# RESCODE: Accurate Expert Compression of Sparse MoE Models via Residual-code Restoration

**Hyunwoo Yang, Minjun Kim, U Kang***
Seoul National University
{jg020247, minjun.kim, ukang}@snu.ac.kr

## Abstract

How can we accurately compress a sparse mixture-of-experts (SMoE) model? SMoE models are widely adopted in modern LLMs for reducing activated computation, but their deployment remains memory-intensive due to extensive expert parameters. To address this bottleneck, expert compression reduces redundant expert parameters. However, previous methods decrease the number of experts involved in token-dependent routing, inducing output deviations from the original model. Expert decomposition is a promising alternative, but existing approaches suffer from three major limitations: a single shared base, component memory overhead, and suboptimal objectives. In this work, we introduce five desired properties for accurate expert compression. Then, we propose RESCODE, an accurate expert decomposition method based on residual-code restoration. RESCODE forms compact expert clusters, coalesces residuals into cluster-shared low-rank bases, and learns compact codes to accurately recover the original routed outputs. RESCODE provides the state-of-the-art performance for compressing SMoE models on various tasks, achieving up to 2.43 percentage points higher average accuracy than the strongest baseline.

## 1 Introduction

*Given a pretrained sparse mixture-of-experts (SMoE) model, how can we reduce the number of expert parameters while preserving accuracy?* SMoE models (Shazeer et al., 2017; Fedus et al., 2022; Cai et al., 2025) are common in modern LLMs, as they reduce activated computation through sparse routing over multiple expert networks. Despite this sparse computation, deploying SMoE models on resource-limited devices remains challenging due to the large memory footprint of expert parameters. To reduce this footprint, prior works explore two complementary directions: lowering the bit-width of expert weights through quantization (Huang et al., 2025b; Chen et al., 2025b; Xie et al., 2025; Huang et al., 2025a) and reducing the number of expert parameters through expert compression (Yang et al., 2024; Liu et al., 2024; Bai et al., 2025; Li et al., 2026). We focus on expert compression, as it directly targets the parameter count of SMoE experts. As recent SMoE models continue to increase both the number and size of experts, reducing expert memory while preserving performance is an increasingly important problem (Ludziejewski et al., 2025; Liu et al., 2026).

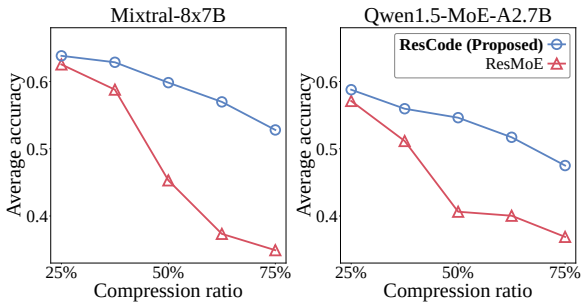
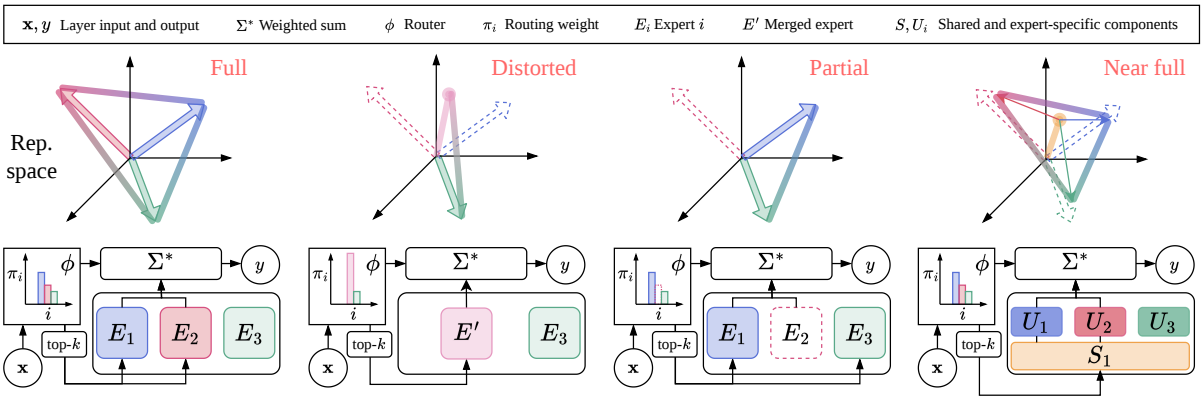



Figure 1: Accuracy–compression trade-off on Mixtral-8×7B and Qwen1.5-MoE-A2.7B models across compression ratios from 25% to 75%. RESCODE maintains substantially higher accuracy than ResMoE. Exact values are listed in Appendix F.

Expert compression methods are broadly grouped into three categories based on how they reduce expert parameters: merging, pruning, and decomposition. Expert merging and pruning directly reduce the number of experts by merging similar experts or removing less important ones (Chen et al., 2025a; Huang et al., 2025b; Lasby et al., 2026). However, once the number of experts is reduced, tokens are no longer processed by the same experts selected in the original model, which leads to output deviation. By contrast, expert decomposition keeps the routed experts in place and compresses their internal representations, making it better suited to preserve the original model out-

*Corresponding author.




(a) Original SMoE layer (b) Expert merging (c) Expert pruning (d) Expert decomposition

Figure 2: Comparison of expert compression techniques. While expert merging and pruning reduce the number of experts, expert decomposition stores expert-specific components to align assignment-level routing.

puts (Li et al., 2025). Residual restoration instantiates this idea by expressing each expert as a shared common expert and an expert-specific residual (Ai et al., 2025). This design aligns well with SMoE structure: the common expert captures redundancy shared across experts, while the residuals preserve the expert-specific behavior for token-wise routing.

However, existing residual restoration methods face the following three key challenges.

- **One size does not fit all.** Existing methods rely on a single shared component for all experts. This ignores expert heterogeneity and thereby results in larger output errors under compression (see Table 1).
- **Memory overhead of residuals.** Restoring expert-specific behavior requires residual components for each individual expert. This residual overhead scales with the number of experts, reducing the actual memory savings under aggressive compression.
- **Suboptimal restoration.** Existing methods optimize weight reconstruction error, although performance degradation is more directly affected by output reconstruction error.

To address these challenges, we first identify five desired properties for expert compression methods: per-expert preservation, inter-layer budget flexibility, intra-layer budget flexibility, linear complexity, and data-aware compression. Based on these properties, we then propose ResCode, an accurate expert compression method for SMoE models through diagonal residual-code coalescing. ResCode merges experts into saliency-aware clusters, coalesces expert residuals into cluster-shared low-rank bases, and learns compact residual codes to recover the original routed outputs. Figure 1 shows that ResCode maintains substantially higher accuracy than existing methods under aggressive compression (see Section 5.2).

We summarize our contributions as follows:

- **Objective formulation.** We identify five desired properties for expert compression methods, providing a clear objective for compression algorithms (see Table 1 and Section 3.2).
- **Algorithm.** We propose ResCode, an accurate expert compression method. ResCode compresses SMoE experts by forming expert clusters, representing residuals with compact diagonal codes, and learning lightweight residual codes for output recovery (see Section 4).
- **Experiments.** We show that ResCode consistently outperforms existing competitors across diverse tasks, achieving up to 2.43 percentage points (pp) higher average accuracy (see Section 5).

Our code is available at https://github.com/snudatalab/ResCode.

## 2 Preliminaries and Related Works

We briefly describe the preliminaries and related works on expert compression. We define frequently used notations and provide an extended review of related work in Appendices A and C, respectively.

**SMoE layer.** A sparsely activated mixture-of-experts (SMoE) layer consists of $N$ expert functions $E_i$ and a router $\phi$ that selects only a small subset of experts for each input. Given an input $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$, let $\mathbf{W} = \{\mathbf{W}_i\}_{i=1}^{N}$ denote the expert parameters, where $\mathbf{W}_i$ parameterizes the expert $E_i : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$. The layer output $y^{\ell}(\mathbf{x}) \in \mathbb{R}^{d_{\text{out}}}$ is

computed as a weighted sum of expert outputs:

$$y(\mathbf{x}) = \sum_{i \in T_k(\mathbf{x})} \pi_i(\mathbf{x}) E_i(\mathbf{x}), \tag{1}$$

$$E_i(\mathbf{x}) = \mathbf{W}_i^{(\text{down})} \left( \sigma(\mathbf{W}_i^{(\text{gate})} \mathbf{x}) \odot \mathbf{W}_i^{(\text{up})} \mathbf{x} \right),$$

where $T_k(\cdot)$ denotes the set of top-$k$ experts selected by router $\phi$, $\pi_i(\mathbf{x})$ denotes the routing weight for expert $E_i$, $\sigma(\cdot)$ denotes the activation function, and $\odot$ denotes the element-wise product.

**Expert pruning and merging.** Expert pruning (He et al., 2025; Lasby et al., 2026) and merging (Li et al., 2024b; Chen et al., 2025a) compress the MoE layer by reducing the number of experts, resulting in the compressed expert parameters

$$\hat{\mathbf{W}} = \{\mathbf{W}'_c\}_{c=1}^{C} \ (C < N),$$

where each $\mathbf{W}'_c$ parameterizes an expert $E'_i : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$. Expert pruning reduces the expert count by retaining only salient experts, whereas expert merging reduces it by clustering the original experts and replacing each cluster with a merged expert. As parameter reduction comes at the cost of removing or replacing routed experts, this disrupts the original token-to-expert assignments and leads to output deviation (see Property 1 in Section 3.2).

