores, as shown in Equation 3.

$$\mathrm{L} = XW \tag{1}$$

$$S = \mathrm{softmax}(L) \tag{2}$$

$$Y = \sum_{i \in \text{top-}k(S)} s_i \cdot FFN_i(X) \tag{3}$$

2.2 Mixture-of-Experts Deployment

LLM inference consists of two distinct phases: prefill and decode. During prefill, all tokens of the input prompt are processed simultaneously, whereas decode performs autoregressive generation, operating on one token at a time to produce the next token. The computational and memory characteristics of these phases differ substantially (Kamath et al., 2025). The prefill phase is largely compute-bound because static model parameters are reused across a large number of prompt tokens. These tokens collectively activate all or nearly all experts, and the resulting parameter reuse and high arithmetic intensity mean that expert reuse is not a primary prefill bottleneck. In contrast, the autoregressive decode phase processes only one token per sequence at each generation step. Its lower arithmetic intensity and input-dependent expert selections make expert-weight movement a major memory-bandwidth bottleneck. Accordingly, `TokenMix` primarily targets the decode phase.

Inference bottleneck: *Data movement from off-chip memory makes the decode phase memory-bound*. Most efforts to address this challenge have focused on reducing KV-cache traffic. Techniques such as Grouped Query Attention (GQA) (Ainslie et al., 2023) and Multi-Head Latent Attention (MLA) (DeepSeek-AI, 2024) reduce KV-cache size and access costs, shifting the dominant memory bottleneck toward static model-parameter fetches. In large mixture-of-experts (MoE) models, expert-weight fetches emerge as a major contributor to decode-time memory traffic. For example, during the decode phase of Qwen3-235B with batch size 1, expert-weight fetches account for 78.90% and 78.04% of total data transfer at sequence lengths of 2048 and 4096, respectively, compared with 1.1% and 2.2% attributable to KV-cache accesses. As shown in Fig. 1, larger batch sizes exacerbate this bottleneck because greater diversity in token-level expert choices causes more experts to be selected.

In `TokenMix`, we alleviate the memory-bound nature of the decode phase in MoE models by reducing the number of unique experts, and hence the volume of expert weights, fetched per layer. We quantify this reduction using the Expert Reuse Improvement (ERI) metric (Equation 4), where $\bar{E}_{\text{base}}$ and $\bar{E}_{\text{TM}}$ denote the average numbers of unique experts activated per token during the decode phase under baseline routing and `TokenMix`, respectively. A higher ERI corresponds to fewer unique expert fetches, directly reducing decode-phase memory traffic and thereby improving generation throughput.

$$\mathrm{ERI}(\%) = \left(\frac{\bar{E}_{\text{base}}}{\bar{E}_{\text{TM}}} - 1 \right) \times 100 \tag{4}$$

3 Improving Expert Reuse: Challenges and Key Insights

In this section, we investigate the factors contributing to poor expert reuse and uncover key insights that motivate our approach.

3.1 Challenges

We observe that MoE routing naturally leads to diverse expert assignments across tokens within a batch for two key reasons. First, tokens in a batch are typically drawn from independent input prompts, and routers operate in a fully *token-parallel* fashion—each token is routed without any awareness of other tokens in the batch. This independence inherently reduces the likelihood that different inputs select the same experts. Second, recent MoE architectures use *larger expert pools* that offer finer-grained specialization than those in earlier models. Thus, as the number of experts increases, the probability of reuse naturally decreases. For example, Qwen3-30B, with 128 experts, exhibits natural reuse of only 8.7%, compared with

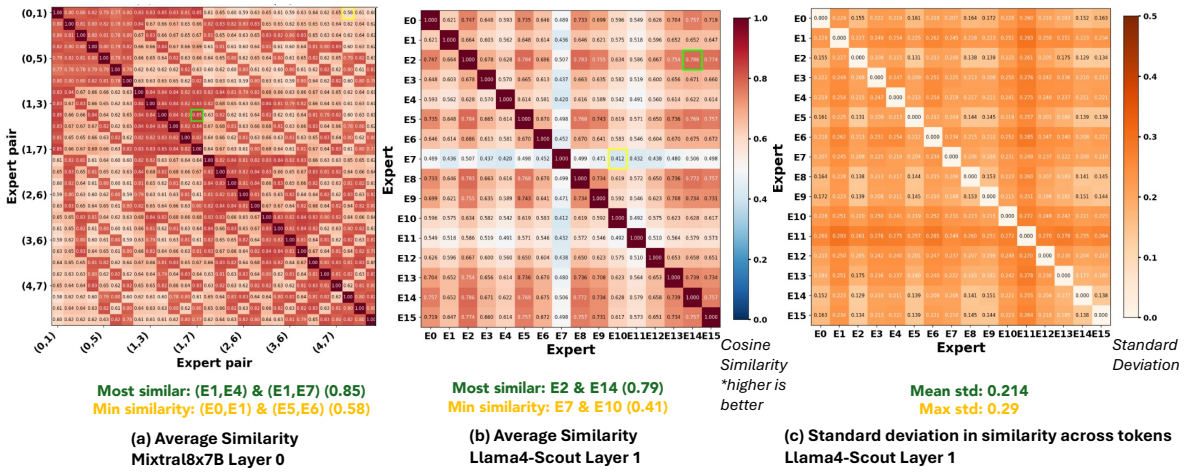


Figure 2: (a) Average cosine similarity between MoE layer outputs under different activated expert combinations in Mixtral8x7B (layer 0), and (b,c) average cosine similarity and its standard deviation across tokens for different activated experts in Llama4-Scout (layer 1). While different expert combinations yield similar MoE layer outputs on average, the degree of similarity varies considerably across tokens (mean standard deviation: 0.214; maximum standard deviation: 0.29), motivating TokenMix's dynamic token-level expert selection strategy.