**Expert decomposition.** Expert decomposition compresses experts by representing each expert with shared and expert-specific components. The compressed parameters are

$$\hat{\mathbf{W}} = \{\mathcal{S}, \mathcal{U}\},$$

where $\mathcal{S} = \{\mathbf{S}_m\}_{m=1}^{M}$ and $\mathcal{U} = \{\mathbf{U}_i\}_{i=1}^{N}$ denote the shared and expert-specific components, respectively. Each expert parameter set $\mathbf{W}_i$ is decomposed into a shared component $\mathbf{S}_{\gamma(i)}$ and an expert-specific component $\mathbf{U}_i$, yielding the approximation $\hat{\mathbf{W}}_i(\mathbf{S}_{\gamma(i)}, \mathbf{U}_i)$. Unlike merging or pruning, expert decomposition avoids assignment-level deviation caused by reduction of experts.

Expert decomposition methods are categorized by the reconstruction form: multiplicative reconstruction and additive residual restoration. Multiplicative reconstruction methods such as MoE-SVD (Li et al., 2025) reconstruct the expert through the product of shared and expert-specific factors.

$$\hat{\mathbf{W}}_i = \mathbf{S}_{\gamma(i)} \mathbf{U}_i.$$

Table 1: Desired properties of expert compression methods. P$i$ refers to Property $i$ (see Section 3.2).

| Method | P1 | P2 | P3 | P4 | P5 |
|---|---|---|---|---|---|
| HC-SMoE (2025a) | | ✓ | | ✓ | ✓ |
| REAP (2026) | | ✓ | | ✓ | ✓ |
| MoE-SVD (2025) | | ✓ | ✓ | ✓ | |
| ResMoE (2025) | ✓ | ✓ | ✓ | | |
| **RESCODE (Proposed)** | ✓ | ✓ | ✓ | ✓ | ✓ |

However, the product form limits expressivity because the reconstructed expert remains constrained by the low-rank bottleneck of shared components.

In contrast, *residual restoration* methods such as ResMoE (Ai et al., 2025) recover each expert by adding a compact residual to the assigned shared component as follows:

$$\hat{\mathbf{W}}_i = \mathbf{S}_{\gamma(i)} + \mathbf{U}_i.$$

Such additive form is more expressive as each expert is recovered by adding a low-rank residual $\mathbf{U}_i$ to a full-rank shared component $\mathbf{S}_{\gamma(i)}$, rather than imposing a low-rank constraint on the entire expert.

## 3 Problem and Desired Properties

### 3.1 Problem Definition

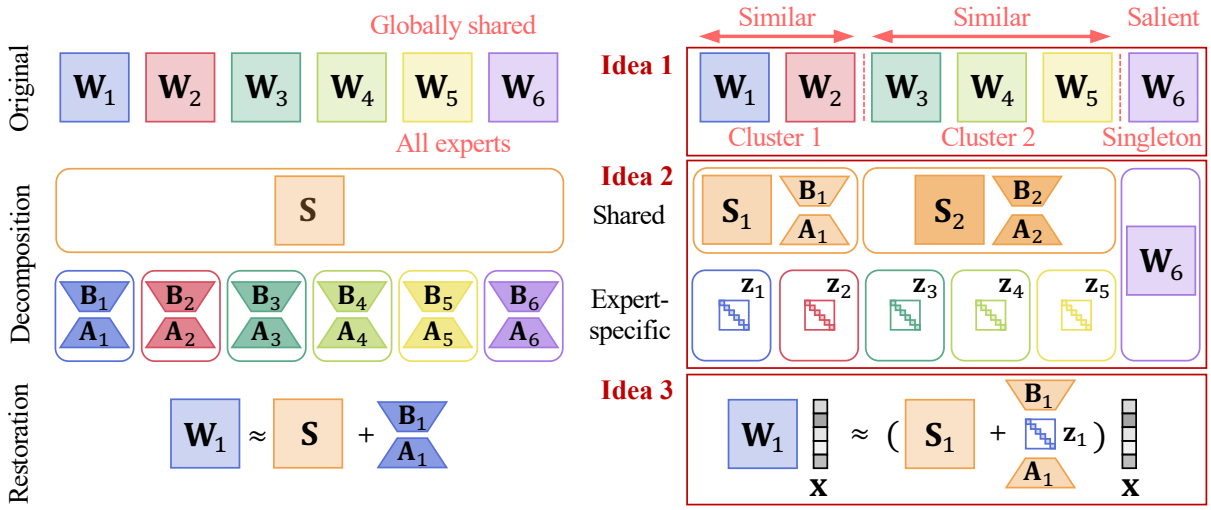
Given a pre-trained SMoE model, expert compression aims to reduce expert parameters while preserving the original model performance. We formulate the problem in Problem 1.

**Problem 1** (Expert Compression)**.** *Given a pre-trained SMoE model with parameters $\Theta_E$, target ratio $\tau \in (0, 1)$, and calibration data $\mathcal{D}$, the goal of expert compression $f_\tau(\cdot)$ is to obtain compressed parameters $\widehat{\Theta}_E = f_\tau(\Theta_E; \mathcal{D})$ that achieve the highest accuracy while satisfying $|\widehat{\Theta}_E| \leq \tau |\Theta_E|$.*

### 3.2 Desired Properties

We propose five desired properties for expert compression methods to preserve the original model's behavior. Figure 2 shows that merging collapses expert outputs into an averaged direction, pruning replaces removed experts with remaining ones, and decomposition restores the original outputs with expert-specific components. These observations show that preserving the original routing behavior is essential to avoid output deviation. We introduce Property 1 that forces a compression method to retain a compressed counterpart for every expert.

**Property 1** (Per-expert Preservation)**.** *A compressed expert set $\{\widehat{E}_i\}$ should have the same size as the original $\{E_i\}$, i.e., $|\{\widehat{E}_i\}| = |\{E_i\}| = N$.*




(a) ResMoE (b) ResCode (Proposed)

Figure 3: Comparison of ResMoE (Ai et al., 2025) and ResCode. Our main ideas are 1) cluster-wise restoration, 2) diagonal residual-code, and 3) output-guided fitting. See Section 4.1 for details.

At the same time, compression-induced degradation differs across layers and experts due to non-uniform routing probabilities and expert output magnitudes. We introduce properties 2 and 3 that allow budget flexibility across and within layers.

**Property 2** (Inter-layer Budget Flexibility)**.** *Given a budget $B$ and a model with $L$ layers, a compression method should support non-uniform layer budgets $\{B_\ell\}_{\ell=1}^{L}$ satisfying $\sum_\ell B_\ell \le B$, where $B_\ell$ is not constrained to be $B/L$.*

**Property 3** (Intra-layer Budget Flexibility)**.** *For each layer $\ell$ with $N$ experts, a compression method should support non-uniform expert budgets $\{B_{\ell,i}\}_{i=1}^{N}$ satisfying $\sum_i B_{\ell,i} \le B_\ell$, where $B_{\ell,i}$ is not constrained to be $B_\ell/N$.*

Also, we require compression to scale linearly with the model size to support scalability, which is essential for practical applications.

**Property 4** (Linear Complexity)**.** *Given expert parameters $\Theta_E$, calibration data $\mathcal{D}$, and a compression function $f_\tau(\Theta_E; \mathcal{D})$, the time complexity of $f_\tau$ for generating $\widehat{\Theta}_E$ should be $O(|\mathcal{D}|\,|\Theta_E|)$.*

Lastly, it is necessary for a compression method to account for the activations induced by $\mathcal{D}$. In this regard, we introduce Property 5 that forces a compression method to reflect both the expert weights and the input distribution.

**Property 5** (Data-aware Compression)**.** *A compression method should reflect both the expert weights $\Theta_E$ and the input distribution underlying $\mathcal{D}$: if $\mathcal{D}_1 \neq \mathcal{D}_2$, $f_\tau(\Theta_E; \mathcal{D}_1) \neq f_\tau(\Theta_E; \mathcal{D}_2)$.*

# 4 Proposed Method

## 4.1 Overview

We propose **ResCode**, an accurate expert compression method. There are three main challenges that should be tackled:

**C1. One size does not fit all.** Existing methods restore all experts from a single shared base, which ignores heterogeneity among experts. How can we design a restoration structure that allocates capacity at a finer granularity?

**C2. Per-expert residual overhead.** Per-expert residuals preserve expert behavior, but their memory grows linearly with the number of experts. How can we preserve diverse expert behavior under a small memory budget?

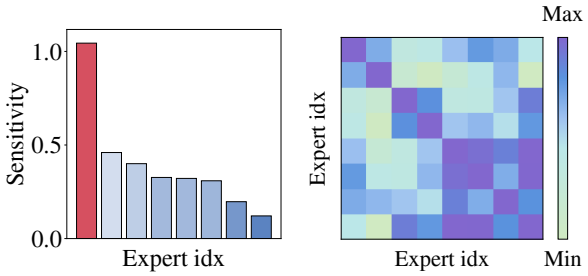
**C3. Output-misaligned restoration.** Existing methods minimize weight reconstruction error, which ignores expert behavior on actual input distributions. How can we align compression with the resulting performance degradation?

We propose three main ideas to address the aforementioned challenges:

**I1. Cluster-wise restoration (Section 4.2).** We construct expert clusters along saliency and output similarity, and restore each cluster instead of a single base for all experts.

**I2. Diagonal residual-code (Section 4.3).** We introduce per-expert diagonal codes to replace dense residuals, so that residual memory no longer scales with the number of experts.

**I3. Output-guided fitting (Section 4.4).** We fit the compressed experts with a routed output




(a) Sensitivity (b) Output similarity

Figure 4: (a) Expert sensitivity and (b) output similarity of Mixtral 8×7B. Sensitivity varies largely across experts and similarity exposes groups of similar experts.

reconstruction objective, enabling the residual parameters to recover output behavior that is not captured by static weight reconstruction.

Figure 3 depicts the overall procedure of RESCODE. Given a pre-trained SMoE layer, we first identify salient experts, and group output-similar redundant experts into shared restoration bases. We then represent expert-specific corrections with diagonal residual codes, reducing the cost of storing separate residuals. Finally, RESCODE fits the residual codes on calibration data to recover the original routed outputs. We formulate the algorithm of RESCODE in Algorithm 1.

### 4.2 Cluster-wise Restoration

**Observation.** We present an empirical observation that expert sensitivity varies significantly across experts, making single-base restoration suboptimal. We prepare a full-precision Mixtral 8×7B model and drop each expert within a layer while keeping other components unchanged to isolate its effect. Figure 4a shows that expert sensitivity varies substantially across experts, while Figure 4b reveals that output similarity exposes groups of similar experts. However, existing residual restoration methods (Ai et al., 2025) uniformly compress all experts, ignoring sensitivity and output similarity. A single shared base differs substantially from sensitive experts, leaving a large parameter gap that a low-rank residual struggles to approximate.

**Solution.** Motivated from the observation, RESCODE partitions experts into clusters by saliency and output similarity, restoring each cluster from its own base. Each cluster-wise base mitigates the difference from its experts, reducing the burden on the low-rank residual. We design a three-step scheme that 1) protects salient experts as singleton clusters, 2) groups the remaining ones by output similarity, and 3) constructs a saliency-weighted base for each cluster.

**Step 1. Protecting salient experts.** Our goal is to construct a shared base close to the experts it represents. For this goal, experts with the largest contributions to the layer output should not be merged into a shared base, since any base would differ substantially from them. RESCODE estimates expert saliency on calibration data and keeps the top-$\rho$ fraction of experts in each layer as singleton clusters, with saliency:

$$a_i = \mathbb{E}_{\mathbf{x}} \left[ \mathbf{1}(i \in T_k(\mathbf{x}))\, \pi_i(\mathbf{x})\, \|E_i(\mathbf{x})\|_2 \right], \quad (2)$$

where $\mathbf{1}(\cdot)$ is the indicator function and $T_k(\mathbf{x})$, $\pi_i(\mathbf{x})$, $E_i(\mathbf{x})$ are defined in Section 2. The singleton ratio $\rho$ is a hyperparameter that controls how many salient experts are protected per layer; we use $\rho = 0.15$ in our experiments (see Appendices E and F for sensitivity analyses and selection guidance).

**Step 2. Similarity-based grouping.** Once salient experts are isolated, the remaining ones should be grouped to share common bases with small output deviation. Experts with similar outputs are placed in one cluster, while dissimilar ones are kept separate. Following HC-SMoE (Chen et al., 2025a), RESCODE groups the non-protected experts by output distance $dist_{\text{out}}(i, j)$ on calibration data $\mathcal{D}$:

$$dist_{\text{out}}(i, j) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \|E_i(\mathbf{x}) - E_j(\mathbf{x})\|_2^2.$$

Starting from each expert as its own cluster, RESCODE repeatedly merges the two closest clusters until the target number of clusters is reached.

**Step 3. Saliency-aware base construction.** Our goal is to build one base per cluster that reflects the routed-output contributions of its experts. A uniform cluster average is suboptimal, since experts in the same cluster differ in their contributions to the layer output. RESCODE builds the base $\mathbf{S}_{\mathcal{G}}$ of each non-singleton cluster $\mathcal{G}$ by saliency-weighted averaging as follows:

$$\mathbf{S}_{\mathcal{G}} = \sum_{i \in \mathcal{G}} \alpha_i^{\mathcal{G}} \mathbf{W}_i, \quad \alpha_i^{\mathcal{G}} = \frac{a_i + \delta}{\sum_{j \in \mathcal{G}} (a_j + \delta)}, \quad (3)$$

where a small $\delta$ value prevents zero merge weights.

### 4.3 Diagonal Residual-code

**Observation.** Previous residual restoration methods restore each expert with an independent residual component, although the residuals across experts are not fully independent. For each expert

Table 2: Residual memory of ResMoE (Ai et al., 2025) and RESCODE. RESCODE reduces the dependence from $N$ experts to $C$ clusters.

| Method | Residual memory |
|---|---|
| ResMoE (2025) | $O\big(r(d_{\text{in}} + d_{\text{out}})\, N\big)$ |
| **RESCODE (Proposed)** | $O\big(r(d_{\text{in}} + d_{\text{out}})\, C\big)$ |

$i$ among $N$ experts, the expert-specific residual is $\mathbf{U}_i = \mathbf{W}_i - \mathbf{S}$, where $\mathbf{S}$ is a single shared base used by all experts within the same layer. This induces the additional memory cost, which linearly scales with the number of experts. Table 2 compares the residual memory of ResMoE and RESCODE on an MoE layer with $N$ experts and $C$ clusters, where $r$ is the residual rank and $d_{\text{in}}, d_{\text{out}}$ are the input and output dimensions of the expert weight matrices. ResMoE stores residual per expert, so its memory scales linearly with $N$, which limits compression as the number of experts grows.

**Solution.** To mitigate this memory overhead, RESCODE factorizes per-expert residuals within each cluster into shared bases and diagonal codes. Whereas ResMoE shares a single base $\mathbf{S}$ across all experts, RESCODE maintains a distinct base $\mathbf{S}_{\mathcal{G}}$ per cluster $\mathcal{G}$ (Eq. (3)), so the expert-specific residual for expert $i \in \mathcal{G}$ is $\mathbf{U}_i = \mathbf{W}_i - \mathbf{S}_{\mathcal{G}}$.

The residuals within the same cluster are not fully independent, as experts in same cluster already share similar output behavior. RESCODE exploits redundancy between residuals by coalescing expert-wise residuals into cluster-shared low-rank bases and expert-specific diagonal codes:

$$\mathbf{U}_i \approx \mathbf{A}_{\mathcal{G}}\ \mathrm{diag}(\mathbf{z}_i)\, \mathbf{B}_{\mathcal{G}}, \tag{4}$$

where $\mathbf{A}_{\mathcal{G}}, \mathbf{B}_{\mathcal{G}}$ are low-rank bases shared by cluster $\mathcal{G}$, and $\mathbf{z}_i$ is the compact code of expert $E_i$. Eq. (4) stores common residual directions per cluster and represents each expert identity with a small code vector. The compressed expert weight is then

$$\hat{\mathbf{W}}_i = \mathbf{S}_{\mathcal{G}} + \mathbf{A}_{\mathcal{G}}\ \mathrm{diag}(\mathbf{z}_i)\, \mathbf{B}_{\mathcal{G}}. \tag{5}$$

The compressed expert output is then $\hat{E}_i(\mathbf{x})$, computed by passing $\mathbf{x}$ through the SMoE expert function with weight $\hat{\mathbf{W}}_i$ replacing $\mathbf{W}_i$ in Section 2. At inference, $\mathrm{diag}(\mathbf{z}_i)$ is never materialized; the residual is applied as two matrix multiplications and an element-wise product (see Appendix D).

### 4.4 Output-guided Fitting

**Observation.** Existing methods mainly approximate expert components in weight space. This objective is misaligned with the actual MoE error, which is determined by router-weighted expert outputs after activations, nonlinearities, and projection interactions. Minimizing weight reconstruction error does not directly preserve final outputs.

**Solution.** To align the compression with the actual MoE error, RESCODE fits the residual bases and codes by directly matching the original routed outputs on calibration data. RESCODE jointly minimizes the output reconstruction loss $\mathcal{L}_{\text{rec}}$ together with language modeling loss $\mathcal{L}_{\text{LM}}$:

$$\mathcal{L}_{\text{RESCODE}} = \mathcal{L}_{\text{rec}} + \lambda\, \mathcal{L}_{\text{LM}},$$

where $\lambda$ is a balancing hyperparameter. The output reconstruction loss averages the per-layer output mismatch over $L$ MoE layers and calibration data:

$$\mathcal{L}_{\text{rec}} = \frac{1}{|\mathcal{D}|\, L} \sum_{\mathbf{x} \in \mathcal{D}} \sum_{\ell=1}^{L} \|\hat{y}^{\ell}(\mathbf{x}) - y^{\ell}(\mathbf{x})\|_2^2,$$

where $y^{\ell}(\mathbf{x})$ is the original layer-$\ell$ output defined in Eq. (1) and $\hat{y}^{\ell}(\mathbf{x})$ is its compressed counterpart obtained by replacing $E_i(\mathbf{x})$ with $\hat{E}_i(\mathbf{x})$. As RESCODE preserves the original router and top-$k$ selection (Property 1), each layer output gap reduces to a weighted sum of expert output gaps:

$$\hat{y}(\mathbf{x}) - y(\mathbf{x}) = \sum_{i \in T_k(\mathbf{x})} \pi_i(\mathbf{x}) \big(\hat{E}_i(\mathbf{x}) - E_i(\mathbf{x})\big).$$

During calibration, all non-residual modules are frozen, and only $\mathbf{A}_{\mathcal{G}}$, $\mathbf{B}_{\mathcal{G}}$, and $\mathbf{z}_i$ are trained. The calibration is label-free: $\mathcal{L}_{\text{rec}}$ matches teacher outputs from the original model, while $\mathcal{L}_{\text{LM}}$ is guided by next-token targets.

### 4.5 Complexity Analysis

We analyze the time complexity of RESCODE, where $|\mathcal{D}|$ is the number of calibration samples and $L$ is the number of MoE layers in the model.

**Theorem 1** (Time Complexity of RESCODE)**.** *Given a model with per-layer inference cost $T_\theta$, the time complexity for the compression procedure (Algorithm 1) of* RESCODE *is $O\big(|\mathcal{D}|\, L\, T_\theta\big)$.*

*Proof.* See Appendix D. □

Theorem 1 demonstrates that RESCODE offers an efficient solution, with time complexity growing linearly in the numbers $|\mathcal{D}|$ of calibration samples and $L$ of MoE layers (see Section 5.3).