15.9% in Mixtral8x7B, which has 8 experts. Under uniform random selection, the likelihood of selecting a specific subset of k experts from a pool of n experts is $1/\binom{n}{k}$, which decreases rapidly with n for a given k.

3.2 Key Insight: Functional Similarity Across Expert Combinations

Despite these challenges, we observe a key property of MoE experts that TokenMix exploits: for a particular token, different expert combinations can yield functionally similar outputs. Importantly, the expert combinations that yield functionally similar outputs vary from token to token, thereby motivating the need for a token-level strategy to boost expert reuse.

We analyze Mixtral8x7B (top-2 routing) as a case study. With 2 out of 8 experts selected per token, Mixtral8x7B yields $\binom{8}{2} = 28$ possible expert-pair combinations—a tractable number that enables exhaustive pairwise visualization. For each pair $\langle i, j \rangle$, we compute a uniform mixture $\text{Out}_{\langle i,j \rangle} = 0.5 \cdot E_i + 0.5 \cdot E_j$ and measure pairwise cosine similarity across all 28 combinations, producing a 28×28 similarity matrix averaged over 16,051 tokens per layer from 128 samples in the WikiText-2-raw (Merity et al., 2016) training split. As shown in Figure 2(a), neighboring expert combinations produce similar outputs *on average*. Figure 2(b) reveals a similar pattern of average similarity in Llama4-Scout (top-1 routing, 16 experts), with pairwise expert similarity ranging from 0.41 to 0.79. However, Figure 2(c) shows that the *degree* of similarity between MoE-layer outputs produced by different activated experts varies significantly across tokens, with a mean standard deviation of 0.214 and a maximum of 0.29. This result indicates that functional similarity among experts is strongly token-dependent, making arbitrary expert substitution nonviable. For instance, random expert selection causes accuracy to drop from 86% to 2% in Qwen3-30B (Section A.4.3). Although the average similarity across expert combinations presents an opportunity for reuse, the substantial token-level variation shows that experts are not inherently redundant—they cannot simply be merged statically or pruned without risking accuracy loss. Instead, expert similarity must be exploited *dynamically* on a per-token basis. TokenMix achieves this by identifying alternative expert combinations for each token that preserve output fidelity while improving batch-level reuse.

4 TokenMix

Motivated by the insights in Section 3, we propose TokenMix, a training-free technique that dynamically exploits expert-output similarity to improve batched expert reuse.

4.1 Overview

Figure 3 provides an overview of TokenMix and its two variants. Let us first focus on Figure 3(a). Consider a batch B of input tokens in $\mathbb{R}^d$ fed to the router of an MoE layer. TokenMix applies perturba-

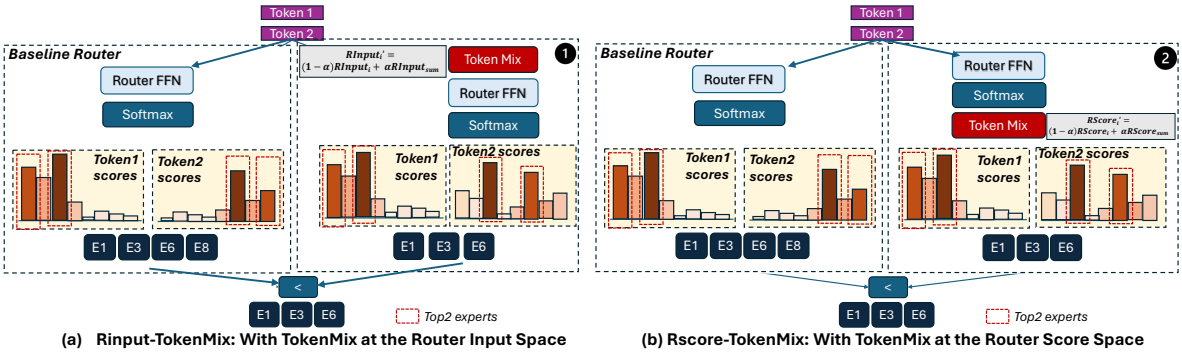


Figure 3: Overview of TokenMix. For illustration, the router selects the top 2 of 8 experts based on its output.

tions to each original input token in B, producing a new batch of tokens B'. Subsequently, both B and B' are passed to the router. The resulting routing outputs for B and B' are compared, and the configuration that yields the highest batch-level expert reuse is selected. Although the experts and associated router weights may be selected using B', we always pass the original batch of tokens B to the expert networks to compute the MoE-layer output in Equation 3. Furthermore, because the same router processes both B and B', these evaluations can be batched and executed in parallel with minimal overhead. The variant in Figure 3(b) differs in where the perturbations are applied - on the scores produced by the router's SoftMax operation rather than the inputs to the router. The tradeoffs represented by these variants are discussed further in Section 4.3.