Table 3: Zero-shot accuracy of compressed Qwen1.5-MoE-A2.7B (Qwen Team, 2024) and Mixtral 8×7B (Jiang et al., 2024) models across benchmark tasks. Bold numbers denote the best result among compressed methods at the same compression ratio. RESCODE outperforms existing methods on various tasks.

| Model | Ratio | Method | Type* | ARC-c | ARC-e | BoolQ | HSwag | MMLU | OBQA | RTE | WinoG | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Qwen1.5-MoE-A2.7B | Baseline | - | - | 0.3951 | 0.7012 | 0.8135 | 0.5932 | 0.6047 | 0.3100 | 0.7329 | 0.6559 | 0.6008 |
| | 25% | S-prune (2025) | P | 0.3464 | 0.6061 | 0.7128 | 0.5228 | 0.4930 | 0.2640 | 0.6534 | 0.5935 | 0.5240 |
| | | REAP (2026) | P | 0.3737 | 0.6789 | 0.7425 | **0.5789** | 0.4981 | 0.2840 | 0.6245 | 0.6417 | 0.5528 |
| | | M-SMoE (2024b) | M | 0.3473 | 0.6157 | 0.7544 | 0.5157 | 0.4182 | 0.2620 | 0.7292 | 0.6377 | 0.5350 |
| | | HC-SMoE (2025a) | M | 0.3660 | 0.6578 | **0.7948** | 0.5520 | 0.5332 | 0.2720 | **0.7509** | 0.6464 | 0.5716 |
| | | ResMoE (2025) | D | 0.3814 | 0.6730 | 0.7829 | 0.5232 | **0.5405** | 0.2660 | 0.7292 | 0.6725 | 0.5711 |
| | | **RESCODE (Proposed)** | D | **0.3985** | **0.7071** | 0.7924 | 0.5513 | 0.5387 | **0.2880** | 0.7401 | **0.6867** | **0.5878** |
| | 50% | S-prune (2025) | P | 0.2500 | 0.4756 | 0.6388 | 0.4041 | 0.3471 | 0.1960 | 0.6209 | 0.5146 | 0.4309 |
| | | REAP (2026) | P | 0.3089 | 0.5619 | 0.6572 | 0.4922 | 0.3232 | 0.2500 | 0.6029 | 0.5951 | 0.4739 |
| | | M-SMoE (2024b) | M | 0.1945 | 0.2786 | 0.4462 | 0.2837 | 0.2475 | 0.1600 | 0.4477 | 0.5185 | 0.3221 |
| | | HC-SMoE (2025a) | M | 0.3532 | 0.6149 | 0.7535 | 0.4695 | 0.4534 | 0.2280 | 0.6606 | 0.6456 | 0.5223 |
| | | ResMoE (2025) | D | 0.2944 | 0.5362 | 0.5229 | 0.3836 | 0.2312 | 0.2240 | 0.5740 | 0.5935 | 0.4200 |
| | | **RESCODE (Proposed)** | D | **0.3669** | **0.6473** | **0.7596** | **0.4950** | **0.4708** | **0.2800** | **0.6895** | **0.6638** | **0.5466** |
| Mixtral 8×7B | Baseline | - | - | 0.5717 | 0.8409 | 0.8529 | 0.6492 | 0.6790 | 0.3520 | 0.7076 | 0.7640 | 0.6772 |
| | 25% | S-prune (2025) | P | 0.4991 | 0.7891 | 0.7801 | 0.5984 | 0.5103 | **0.3400** | 0.5704 | **0.7735** | 0.6076 |
| | | REAP (2026) | P | 0.4940 | 0.7887 | 0.8401 | 0.6181 | 0.6119 | 0.3360 | **0.7220** | 0.7656 | 0.6471 |
| | | M-SMoE (2024b) | M | 0.2619 | 0.5564 | 0.5208 | 0.4320 | 0.2503 | 0.1940 | 0.5271 | 0.5848 | 0.4159 |
| | | HC-SMoE (2025a) | M | **0.5145** | **0.8043** | **0.8554** | 0.6142 | 0.6043 | 0.3240 | 0.6715 | 0.7514 | 0.6425 |
| | | ResMoE (2025) | D | 0.4744 | 0.7618 | 0.8190 | 0.5371 | 0.5662 | 0.2980 | 0.6787 | 0.7190 | 0.6068 |
| | | **RESCODE (Proposed)** | D | 0.5017 | 0.7992 | 0.8462 | **0.6249** | **0.6184** | 0.3320 | 0.7112 | 0.7545 | **0.6485** |
| | 50% | S-prune (2025) | P | 0.2235 | 0.4339 | 0.6300 | 0.4250 | 0.2554 | 0.1880 | 0.5235 | 0.5699 | 0.4062 |
| | | REAP (2026) | P | 0.4667 | 0.7576 | 0.8046 | 0.5629 | 0.4932 | 0.2980 | **0.6931** | 0.7127 | 0.5986 |
| | | M-SMoE (2024b) | M | 0.2116 | 0.2765 | 0.4954 | 0.2767 | 0.2452 | 0.1080 | 0.4910 | 0.4964 | 0.3251 |
| | | HC-SMoE (2025a) | M | 0.4573 | 0.7454 | 0.8018 | 0.5709 | 0.4571 | 0.2700 | 0.5523 | 0.7285 | 0.5729 |
| | | ResMoE (2025) | D | 0.3072 | 0.6115 | 0.6590 | 0.3167 | 0.3408 | 0.2020 | 0.5451 | 0.5975 | 0.4475 |
| | | **RESCODE (Proposed)** | D | **0.4863** | **0.7677** | **0.8343** | **0.5993** | **0.5330** | **0.3040** | 0.6282 | **0.7238** | **0.6096** |

* P: Pruning, M: Merging, D: Decomposition.

## 5 Experiments

We perform experiments to answer the following questions about RESCODE.

**Q1. Performance comparison** (Section 5.2). How consistently does RESCODE outperform existing expert compression methods across different MoE architectures and evaluation tasks?

**Q2. Runtime analysis** (Section 5.3). How does RESCODE improve the compression runtime–accuracy trade-off compared to existing expert compression methods?

**Q3. Bit allocation** (Section 5.4). How does RESCODE allocate bit budgets across experts and layers to improve the accuracy of the compressed MoE model?

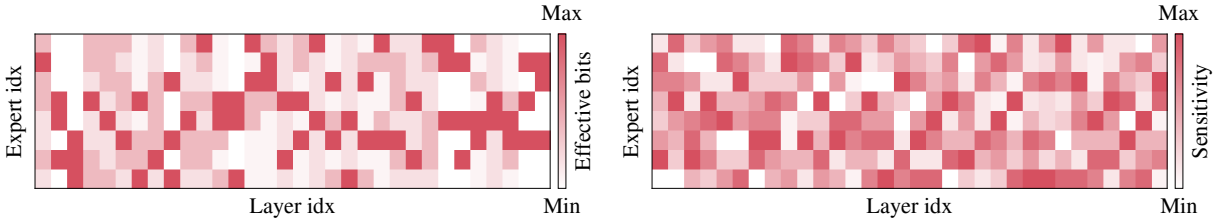
**Q4. Calibration dataset** (Section 5.5). How much does the evaluation performance of RESCODE vary across different calibration datasets?

**Q5. Ablation study** (Section 5.6). How does each of the ideas of RESCODE affect the accuracy of the compressed MoE model?

**Q6. Hyperparameter analysis** (Section 5.7). How sensitive is RESCODE to the residual rank and singleton ratio?

### 5.1 Experimental Setup

We briefly introduce our experimental setup. Further details are provided in Appendix E. We evaluate RESCODE on three open-source SMoE LLMs with distinct routing configurations: Qwen1.5-MoE-A2.7B-Chat (Qwen Team, 2024) (60 experts + 1 shared, top-4), Mixtral-8×7B-v0.1 (Jiang et al., 2024) (8 experts, top-2), and DeepSeek-MoE-16B (Dai et al., 2024) (64 experts + 2 shared, top-6). The calibration dataset is DCLM (Li et al., 2024a) for default, where we evaluate zero-shot performance on eight tasks. We compare RESCODE with two pruning (S-prune (He et al., 2025), REAP (Lasby et al., 2026)), two merging (M-SMoE (Li et al., 2024b), HC-SMoE (Chen et al., 2025a)) and one residual restoration (ResMoE (Ai et al., 2025)) baselines.




(a) Bit allocation (b) Sensitivity

Figure 5: Heatmap of (a) bit allocation by RESCODE and (b) sensitivity on Mixtral 8×7B at 50% compression ratio. RESCODE performs sensitivity-aware bit allocation as intended.

Table 4: Accuracy and compression runtime comparison between ResMoE and RESCODE.

| Method | Accuracy | Time (min) |
|---|---|---|
| ResMoE | 0.4475 | 83.2 |
| **RESCODE (Proposed)** | **0.6096** | **59.5** |

### 5.2 Performance Comparison (Q1)

We evaluate the accuracy of SMoE models compressed by RESCODE against existing methods in Table 3. RESCODE consistently improves compressed models across both Qwen1.5-MoE-A2.7B-Chat and Mixtral-8×7B-v0.1, achieving up to 2.43 pp higher average accuracy. Notably, RESCODE becomes increasingly effective as the compression ratio increases, highlighting its robustness in challenging high-compression settings. Repeating the complete compression pipeline with three independent calibration-sampling and optimization seeds yields average accuracies of $0.6095 \pm 0.0035$ on Mixtral and $0.5446 \pm 0.0058$ on Qwen at 50% expert-memory reduction, i.e., mean improvements of 1.09 pp and 2.23 pp over the strongest baselines (per-task results in Appendix F). On DeepSeek-MoE-16B, RESCODE achieves average accuracy of 0.5018, outperforming REAP, the strongest baseline, by 1.68 pp (Table 9 in Appendix F).

### 5.3 Runtime analysis (Q2)

We perform a runtime analysis to examine the computational overhead by RESCODE. Table 4 compares the compression time and accuracy of Mixtral 8×7B model at 50% compression. RESCODE runs 28.5% faster than existing methods while providing higher accuracy, supporting its efficiency and scalability with the linearity shown in Theorem 1. Table 8 in Appendix D benchmarks deployment on a single NVIDIA B200 GPU with standard PyTorch operations. At 50% expert-memory reduction, RESCODE reduces the resident GPU memory of BF16 Mixtral-8×7B by 46.3% (from 87.0 GB to 46.7 GB) while running 2.1× and 2.9× faster than ResMoE-SVD in decoding and prefill, respectively.

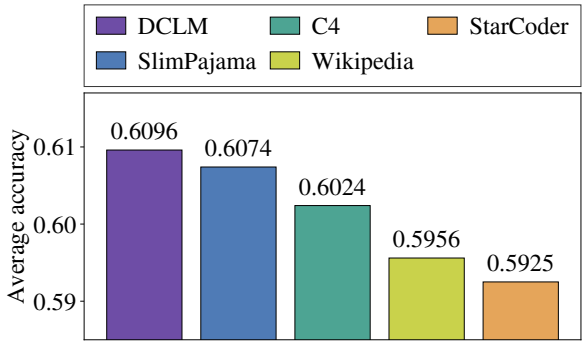



Figure 6: Accuracy across various calibration dataset on Mixtral 8×7B at 50% compression ratio. DCLM (Li et al., 2024a) dataset yields the best accuracy.

### 5.4 Bit allocation (Q3)

We conduct a case study to examine how RESCODE allocates bits and whether it reflects expert-level sensitivity. Figure 5 shows the expert-wise effective bit-widths and sensitivity of Mixtral 8×7B at 50% compression. The close resemblance between the two heatmap distributions shows that RESCODE successfully allocates higher bits to salient experts while assigning lower bits to less salient ones.

### 5.5 Calibration dataset (Q4)

We investigate how the selection of calibration datasets affects the performance of RESCODE. Figure 6 reports the average accuracy of RESCODE on Mixtral 8×7B at 50% compression ratio. RESCODE achieves the best accuracy of 0.6096 with DCLM, outperforming C4 and StarCoder by 0.72 pp and 1.71 pp, respectively. These results show that representative calibration data are important for accurate output recovery. The corpus ranking is identical on Qwen1.5-MoE, where DCLM, SlimPajama, C4, Wikipedia, and StarCoder yield 0.5424, 0.5369, 0.5363, 0.5291, and 0.4967 in a separate single-seed run. We use 2,080 calibration sequences, and enlarging the calibration set

Table 5: Ablation study of RESCODE components on Mixtral 8×7B at 50% expert-memory reduction with DCLM calibration; all variants share the cluster assignments and memory budget. Base is identical to the ResMoE run in Table 3. See Section 5.6 for details.

| Method | I1 | I2 | I3 | Accuracy |
|---|---|---|---|---|
| Base: ResMoE | | | | 0.4475 |
| Base + I1 | ✓ | | | 0.5862 |
| Base + I3 | | | ✓ | 0.5847 |
| Base + I1 + I2 | ✓ | ✓ | | 0.5885 |
| Base + I1 + I3 | ✓ | | ✓ | 0.6015 |
| **RESCODE (Proposed)** | ✓ | ✓ | ✓ | **0.6076** |

to 9,600 sequences brings no further gain (Appendix F).

### 5.6 Ablation study (Q5)

We perform an ablation study to validate the effectiveness of each main component of RESCODE. Table 5 summarizes the results on Mixtral 8×7B at 50% expert-memory reduction with DCLM calibration. Cluster-wise restoration (I1) improves the accuracy by 13.87 pp over the baseline, showing its importance for preserving salient experts. Diagonal residual-code (I2) further improves the compressed model by preserving the higher rank residual, whereas the per-expert residuals of ResMoE are truncated to a much lower rank under the same memory budget. Output-guided fitting (I3) provides a further gain, and the full RESCODE achieves the best accuracy, validating the complementary benefit of the three ideas. The comparison with ResMoE under identical output-guided fitting is provided in Appendix F (Table 10).

### 5.7 Hyperparameter analysis (Q6)

We analyze the sensitivity of RESCODE to its hyperparameters and derive practical selection rules. Table 6 varies the residual rank $r$ with the clustering configuration fixed; the default $r = 128$ performs the best on both models. We choose the singleton ratio $\rho$ by compression ratio, using larger values under mild compression and smaller ones under aggressive compression. Figure 7 reports representative sweeps of the singleton ratio $\rho$. Protecting more salient experts improves accuracy under mild compression (Qwen at 25%), whereas a smaller ratio performs better under aggressive compression (Mixtral at 50%).

Table 6: Sensitivity to the residual rank $r$ at 50% expert-memory reduction with fixed clustering. The default $r = 128$ performs the best on both models.

| Model | $r = 16$ | $r = 32$ | $r = 64$ | $r = 128$ | $r = 256$ |
|---|---|---|---|---|---|
| Qwen | 0.5376 | 0.5368 | 0.5394 | **0.5466** | 0.5455 |
| Mixtral | 0.5940 | 0.5950 | 0.5992 | **0.6096** | 0.5936 |

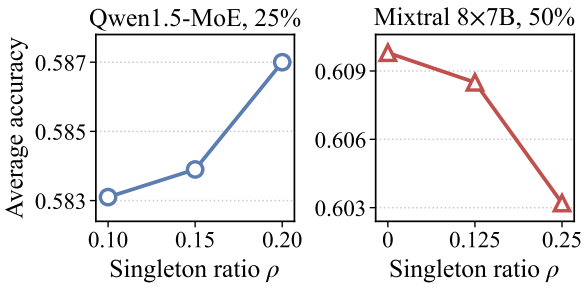



Figure 7: Sensitivity to the singleton ratio $\rho$ under mild (Qwen at 25%) and aggressive (Mixtral at 50%) compression. See Appendix F for detailed results.

## 6 Conclusion

We propose RESCODE, an accurate and efficient expert compression method for SMoE models. RESCODE addresses the limitations of existing merging, pruning, and residual restoration methods by combining cluster-wise restoration, diagonal residual-code, and output-guided calibration fitting. This design preserves routing diversity, protects salient experts, and recovers routed outputs with compact residual codes. Experiments show that RESCODE provides a practical compression–performance trade-off for deploying SMoE models under limited memory budgets. As future work, we plan to extend RESCODE to frontier-scale SMoE models, multilingual/long-form generation settings, and downstream adaptation, and to combine it with quantization for further memory savings.


## Acknowledgments

This work was supported by System LSI (S.LSI) Business, Samsung Electronics Co., Ltd. U Kang is the corresponding author.


## Limitations

RESCODE focuses on reducing expert memory while preserving the behavior of a pretrained SMoE model, but it has several limitations. First, it requires calibration data to estimate expert similarity, saliency, and output reconstruction targets, so the quality and coverage of calibration samples may affect the final compressed model. Second, the compression-performance trade-off depends on

hyperparameters such as cluster size, residual rank, and the choice of restored projection branches; these hyperparameters interact, and we provide sensitivity analyses and selection rules (Section 5.7 and Appendices E and F). Third, RESCODE primarily targets expert-parameter memory and does not directly compress non-expert components such as attention layers and routers; at 50% expert-memory reduction, the end-to-end resident memory of Mixtral-8×7B decreases by 46.3% (from 87.0 GB to 46.7 GB; Section 5.3). Finally, although we evaluate RESCODE on representative SMoE models and benchmarks, broader validation across larger models, multilingual settings, and downstream adaptation scenarios remains as future work.

## Ethics Statement

This work proposes a post-training compression method for pretrained sparse mixture-of-experts (SMoE) language models, and does not involve new data collection, human subjects, or training from scratch. All models and corpora used in our experiments are publicly released artifacts, and we use them consistently with their respective licenses and intended research use. RESCODE reduces the memory footprint of SMoE experts and is intended to make their deployment more accessible on resource-limited hardware. We note that compression does not remove the biases or potential harms already present in the underlying pretrained models, and recommend that practitioners apply the same safety and factuality evaluations as for the original models. We used AI assistants solely for polishing the language of the manuscript.

## References


Mengting Ai, Tianxin Wei, Yifan Chen, Zhichen Zeng, Ritchie Zhao, Girish Varatkar, Bita Darvish Rouhani, Xianfeng Tang, Hanghang Tong, and Jingrui He. 2025. ResMoE: Space-efficient compression of mixture of experts LLMs via residual restoration. In *KDD*.

Sikai Bai, Haoxi Li, Jie Zhang, Zicong Hong, and Song Guo. 2025. DiEP: Adaptive mixture-of-experts compression through differentiable expert pruning. In *NeurIPS*.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth PASCAL recognizing textual entailment challenge. In *TAC*.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. 2025. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*.

I-Chun Chen, Hsu-Shen Liu, Wei-Fang Sun, Chen-Hao Chao, Yen-Chang Hsu, and Chun-Yi Lee. 2025a. Retraining-free merging of sparse MoE via hierarchical clustering. In *ICML*.

Tianyu Chen, Shaohan Huang, Yuan Xie, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. 2022. Task-specific expert pruning for sparse mixture-of-experts. *arXiv preprint arXiv:2206.00277*.

Zhixuan Chen, Xing Hu, Dawei Yang, Zukang Xu, Chen Xu, Zhihang Yuan, Sifan Zhou, and Jiangyong Yu. 2025b. MoEQuant: Enhancing quantization for mixture-of-experts large language models via expert-balanced sampling and affinity guidance. In *ICML*.