4.2 Increasing Expert Reuse via Directional Alignment

Computing Equation 1 amounts to computing the dot product between the input token X and each router expert representation. Thus, L can be rewritten as shown in Equation 5. Here, N is the number of experts; $\|X\|_2$ and $\|W_i\|_2$ denote the magnitudes (L2 norms) of the input token X and the i^{th} router expert representation W_i, respectively; and $\beta_1, \beta_2, \ldots, \beta_N$ denote the angles between X and the router expert representations.

$$\begin{aligned} \mathrm{L} &= XW \\ &= [X \cdot W_i]_{i=1}^{N} \\ &= [\|X\|_2 \cdot \|W_i\|_2 \cos \beta_i]_{i=1}^{N}. \end{aligned} \tag{5}$$

Equation 5 suggests that MoE routing depends strongly on the cosine similarity between token representations—perfectly aligned tokens select identical sets of experts, whereas orthogonal tokens share no expert overlap. We refer the reader to Appendix A.1 for further discussion of this observation. `TokenMix` exploits this directional dependence to increase expert reuse across tokens within a batch. Our objective is to identify alternative expert combinations that improve reuse by increasing the directional (cosine) similarity among token representations. To this end, we seek a vector $v \in \mathbb{R}^d$ such that every token in the batch has a nonzero component along this common direction. The sum of all token representations in a batch naturally satisfies this property because it aggregates contributions from every token. For a batch of inputs $X_1, X_2, \ldots, X_B$, the sum vector X_{sum} is given by Equation 6. Once this common vector is identified, our goal is to increase the component of each token vector along X_{sum} while decreasing its component orthogonal to X_{sum}. We achieve this by mixing each token vector X_i with X_{sum}, as shown in Equation 7, where $\alpha \in [0, 1]$ controls the mixing strength.

$$X_{\text{sum}} = X_1 + X_2 + \cdots + X_B \tag{6}$$

$$X_i' = (1 - \alpha)X_i + \alpha X_{\text{sum}}, \tag{7}$$

Essentially, `TokenMix` realigns the inputs in a batch, rotating them closer to X_{sum}. As a result, the token representations become more directionally similar, increasing the likelihood of overlapping expert selections by the router. Through this mechanism, `TokenMix` overcomes the limitations of standard MoE routing by explicitly incorporating batch-level information into each router input.

Calibration of α: The parameter α balances reuse and accuracy. Setting $\alpha = 1$ results in complete

Model	Method	Accuracy (%) on GSM8K (Batch Size = 2 / 4 / 8 / 16)	F1 score (%) on Qasper (Batch Size = 2 / 4 / 8 / 16)	Accuracy (%) on IFEval (Batch Size = 2 / 4 / 8 / 16)
Mixtral8x7B	Baseline	60.8	92.23	61.27
	RInput-TokenMix	59.9 / 59 / 59.5 / 59.8	92.50 / 93.06 / 92.51 / 92.23	60.91 / 60.81 / 60.43 / 61.39
	RScore-TokenMix	59.5 / 59.5 / 59.6 / 61	92.51 / 93.06 / 92.51 / 92.23	60.67 / 61.51 / 61.71 / 62.21
Qwen3-30B	Baseline	86.35	75.08	88.97
	RInput-TokenMix	88.1 / 85.4 / 87.19 / 87.9	73.94 / 74.70 / 75.45 / 75.08	79.8/ 85.4 / 82.3 / 80.2
	RScore-TokenMix	89.6 / 89.6 / 88.9 / 90.2	73.94 / 74.70 / 75.45 / 75.08	88.49 / 88.51 /88.73 / 88.13
Llama4-Scout	Baseline	80.59	98.29	89.45
	RInput-TokenMix	81.27 / 81.73 / 81.12 / 80.97	98.29 / 98.54 / 98.54 / 98.54	89.45/ 89.70 / 88.9 / 89
	RScore-TokenMix	81.35 / 81.12 / 81.65 / 80.89	98.29 / 98.54 / 98.55 / 98.54	89.33/ 90.05 / 89.69 / 90.29
Qwen3-235B	Baseline	91.81	73.93	90.53
	RInput-TokenMix	87.11 / 87.72 / 87.34 / 89.39	73.94 / 74.70 / 75.08 / 73.56	88.97 / 87 / 87.2 / 86.6
	RScore-TokenMix	91.89 / 91.74 / 91.59 / 91.66	73.94 / 74.70 / 75.08 / 73.5	90.89 / 91.85 / 91.13 / 90.89
DeepSeek-V2-Lite	Baseline	43.5	41.5	37.46
	RInput-TokenMix	38.82 / 40.56 / 42.3 / 44.1	35.5 / 42.52 / 41.53 / 41.53	37.15 / 38.08 / 36.41 / 36.04
	RScore-TokenMix	38.82 / 44.1 / 43.4 / 42.6	35.5 / 41.4 / 41.4 / 41.3	37.0 / 38.1 / 39.6 / 37.7

Table 1: Accuracy impact of TokenMix on the GSM8K, Qasper, and IFEval datasets.

expert reuse but is likely to severely affect generation quality, whereas setting $\alpha = 0$ leaves the baseline routing choices unchanged. Across all our experiments, we find that setting $\alpha = 1/B$, where B is the batch size during generation, yields high reuse gains without affecting accuracy. Although we adopt $\alpha = 1/B$ as a practical default, its optimal value can be calibrated for each model and deployment scenario using a binary search over $[0, 1]$ (see Alg. 1 in Appendix A.2). Calibration is a one-time cost because it needs to be performed only once for a given model.