Ikhyun Cho and U Kang. 2022. Pea-KD: Parameter-efficient and accurate knowledge distillation on BERT. *PLOS ONE*.

Wonjin Cho, Jeongin Yun, and U Kang. 2026. Accurate zero-shot quantization via hierarchical teacher-assistant distillation. In *ECCV*.

Mohammed Nowaz Rabbani Chowdhury, Kaoutar El Maghraoui, Hsinyu Tsai, Naigang Wang, Geoffrey W. Burr, Liu Liu, and Meng Wang. 2026. Efficient quantization of mixture-of-experts with theoretical generalization guarantees. In *ICLR*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

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. In *ACL*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *JMLR*.

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. 2023. A framework for few-shot language model evaluation. Zenodo.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *ICLR*.

Beichen Huang, Yueming Yuan, Zelei Shao, and Minjia Zhang. 2025a. MiLo: Efficient quantized MoE inference with mixture of low-rank compensators. In *MLSys*.

Wei Huang, Yue Liao, Jianhui Liu, Ruifei He, Haoru Tan, Shiming Zhang, Hongsheng Li, Si Liu, and Xiaojuan Qi. 2025b. Mixture compressor for mixture-of-experts LLMs gains more. In *ICLR*.

Jun-Gi Jang, Chun Quan, Hyun Dong Lee, and U Kang. 2023. Falcon: lightweight and accurate convolution based on depthwise separable convolution. *Knowledge and Information Systems*.

Hyojin Jeon, Seungcheol Park, Jin-Gee Kim, and U Kang. 2023. PET: Parameter-efficient knowledge distillation on transformer. *PLOS ONE*.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Junghun Kim, Jinhong Jung, and U Kang. 2021. Compressing deep graph convolution network with multi-staged knowledge distillation. *PLOS ONE*.

Minjun Kim, Jaehyeon Choi, Jongkeun Lee, Wonjin Cho, and U Kang. 2025a. Zero-shot quantization: A comprehensive survey. In *IJCAI*.

Minjun Kim, Jaehyeon Choi, Hyunwoo Yang, Jongjin Kim, Jinho Song, and U Kang. 2026a. Prune-then-quantize or quantize-then-prune? understanding the impact of compression order in joint model compression. In *ICLR*.

Minjun Kim, Jongjin Kim, and U Kang. 2025b. SynQ: Accurate zero-shot quantization by synthesis-aware fine-tuning. In *ICLR*.

Minjun Kim, Jaeri Lee, Jongjin Kim, Jeongin Yun, Yongmo Kwon, and U Kang. 2026b. LampQ: Towards accurate layer-wise mixed precision quantization for vision transformers. In *AAAI*.

Mike Lasby, Ivan Lazarevich, Nish Sinnadurai, Sean Lie, Yani Ioannou, and Vithursan Thangarasa. 2026. REAP the experts: Why pruning prevails for one-shot MoE compression. In *ICLR*.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, and 1 others. 2024a. DataComp-LM: In search of the next generation of training sets for language models. In *NeurIPS*.

Lujun Li, Qiyuan Zhu, Jiacheng Wang, Xiaoyu Qin, Wei Li, Hao Gu, Sirui Han, and Yike Guo. 2026. SubMoE: Efficient mixture-of-expert LLMs compression via subspace expert merging. In *AAAI*.

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

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, and 1 others. 2023. StarCoder: May the source be with you! *TMLR*.

Wei Li, Lujun Li, Hao Gu, You-Liang Huang, Mark G. Lee, Shengjie Sun, Wei Xue, and Yike Guo. 2025. MoE-SVD: Structured mixture-of-experts LLMs compression via singular value decomposition. In *ICML*.

Boan Liu, Liang Ding, Li Shen, Keqin Peng, Yu Cao, Dazhao Cheng, and Dacheng Tao. 2024. Diversifying the mixture-of-experts representation for language models with orthogonal optimizer. In *ECAI*.

Jiacheng Liu, Peng Tang, Wenfeng Wang, Yuhang Ren, Xiaofeng Hou, Pheng Ann Heng, Minyi Guo, and Chao Li. 2026. A survey on inference optimization techniques for mixture of experts models. *ACM Computing Surveys*, 58(10):1–37.

Jan Ludziejewski, Maciej Pióro, Jakub Krajewski, Maciej Stefaniak, Michał Krutul, Jan Małaśnicki, Marek Cygan, Piotr Sankowski, Kamil Adamczewski, Piotr Miłoś, and Sebastian Jaszczur. 2025. Joint MoE scaling laws: Mixture of experts can be memory efficient. In *ICML*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Seungcheol Park, Jeongin Bae, Beomseok Kwon, Minjun Kim, Byeongwook Kim, Se Jung Kwon, U Kang, and Dongsoo Lee. 2025a. Unifying uniform and binary-coding quantization for accurate compression of large language models. In *ACL*.

Seungcheol Park, Hojun Choi, and U Kang. 2024. Accurate retraining-free pruning for pretrained encoder-based language models. In *ICLR*.

Seungcheol Park, Jaehyeon Choi, Sojin Lee, and U Kang. 2026. A comprehensive survey of compression algorithms for language models. *ACM Computing Surveys*.

Seungcheol Park, Sojin Lee, Jongjin Kim, Jinsik Lee, Hyunjik Jo, and U Kang. 2025b. Accurate sublayer pruning for large language models by exploiting latency and tunability information. In *IJCAI*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, and 1 others. 2019. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*.

Tairen Piao, Ikhyun Cho, and U Kang. 2022. SensiMix: Sensitivity-aware 8-bit index & 1-bit value mixed precision quantization for BERT compression. *PLOS ONE*.

Qwen Team. 2024. Qwen1.5-MoE: Matching 7B model performance with 1/3 activated parameters. Blog post.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. WinoGrande: An adversarial winograd schema challenge at scale. *CACM*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R. Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

Wikimedia Foundation. 2024. Wikipedia: The free encyclopedia.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, and 1 others. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP System Demonstrations*.

Zhanhao Xie, Yuexiao Ma, Xiawu Zheng, Fei Chao, Wanchen Sui, Yong Li, Shen Li, and Rongrong Ji. 2025. Automated fine-grained mixture-of-experts quantization. In *ACL Findings*.

Cheng Yang, Yang Sui, Jinqi Xiao, Lingyi Huang, Yu Gong, Yuanlin Duan, Wenqi Jia, Miao Yin, Yu Cheng, and Bo Yuan. 2024. MoE-I$^2$: Compressing mixture of experts models through inter-expert pruning and intra-expert low-rank decomposition. In *EMNLP Findings*.

Jaemin Yoo, Minyong Cho, Taebum Kim, and U Kang. 2019. Knowledge extraction with no observable data. In *NeurIPS*.

Jeongin Yun, Jaeri Lee, Jongjin Kim, Minjun Kim, Jinho Song, and U Kang. 2026. SharVeT: Similarity-aware parameter sharing with vector-based tuning for efficient LLM compression. In *ACL*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *ACL*.

# Appendix

## A Notation

Table 7 summarizes the general notation used throughout the paper. Unless otherwise stated, we describe a single MoE layer and omit the layer index $\ell$ for readability.

Table 7: Frequently used notations.

| Symbol | Description |
|---|---|
| $L$ | Number of MoE layers. |
| $N$ | Number of original experts per MoE layer. |
| $C$ | Number of compressed expert groups, $C < N$. |
| $k$ | Number of experts selected by top-$k$ routing. |
| $\mathcal{G}$ | Expert cluster (a set of expert indices). |
| $\lvert\mathcal{G}\rvert$ | Number of experts within cluster $\mathcal{G}$. |
| $d_{\text{in}}, d_{\text{out}}$ | Expert input and output dimensions. |
| $\mathcal{D}$ | Calibration data. |
| $E_i^\ell$ | Expert in layer $\ell$. |
| $\hat{E}_i^\ell$ | Compressed expert in layer $\ell$. |
| $\mathbf{x}$ | Input token representation. |
| $y^\ell(\mathbf{x})$ | Original SMoE-layer output. |
| $\hat{y}^\ell(\mathbf{x})$ | Compressed SMoE-layer output. |
| $\phi$ | Router. |
| $T_k(\mathbf{x})$ | Top-$k$ expert set. |
| $\pi_i(\mathbf{x})$ | Routing weight of expert $i$. |
| $\tau$ | Target expert-memory ratio. |
| $\mathbf{S}_\mathcal{G}$ | Shared component for cluster $\mathcal{G}$. |
| $\mathbf{U}_i$ | Expert-specific component for expert $E_i$. |
| $\mathbf{A}_\mathcal{G}, \mathbf{B}_\mathcal{G}$ | Low-rank bases shared by cluster $\mathcal{G}$. |
| $\mathbf{z}_i$ | Compact code of expert $E_i$. |
| $a_i$ | Saliency of each expert $E_i$. |
| $\mathcal{L}_{\text{ResCode}}$ | Target loss of ResCode. |
| $\mathcal{L}_{rec}$ | Output reconstruction loss. |
| $\mathcal{L}_{LM}$ | Language modeling loss. |
| $\lambda$ | Loss balancing hyperparameter. |

## B Algorithm

We present the overall procedure of the proposed method. Algorithm 1 outlines ResCode, which compresses a pre-trained SMoE model by 1) constructing saliency-aware expert clusters with shared bases (Idea 1), 2) coalescing per-expert residuals into cluster-shared low-rank bases and diagonal codes (Idea 2), and 3) jointly fitting the residual parameters across all layers with a routed output reconstruction objective (Idea 3). Steps 1 and 2 are performed independently per MoE layer, while Step 3 jointly fits all residual parameters on calibration data $\mathcal{D}$.