4.3 Applying TokenMix at Router Inputs vs. Outputs

The proposed token-mixing operation can be introduced at multiple points in the MoE routing pipeline, including the router input space, the router output space (i.e., the output of the router feed-forward network (FFN)), and the post-softmax routing-score space. Owing to the linear structure of the router FFN, perturbations applied in the input space are equivalent to perturbations applied in the router output space. In contrast, perturbations applied after the softmax lead to different accuracy–reuse trade-offs, as evaluated in Section 5. This difference arises from the nonlinear nature of the softmax operation, which amplifies relative differences between logits through normalization. We refer to token mixing in the router input space as RInput-TokenMix (❶ in Fig. 3) and mixing in the post-softmax space as RScore-TokenMix (❷ in Fig. 3).

Model	Chosen / Total Experts	Active Params	Total Params
Mixtral8x7B (Jiang et al., 2024)	2 / 8	13B	47B
Qwen3-30B (Yang et al., 2025)	8 / 128	3B	30B
Qwen3-235B (Yang et al., 2025)	8 / 128	22B	235B
Llama4-Scout (Meta AI, 2025)	1 / 16	17B	109B
DeepSeek-V2-Lite (DeepSeek-AI, 2024)	6 / 64	2.4B	15.7B

Table 2: MoE benchmark suite.

5 Evaluation

We present our evaluation setup and the key results that demonstrate the effectiveness of TokenMix.

5.1 Setup

Models. For the primary evaluation, we consider the five models listed in Table 2. These models vary widely in total parameter count, number of experts selected per input, and total number of experts. All models are evaluated at bfloat16 precision (Burgess et al., 2019), except Mixtral, which is evaluated at 4-bit precision. For all models, we conduct inference with a maximum prompt length of 4096. All experiments use the lm-eval package (Gao et al., 2024). We use each model's default parallelism strategy, typically tensor or data parallelism. For the code-generation evaluation, we additionally consider Qwen3-Coder-30B-A3B-Instruct (Qwen Team, 2025), a coding-specialized MoE model, and Qwen3.5-35B-A3B (Qwen Team, 2026), the latest in the Qwen MoE model family.

Datasets. We evaluate TokenMix across four domains: mathematical reasoning (the GSM8K (Cobbe et al., 2021) test set, which contains 1,319 samples), knowledge-based understanding (the Qasper (Dasigi et al., 2021) test set, which contains 431 examples), instruction following (the

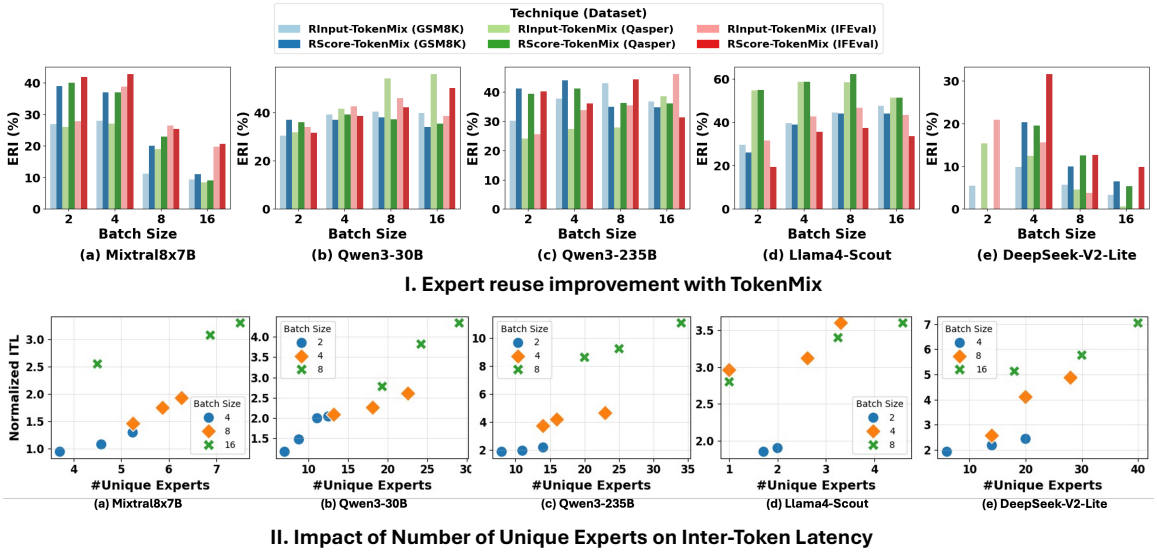


Figure 4: I. Expert reuse improvement (ERI, %) with RInput-TokenMix and RScore-TokenMix across the benchmark models, datasets, and batch sizes. II. Impact of reducing unique expert fetches on inter-token latency during decode.

IFEval (Zhou et al., 2023) test set, which contains 541 examples), and code generation (the HumanEval+ (Liu et al., 2023) test set, which contains 164 examples). We report exact-match accuracy for GSM8K, F1 scores for Qasper, prompt-level strict accuracy for IFEval, and pass@1 accuracy for HumanEval+.

Hardware. All experiments were conducted on NVIDIA H100-80GB GPUs.

5.2 Impact of TokenMix

Accuracy vs. Expert Reuse. Table 1 and Figure 4(I) summarize the accuracy impact and reuse improvements of TokenMix over the baselines for the GSM8K, Qasper, and IFEval datasets. The ERI values in Fig. 4(I) are computed using Equation 4. Across a diverse set of MoE models, TokenMix improves expert reuse by 25–60% with negligible accuracy degradation. To evaluate scaling beyond the batch sizes reported above, we extend the Qwen3-30B experiments on GSM8K to batch sizes 32 and 64. TokenMix maintains robust ERI gains of 37% and 26%, respectively, with accuracy changes of only $+1.0$ and -0.1 percentage points relative to the baseline. These results demonstrate that TokenMix continues to improve expert reuse at larger batch sizes without compromising accuracy. Comparing the mixing strategies, RInput-TokenMix achieves higher reuse than RScore-TokenMix but exhibits mild accuracy degradation on larger models such as Qwen3-235B and on complex tasks such as instruction following. This behavior stems from softmax amplification, as discussed in Section 4: perturbations introduced before the softmax can be magnified at the score level, increasing both reuse potential and accuracy loss. DeepSeek-V2-Lite is particularly sensitive to the approximations introduced by TokenMix at the default $\alpha = 1/B$ setting and exhibits noticeable accuracy degradation. To preserve accuracy, we adopt a more conservative setting of $\alpha = 1/B^2$ for this model. This setting produces lower reuse improvements than those observed for the other models, with reuse gains of less than 2% in some configurations. This result highlights the importance of calibrating α for each model, as discussed in Section 4. Table 3 summarizes the impact of TokenMix on code-generation benchmarks. RInput-TokenMix causes substantial accuracy degradation on these tasks because perturbations introduced before the softmax can be amplified at the routing-score level. In contrast, RScore-TokenMix operates directly on the post-softmax routing scores and avoids this amplification, preserving code-generation accuracy while improving expert reuse. We therefore recommend RScore-TokenMix for complex coding workloads and other settings in which output precision is critical.

Model	Method	pass@1 (%) (Batch size = 4/8/16)	Reuse (%) (Batch size = 4/8/16)
Qwen3-Coder	Baseline	86.6 / 88.4 / 85.4	-
	RInput-TokenMix	40.2 / 37.2 / 42.7	15.64 / 20.47 / 26.15
	RScore-TokenMix	85.4 / 86.6 / 86.0	36.2 / 40.65 / 42.01
Qwen3.5 -35B-A3B	Baseline	47.0 / 47.6 / 40.2	-
	RInput-TokenMix	28.7 / 28.7 / 31.1	19.0 / 15.3 / 10.5
	RScore-TokenMix	41.5 / 40.2 / 44.5	50.1 / 55.8 / 63.9

Table 3: Impact of TokenMix on HumanEval+ code-generation performance (pass@1).

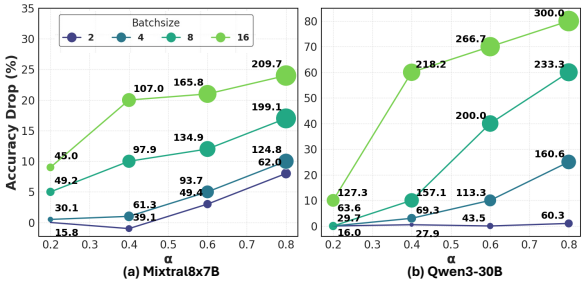


Figure 5: Accuracy–reuse trade-off as α varies. Point size and the adjacent value indicate reuse (%).

Latency Benefits. The inter-token latency (ITL) benefits of TokenMix are quantified in Fig. 4(II), which plots normalized ITL as a function of the number of unique experts fetched during decoding. ITL was measured on eight NVIDIA H100 GPUs and averaged over five runs, with α varied to modulate the average number of experts activated per decoded token. Expert-weight fetches from off-chip memory constitute a dominant fraction of total decode-time memory traffic (Section 2); reducing the number of unique experts fetched therefore directly translates into ITL improvements. Across all benchmarks and batch sizes, TokenMix achieves a **12%–28%** reduction in ITL on NVIDIA H100 GPUs, driven by the expert-reuse improvements shown in Fig. 4(I).

5.3 Ablation Analysis

We examine the robustness of TokenMix along two axes: sensitivity to the parameter α and variability across different batch compositions.

Impact of α. The applicability of $\alpha = 1/B$, where B is the batch size, across a wide range of benchmarks saves users from tuning this hyperparameter separately for each model and dataset. However, users with sufficient compute resources can tune α to achieve different accuracy–runtime trade-offs. Fig. 5 depicts the impact of α on accuracy and reuse for RScore-TokenMix on the Qwen3-30B and Mixtral8x7B MoE models across different batch sizes on GSM8K.