**Algorithm 1** Compression procedure of RESCODE

**Input:** pre-trained SMoE model with expert parameters $\Theta_E$, calibration data $\mathcal{D}$, target ratio $\tau$, residual rank $r$, and hyperparameters $\delta$, $\lambda$, $n_{ep}$.
**Output:** compressed expert parameters $\widehat{\Theta}_E = \{\mathbf{S}_\mathcal{G},\ \mathbf{A}_\mathcal{G},\ \mathbf{B}_\mathcal{G},\ \mathbf{z}_i\}_{\ell=1}^{L}$.

**/** Step 1: Cluster-wise restoration (Idea 1) **/**

**for** each MoE layer $\ell \in [1, \dots, L]$ **do**
  Estimate saliency $a_i = \mathbb{E}_\mathbf{x}\big[\mathbf{1}(i \in T_k(\mathbf{x}))\, \pi_i(\mathbf{x})\, \|E_i(\mathbf{x})\|_2\big]$; keep top-ranked as singletons.
  Group the rest by hierarchical clustering on $dist_{\text{out}}(i,j) = \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \|E_i(\mathbf{x}) - E_j(\mathbf{x})\|_2^2$ into $\{\mathcal{G}\}$.
  Build cluster base $\mathbf{S}_\mathcal{G} = \sum_{i\in\mathcal{G}} \alpha_i^\mathcal{G}\, \mathbf{W}_i$ with $\alpha_i^\mathcal{G} = (a_i + \delta)/\sum_{j\in\mathcal{G}}(a_j + \delta)$.
**end for**

**/** Step 2: Diagonal residual-code initialization (Idea 2) **/**

**for** each cluster $\mathcal{G}$ across all MoE layers **do**
  Compute residuals $\mathbf{U}_i = \mathbf{W}_i - \mathbf{S}_\mathcal{G}$ for $i \in \mathcal{G}$.
  Initialize rank-$r$ bases $\mathbf{A}_\mathcal{G}, \mathbf{B}_\mathcal{G}$ and codes $\{\mathbf{z}_i\}$ s.t. $\mathbf{U}_i \approx \mathbf{A}_\mathcal{G}\, \mathrm{diag}(\mathbf{z}_i)\, \mathbf{B}_\mathcal{G}$.
**end for**

**/** Step 3: Output-guided fitting (Idea 3) **/**

Freeze non-residual modules; mark $\{\mathbf{A}_\mathcal{G}, \mathbf{B}_\mathcal{G}, \mathbf{z}_i\}$ as trainable.
**for** each epoch in $[1, \dots, n_{ep}]$ **do**
  $\mathcal{L} \leftarrow 0$.
  **for** each $\mathbf{x} \in \mathcal{D}$ **do**
    Cache $\{y^\ell(\mathbf{x})\}_{\ell=1}^{L}$ from the original model.
    Forward $\mathbf{x}$ through the compressed model with $\hat{\mathbf{W}}_i = \mathbf{S}_\mathcal{G} + \mathbf{A}_\mathcal{G}\, \mathrm{diag}(\mathbf{z}_i)\, \mathbf{B}_\mathcal{G}$ to get $\{\hat{y}^\ell(\mathbf{x})\}_{\ell=1}^{L}$.
    $\mathcal{L}$ += $\frac{1}{L}\sum_{\ell=1}^{L} \big\|\hat{y}^\ell(\mathbf{x}) - y^\ell(\mathbf{x})\big\|_2^2 + \lambda\, \mathcal{L}_{\text{LM}}(\mathbf{x})$.
  **end for**
  Update $\{\mathbf{A}_\mathcal{G}, \mathbf{B}_\mathcal{G}, \mathbf{z}_i\}$ to minimize $\mathcal{L}$.
**end for**
**return** $\widehat{\Theta}_E = \{\mathbf{S}_\mathcal{G},\ \mathbf{A}_\mathcal{G},\ \mathbf{B}_\mathcal{G},\ \mathbf{z}_i\}_{\ell=1}^{L}$

## C Additional Related Works

We provide an extended discussion of related works that complement the main-text coverage in Section 2. We mainly focus on lines of works that are conceptually adjacent to RESCODE but not directly compared in Section 5.

**Additional baselines.** A broader line of work targets expert parameter reduction through structured or learned pruning, merging, or decomposition. MoE-I$^2$ (Yang et al., 2024) combines inter-expert pruning with intra-expert low-rank decomposition to reduce both expert count and per-expert parameters. Task-Specific Expert Pruning (Chen et al., 2022) removes experts based on downstream task signals rather than calibration statistics, trading generality for task-level fidelity. DiEP (Bai et al., 2025) learns differentiable expert importance scores to enable adaptive pruning ratios across layers, but does not preserve the original token-to-expert assignments. Sub-MoE (Li et al., 2026) merges experts in a shared subspace rather than at the full parameter level, which reduces parameter redundancy at the cost of removing routed experts. Beyond pruning and merging, Liu et al. (2024) reduces expert redundancy at training time by diversifying expert representations with an orthogonal optimizer. Compared with these approaches, RESCODE targets the parameter geometry of each routed expert directly: it preserves the original router assignments (Property 1), exploits structured residual sharing within saliency-aware clusters, and aligns the compression objective with the layer's actual routed outputs.

**General model compression.** Beyond MoE-specific approaches, a broad line of work compresses dense models. K-prune (Park et al., 2024) performs accurate retraining-free structured pruning of encoder-based language models, and Park et al. (2025b) prunes sublayers of LLMs by exploiting latency and tunability information. Knowledge distillation offers another route to compact models, including parameter-efficient distillation of BERT and Transformers (Cho and Kang, 2022; Jeon et al., 2023) and multi-staged distillation of graph convolutional networks (Kim et al., 2021), while Falcon (Jang et al., 2023) designs lightweight convolutions based on depthwise separable convolution. SharVeT (Yun et al., 2026) compresses LLMs by similarity-aware parameter sharing with vector-based tuning, which is conceptually close to RESCODE's cluster-shared bases with expert-

specific codes, although it targets dense models rather than routed experts. Compression is also studied without observable data: Yoo et al. (2019) extracts knowledge from a trained model without accessing its training data, SynQ (Kim et al., 2025b) and ZEST (Cho et al., 2026) enable accurate zero-shot quantization via synthesis-aware fine-tuning and hierarchical teacher-assistant distillation, respectively, and Kim et al. (2025a) provide a comprehensive survey of zero-shot quantization. These directions suggest reducing the calibration-data dependence of ResCode as a promising extension.

**Expert quantization.** A complementary direction to expert compression is expert quantization, which reduces the bit-width of expert weights rather than their count or rank (Chowdhury et al., 2026). MC-MoE (Huang et al., 2025b) jointly quantizes and prunes experts under a mixed-precision allocation guided by expert importance. MoE-Quant (Chen et al., 2025b) introduces expert-balanced sampling and affinity-guided calibration to improve quantization fidelity for skewed expert activation distributions. Automated fine-grained MoE quantization (Xie et al., 2025) searches per-expert and per-layer bit configurations under a global memory budget, exposing the heterogeneity of expert sensitivity that also motivates our saliency-aware design. MiLo (Huang et al., 2025a) pairs low-bit experts with low-rank compensators to recover quantization error at inference time. These methods are orthogonal to ResCode: quantization compresses the bit-width of stored weights, whereas ResCode compresses the residual structure of expert weights via cluster-shared low-rank bases and diagonal codes. The two directions can in principle be stacked, applying quantization on top of ResCode's compressed representation for further memory savings, for example with unified uniform and binary-coding quantization (Park et al., 2025a) or sensitivity-aware mixed-precision allocation (Piao et al., 2022; Kim et al., 2026b); the ordering of such joint compression itself affects the final accuracy (Kim et al., 2026a).

**Surveys and scaling perspectives.** Several surveys provide broader context for the MoE compression problem. The MoE survey of Cai et al. (2025) reviews architectural and training-time variants of sparse mixture-of-experts models, while the inference-optimization survey of Liu et al. (2026) catalogs deployment-time techniques including expert offloading, caching, and routing acceleration. Park et al. (2026) comprehensively review compression algorithms for language models. From a scaling perspective, Ludziejewski et al. (2025) establishes joint MoE scaling laws showing that under fixed compute budgets, increasing the expert count and total parameter memory is often the most efficient design choice. This observation reinforces the motivation behind ResCode: as practical MoE deployment continues to scale up the expert dimension, expert-side memory becomes the dominant bottleneck, making accurate expert compression an increasingly central problem.

## D Deployment Practicality

We provide the proof of Theorem 1 and detail the practicality criterion of deployment discussed in Section 4.5. We distinguish practical compression from parameter-count reduction alone: a compressed expert structure should be inexpensive to construct, regular in memory layout, and compatible with standard dense kernels.

### D.1 Proof of Theorem 1

*Proof.* We bound the time complexity of Algorithm 1 by analyzing its three steps separately. Let $T_\theta$ denote the per-layer inference cost of the SMoE model, so that one forward pass through the $L$-layer model costs $O(L\,T_\theta)$.

**Step 1 (Cluster-wise restoration).** The saliency scores $a_i$ and pairwise output distances $dist_{\text{out}}(i,j)$ are accumulated jointly during a single forward pass over $\mathcal{D}$. This forward pass costs $O\big(|\mathcal{D}|\,L\,T_\theta\big)$. Hierarchical clustering on the $N \times N$ output-distance matrix at each layer costs $O\big(L\,N^2 \log N\big)$, which is independent of $|\mathcal{D}|$ and dominated by the forward-pass term whenever $|\mathcal{D}|\,T_\theta \gtrsim N^2 \log N$. Each cluster base $\mathbf{S}_\mathcal{G}$ is constructed in closed form by saliency-weighted averaging, with total cost $O(|\Theta_E|)$ across all layers, dominated by the forward-pass term.