Expert reuse with randomly shuffled datasets. Fig. 6 shows the variability of expert reuse improvement across different batches of the dataset. We observe low variance with a maximum standard deviation of 3.51% for Llama4-Scout and 2.37% for Qwen3-30B (both at batch size 2). To further quantify reuse variation across inference runs, we evaluate TokenMix on Qasper with DeepSeek-V2-Lite-Chat under four random seeds. Accuracy remains within ±0.5% of baseline across all seeds,

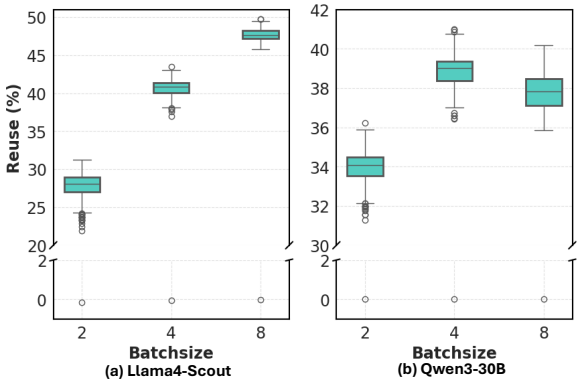


Figure 6: Reuse statistics across different batches of the GSM8K test set with (a) Llama4-Scout and (b) Qwen3-30B.

and mean reuse improvements are 19.43%, 8.96%, and 5.46% at batch sizes 4, 8, and 16 (standard deviations: 1.83%, 3.14%, and 3.22%), respectively. Furthermore, random shuffling of the test dataset preserves both accuracy and average reuse, closely matching the results in Table 1 and Fig. 4. Collectively, these results confirm that TokenMix's improvements are robust to batch composition and consistently translate across different dataset permutations and inference runs.

5.4 Combining TokenMix with Expert Pruning and Cache-Conditional MoE

We evaluate whether TokenMix is complementary to existing efficient MoE inference techniques by combining it with EasyEP (Dong et al., 2025) and Cache-Conditional MoE (Skliar et al., 2025). For both techniques, TokenMix is applied directly without retuning its hyperparameters. As shown in Table 4, TokenMix provides reuse improvements over both baselines without impairing accuracy.

Method	Accuracy (%)	Reuse (%)
EasyEP	85.96	-
TokenMix + EasyEP	85.87	27
Cache-Conditional MoE	87.11	-
TokenMix + Cache-Conditional MoE	86.7	20

Table 4: Applying TokenMix jointly with expert pruning and Cache-Conditional MoE for Qwen3-30B on GSM8K at batch size 16.

6 Related Efforts

Prior work on efficient MoE inference falls into two categories.

Expert size reduction methods such as expert merging, pruning, and quantization (Bai et al., 2025; Dong et al., 2025; Lu et al., 2024) reduce

the memory footprint of individual experts but do not address the poor scaling of expert reuse with batch size. TokenMix is complementary to these approaches and can be combined with them, as demonstrated in Sec. 5.

Cache-aware routing (Skliar et al., 2025) improves temporal expert reuse across consecutive inference steps by biasing selection toward cached experts. Unlike `TokenMix`, this does not increase intra-batch reuse — all experts required by a batch must still be fetched. This approach only targets systems with caches large enough to hold expert weights. Since the two approaches operate along orthogonal reuse axes, they are complementary, as demonstrated in Sec. 5.

7 Conclusion

We propose `TokenMix`, a novel decode-time optimization that improves the efficiency of MoE models by promoting expert reuse across batched inputs. `TokenMix` perturbs the router's internal representations to steer expert selection toward greater reuse across the batch. Extensive evaluations of state-of-the-art MoE models with 16B to 235B parameters demonstrate consistent expert reuse improvements of 25–60% across several domains, with minimal impact on accuracy.

8 Limitations

`TokenMix` incurs modest accuracy degradation in some settings because it prioritizes expert reuse when selecting between the baseline and `TokenMix` routing outputs. Although both routing options largely preserve output quality, prioritizing greater reuse can occasionally result in expert choices that differ from those of the baseline. A more sophisticated selection strategy that jointly considers expert reuse and expert quality may mitigate this degradation but is beyond the scope of this work. Moreover, `TokenMix` does not reduce model size, unlike expert-size-reduction approaches that explicitly prune or merge experts to reduce the parameter count.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore. Association for Computational Linguistics.

Sikai Bai, Haoxi Li, Jie Zhang, Zicong Hong, and Song Guo. 2025. Diep: Adaptive mixture-of-experts compression through differentiable expert pruning. *Preprint*, arXiv:2509.16105.

Neil Burgess, Jelena Milanovic, Nigel Stephens, Konstantinos Monachopoulos, and David Mansell. 2019. Bfloat16 processing for neural networks. In *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, pages 88–91.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *Preprint*, arXiv:2401.06066.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. *CoRR*, abs/2105.03011.

DeepSeek-AI. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *CoRR*, abs/2405.04434.

Zican Dong, Han Peng, Peiyu Liu, Wayne Xin Zhao, Dong Wu, Feng Xiao, and Zhifeng Wang. 2025. Domain-specific pruning of large mixture-of-experts models with few-shot demonstrations. *Preprint*, arXiv:2504.06792.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Shwai He, Daize Dong, Liang Ding, and Ang Li. 2025. Towards efficient mixture of experts: A holistic study of compression techniques. *Preprint*, arXiv:2406.02500.

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, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, and 7 others. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Aditya K. Kamath, Ramya Prabhu, Jayashree Mohan, Simon Peter, Ramachandran Ramjee, and Ashish Panwar. 2025. Pod-attention: Unlocking full prefill-decode overlap for faster LLM inference. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, ASPLOS 2025, Rotterdam, Netherlands, 30 March 2025 - 3 April 2025*, pages 897–912. ACM.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. 2024. Merge, then compress: Demystify efficient smoe with hints from its routing policy. *Preprint*, arXiv:2310.01334.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Preprint*, arXiv:2305.01210.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. 2024. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6159–6172, Bangkok, Thailand. Association for Computational Linguistics.

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

Meta AI. 2025. The llama 4 herd: The beginning of a new era of natively multimodal AI innovation. `https://ai.meta.com/blog/llama-4-multimodal-intelligence/`. Meta AI Blog; accessed Apr. 5, 2025.

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, and 108 others. 2025. gpt-oss-120b gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Qwen Team. 2025. Qwen3-Coder: Agentic Coding in the World. `https://qwen.ai/blog?id=qwen3-coder`. Accessed: 2026-09-30.

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents.

Andrii Skliar, Ties van Rozendaal, Romain Lepert, Todor Boinovski, Mart van Baalen, Markus Nagel, Paul Whatmough, and Babak Ehteshami Bejnordi. 2025. Mixture of cache-conditional experts for efficient mobile device inference. *Preprint*, arXiv:2412.00099.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *Preprint*, arXiv:2311.07911.

A Appendix

In this section, we provide additional details and present further experiments that help us understand the impact of `TokenMix`.

A.1 Mathematical Insights on Routing Decisions

Routers in MoE models are small relative to the expert networks (Jiang et al., 2024; Yang et al., 2024; Dai et al., 2024), yet they play a critical role in determining overall generation throughput by dictating expert reuse.

We analyze Equation 5 to identify the factors that influence expert reuse across tokens. Consider two scenarios for a pair of input token vectors X_1 and X_2: (1) the tokens are aligned in the same direction (cosine similarity equal to 1) but differ in magnitude (L2 norm); and (2) the tokens have equal magnitudes but differ in direction, with an angle θ between them. Let the corresponding router outputs from Equation 5 be L_1 and L_2, respectively.

Case 1: Suppose $X_2 = kX_1$ for $k > 0$; that is, X_2 is directionally aligned with X_1 but has magnitude kx_1. The router output L_2 is given by Equation 8. In this case, only the magnitude of the router output changes with respect to that of X_1, while the relative ordering of the expert scores remains unchanged. Consequently, the top-k expert selections for X_1 and X_2 are identical; only their associated expert weights differ.

$$\begin{aligned} \mathrm{L}_2 &= \left[x_2 w_i \cos\beta_i\right]_{i=1}^{N} \\ &= \left[kx_1 w_i \cos\beta_i\right]_{i=1}^{N}. \end{aligned} \tag{8}$$

Case 2: Let the angles between X_1 and the router expert representations be $\beta_1, \beta_2, \ldots, \beta_N$. The angle between X_2 and the i^{th} router expert representation is therefore $\beta_i \pm \theta$. Unlike in Case 1, Equation 5 shows that θ affects the relative ordering of the expert scores. In the extreme case in which X_2 and X_1 are orthogonal, the router output L_2 is given by Equation 9. Because sine and cosine exhibit opposite trends across angles, orthogonal token vectors tend to select completely different sets of experts and expert routing weights.

$$\begin{aligned} \mathrm{L}_2 &= \left[x_1 w_i \cos(\beta_i \pm \tfrac{\pi}{2})\right]_{i=1}^{N} \\ &= \left[\pm x_1 w_i \sin\beta_i\right]_{i=1}^{N}. \end{aligned} \tag{9}$$

The preceding analysis suggests that token direction plays a critical role in determining expert selection, whereas token magnitude affects expert weights.

A.2 Calibration

In this section, we describe our calibration algorithm for determining the hyperparameter α. As discussed in Section 4, α controls the degree to which the router inputs are mixed. A higher α produces more aggressive mixing and therefore greater reuse, but it is also more likely to cause the experts selected for a token to deviate from the baseline selections. As shown in Algorithm 1, the calibration algorithm begins with $\alpha = 1/B$, where B is the batch size, and evaluates the MoE-layer output for every token in the calibration dataset with and without mixing. The per-token mean squared error (MSE) values between these outputs are then computed and averaged. If the average MSE exceeds a preset threshold T, α is reduced; if it is below T, α is increased.

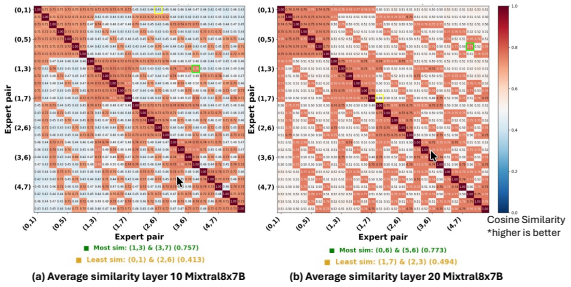


Figure 7: Average cosine similarity between MoE-layer outputs under different activated expert combinations in Mixtral8x7B, for layers 10 and 20.

Algorithm 1 Calibration

Input: MoE model M, dataset D with N tokens, threshold T, batch size B
l = 0.0, h=1.0, $\alpha = 1/B$
Output: α
while $\alpha < 1$ **do**
  for $i \leftarrow 1$ to N **do**
    Calculate the i^{th} token output $O_{i,\text{baseline}}$ using baseline routing
    Calculate the i^{th} token output $O_{i,\texttt{TokenMix}}$ using `TokenMix`
    Calculate MSE_i between $O_{i,\text{baseline}}$ and $O_{i,\texttt{TokenMix}}$
  end for
  $MSE \leftarrow \frac{1}{N}\sum_i \text{MSE}_i$
  if $MSE > T$ **then**
    $h \leftarrow \alpha$
    $\alpha \leftarrow (l + h)/2$ ▷ A high MSE can reduce accuracy; decrease α
  else if $MSE < T$ **then**
    $l \leftarrow \alpha$
    $\alpha \leftarrow (l + h)/2$
  end if
end while

Across all our experiments, we use the WikiText-2-raw training split (Merity et al., 2016) as the calibration dataset.

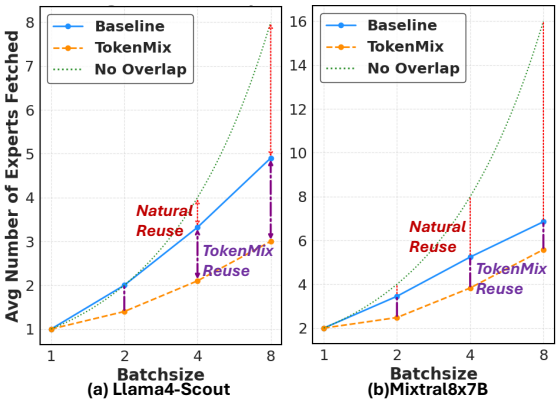


Figure 8: Average number of experts fetched in an MoE layer for Llama4-Scout (1 of 16 experts selected per token) and Mixtral8x7B (2 of 8 experts selected per token) with (i) no overlap between the experts selected by tokens in a batch (worst-case reuse); (ii) natural (baseline) expert overlap across tokens; and (iii) `TokenMix`-induced expert overlap.

A.3 Functional Similarity Across Additional Layers

In this section, we extend the functional similarity analysis presented in Section 3 to additional layers of Mixtral8x7B. Figure 7 shows the average pairwise cosine similarity matrices for layers 10 and 20, complementing the layer 0 analysis shown in Figure 2(a).

The structured pattern of high similarity persists across layers, confirming that functional similarity among expert combinations is not specific to early layers but is a consistent property of the model. In layer 10, the most similar pair of expert combinations is (1, 3) and (3, 7), with a similarity of 0.757, whereas the least similar pair is (0, 1) and (2, 6), with a similarity of 0.413. In layer 20, the most similar pair is (0, 6) and (5, 6), with a similarity of 0.773, whereas the least similar pair is (1, 7) and (2, 3), with a similarity of 0.494.

A.4 Further Ablation Analysis

A.4.1 Average Number of Experts Activated

Fig. 8 illustrates the average number of experts activated across different MoE models on the GSM8K dataset before and after applying `TokenMix`. Fig. 1 presents the same analysis for Qwen3-30B and Qwen3-235B. Across all batch sizes, `TokenMix` reduces the number of experts fetched.

A.4.2 Impact on Intermediate Representations

To measure the impact of `TokenMix`'s routing changes on intermediate representations at every MoE layer, we compute the mean squared error (MSE) between the weighted expert outputs under baseline and `TokenMix` routing at each MoE layer of Qwen3-30B-A3B across 100 input samples and multiple decode steps. The MSE averages 0.003 (standard deviation: 0.02) across all MoE layers, confirming that `TokenMix` produces intermediate representations that are numerically close to those of the baseline at every layer. This result is consistent with the functional similarity across expert combinations established in Section 3 and explains why end-task accuracy is preserved despite routing changes.

A.4.3 Impact of Random Expert Selection on Accuracy

To further demonstrate the effectiveness of `TokenMix`, we conduct an experiment in which the experts for each token are selected randomly. As shown in Table 5, accuracy declines sharply across all models on the GSM8K dataset. This result underscores the need for an intelligent expert-selection policy that balances expert reuse and accuracy.

Model	Accuracy (%)
Mixtral8x7B	32.00
Qwen3-30B	2.00
Llama4-Scout	1.60
Qwen3-235B	0.70
DeepSeek-V2-Lite	0.14

Table 5: Accuracy under random expert selection (batch size = 16, GSM8K).

A.5 Generation Quality with `TokenMix`

We inspect the outputs generated with token mixing in Tables 6 and 7. As these examples illustrate, token mixing does not degrade generation quality.

Prompt	Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?
Baseline	Janet sells 16 - 3 - 4 = «16-3-4=9»9 duck eggs a day. She makes 9 * 2 = $«9*2=18»18 every day at the farmer's market. #### 18
`TokenMix`	Janet sells 16 - 3 - 4 = «16-3-4=9»9 duck eggs a day. She makes 9 * 2 = $«9*2=18»18 every day at the farmer's market. #### 18

Table 6: Example 1.

Prompt	Darrell and Allen's ages are in the ratio of 7:11. If their total age now is 162, calculate Allen's age 10 years from now.
Baseline	The total ratio representing their ages is 7+11= «7+11=18»18 Since the fraction of the ratio that represents Allen's age is 11/18, Allen's current age is 11/18*162 = «11/18*162=99»99 If Allen is currently 99 years old, in 10 years he will be 99+10 = «99+10=109»109 years old #### 109
`TokenMix`	The total ratio representing their ages is 7+11= «7+11=18»18 Since the fraction of the ratio that represents Allen's age is 11/18, Allen's current age is 11/18*162 = «11/18*162=99»99 If Allen is currently 99 years old, in 10 years he will be 99+10 = «99+10=109»109 years old #### 109

Table 7: Example 2.