**Step 2 (Diagonal residual-code initialization).** We analyze the covariance-based initialization used to set the residual bases and diagonal codes; the implementation details are given in Appendix E. For this step, write each expert weight as $\mathbf{W}_i \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, where $d_{\text{out}}$ and $d_{\text{in}}$ denote the row and column dimensions of the weight matrix. For each cluster $\mathcal{G}$, ResCode computes residuals $\mathbf{U}_i = \mathbf{W}_i - \mathbf{S}_\mathcal{G}$ and initializes the shared bases from the

weighted residual covariances

$$\sum_{i\in\mathcal{G}} \omega_i \mathbf{U}_i \mathbf{U}_i^\top, \qquad \sum_{i\in\mathcal{G}} \omega_i \mathbf{U}_i^\top \mathbf{U}_i.$$

Residual construction costs $O(|\Theta_E|)$ across the model. Accumulating the covariances costs

$$O\left(\sum_{\ell=1}^{L}\sum_{\mathcal{G}} |\mathcal{G}|(d_{\text{out}}^2 d_{\text{in}} + d_{\text{out}} d_{\text{in}}^2)\right).$$

Computing the full eigendecomposition of the $d_{\text{out}} \times d_{\text{out}}$ and $d_{\text{in}} \times d_{\text{in}}$ covariances costs

$$O\left(\sum_{\ell=1}^{L}\sum_{\mathcal{G}} (d_{\text{out}}^3 + d_{\text{in}}^3)\right).$$

Initializing the diagonal codes by projection costs

$$O\left(\sum_{\ell=1}^{L}\sum_{\mathcal{G}} |\mathcal{G}|\, r\, d_{\text{out}} d_{\text{in}}\right).$$

Step 2 does not scale with the number of calibration samples. In our setting, with fixed model architecture and residual rank, this one-time initialization overhead is dominated by the calibration forward/backward computation in Steps 1 and 3.

**Step 3 (Output-guided fitting).** The teacher outputs $\{y^\ell(\mathbf{x})\}_{\ell=1}^{L}$ are cached once at the start of fitting, incurring a one-time cost of $O\big(|\mathcal{D}|\, L\, T_\theta\big)$ and no per-epoch overhead. For each of the $n_{ep}$ epochs and each sample $\mathbf{x} \in \mathcal{D}$, the algorithm performs one forward pass through the compressed model to obtain $\{\hat{y}^\ell(\mathbf{x})\}_{\ell=1}^{L}$, followed by one backward pass that updates only the residual parameters $\{\mathbf{A}_\mathcal{G}, \mathbf{B}_\mathcal{G}, \mathbf{z}_i\}$. Both the forward and backward passes cost $O(L\, T_\theta)$ per sample, and the residual-only parameter update is dominated by these passes since the residual parameter count is much smaller than $|\Theta_E|$. Treating $n_{ep}$ as a constant, the total cost of Step 3 is $O\big(|\mathcal{D}|\, L\, T_\theta\big)$.

**Total.** Summing across the three steps, the dominant term is the forward and backward computation in Steps 1 and 3, both of which scale as $O\big(|\mathcal{D}|\, L\, T_\theta\big)$. Therefore, the overall time complexity of Algorithm 1 is $O\big(|\mathcal{D}|\, L\, T_\theta\big)$, which establishes the claim and shows that RESCODE satisfies Property 4 (Linear Complexity). □

### D.2 A Direct Comparison with ResMoE

We compare RESCODE with ResMoE (Ai et al., 2025) in terms of practicality. RESCODE is practical because its two main components are regular and cheap: base construction is a non-iterative weighted average, and restoration uses dense low-rank bases with compact expert codes. This makes the compressed structure easier to deploy than ResMoE, which relies on Wasserstein-barycenter preprocessing or irregular sparse residuals.

**Irregular residuals in ResMoE.** ResMoE restores each expert as a barycenter expert plus an expert-specific residual, and compresses the residual by unstructured pruning or SVD. The unstructured-pruning variant can preserve accuracy, but it produces irregular sparse residual matrices. Let a residual matrix have $D = d_{\text{in}}\, d_{\text{out}}$ entries with sparsity $s$. A dense fp16 matrix costs $16D$ bits. A sparse representation must store both nonzero values and their locations. A common explicit format for unstructured sparse matrices is COO, also known as the coordinate format, which stores three arrays: row indices, column indices, and nonzero values. Since the residual contains $(1-s)D$ nonzero entries, storage requires

$$(1-s)\, D\, (16 + 2b_{\text{idx}})$$

bits, where $b_{\text{idx}}$ is the number of bits per stored index. Each nonzero entry stores one fp16 value and two indices, one for the row and one for the column. This index overhead makes unstructured residual pruning dependent on custom sparse kernels, compact index formats, or dense fallback storage. In contrast, RESCODE adopts dense low-rank bases and diagonal codes, yielding an efficient implementation based on standard matrix multiplications and element-wise products.

**Cost of base construction.** The barycenter expert in ResMoE is obtained by solving a free-support Wasserstein barycenter problem. Consider a cluster $\mathcal{G}$ whose expert MLP is represented by $m$ support points of dimension $q$. Each barycenter iteration builds $|\mathcal{G}|$ cost matrices of size $m \times m$, which costs

$$O\big(|\mathcal{G}|\, m^2\, q\big).$$

Solving the corresponding optimal-transport subproblems with $S$ inner iterations adds

$$O\big(|\mathcal{G}|\, S\, m^2\big).$$

Table 8: Deployment benchmark of BF16 Mixtral-8×7B on a single NVIDIA B200 GPU at 50% expert-memory reduction.

| Metric | Full | ResMoE-SVD | ResCode |
|---|---|---|---|
| Resident memory (GB) | 87.0 | 45.8 | 46.7 |
| Prefill (tok/s) | 27,606 | 5,141 | **14,730** |
| Decode (tok/s) | 158.5 | 70.9 | **149.9** |

Therefore, with $T$ outer barycenter iterations, the base construction cost is

$$O\big(T\,|\mathcal{G}|\,m^2\,(q+S)\big)\,,$$

with $O(|\mathcal{G}|\,m^2)$ transport or cost-matrix storage.

ResCode constructs the cluster base by saliency-weighted averaging:

$$\mathbf{S}_{\mathcal{G}} = \sum_{i\in\mathcal{G}} \alpha_i^{\mathcal{G}}\,\mathbf{W}_i.$$

For each expert weight matrix, this costs only $O(|\mathcal{G}|\,d_{\text{in}}\,d_{\text{out}})$, or $O(|\mathcal{G}|\,m\,q)$ under the same support-point view. Thus, ResCode avoids both the quadratic $m^2$ transport term and iterative optimal transport optimization.

**Residual memory scaling.** We compare residual memory at the cluster level; the per-layer values in Table 2 follow by summing over the $C$ clusters in a layer. If each expert in a cluster $\mathcal{G}$ keeps an independent rank-$r$ residual, the per-cluster residual memory scales as

$$O(|\mathcal{G}|\,r\,(d_{\text{in}}+d_{\text{out}}))\,,$$

which corresponds to $O\big(r\,(d_{\text{in}}+d_{\text{out}})\,N\big)$ at the layer level under ResMoE's single-cluster setting ($|\mathcal{G}|=N$, $C=1$). ResCode instead coalesces residual directions within each cluster into cluster-shared bases:

$$\mathbf{U}_i \approx \mathbf{A}_{\mathcal{G}}\,\operatorname{diag}(\mathbf{z}_i)\,\mathbf{B}_{\mathcal{G}}.$$

Its per-cluster residual memory is

$$O(r\,(d_{\text{in}}+d_{\text{out}})+|\mathcal{G}|\,r)\,,$$

and summing over the $C$ clusters in a layer recovers $O\big(r\,(d_{\text{in}}+d_{\text{out}})\,C\big)$ in Table 2. As $d_{\text{in}}+d_{\text{out}} \gg |\mathcal{G}|$ in SMoE models, storing a shared basis pair and expert codes avoids most of the per-expert overhead.

**Deployment benchmark details.** The benchmark in Table 8 runs BF16 Mixtral-8×7B on a single NVIDIA B200 GPU at batch size 8 with standard PyTorch operations. Because attention, routers, embeddings, and layer norms remain uncompressed, the 50% expert-memory reduction corresponds to 46.3% reduction in end-to-end resident memory (from 87.0 GB to 46.7 GB).

## E Experimental Setup

This section details the experimental setup, including models, calibration data, evaluation tasks, baselines, and implementation details.

**Models.** We evaluate ResCode on three representative open-source SMoE LLMs that differ in scale, expert count, and routing pattern. Qwen1.5-MoE-A2.7B-Chat (Qwen Team, 2024) has 24 MoE layers with 60 fine-grained experts per layer and top-4 routing, and additionally uses an always-on shared expert; we compress the 60 routed experts and leave the shared expert intact. Mixtral 8×7B-v0.1 (Jiang et al., 2024) has 32 MoE layers with 8 experts per layer and top-2 routing. DeepSeek-MoE-16B (Dai et al., 2024) has 28 MoE layers with 64 routed experts per layer, top-6 routing, and two always-on shared experts; we compress the routed experts and leave the shared experts intact (Appendix F). All three models are loaded from the official HuggingFace checkpoints in `bfloat16`, and the non-expert components (embeddings, attention, layer norms, and routers) are kept frozen and uncompressed throughout the compression procedure. Mixtral 8×7B-v0.1 is released under the Apache 2.0 license, Qwen1.5-MoE-A2.7B-Chat under the Tongyi Qianwen license, and DeepSeek-MoE-16B under the DeepSeek License Agreement; we use all checkpoints consistently with their respective terms for research purposes.

**Calibration dataset.** We use DCLM (Li et al., 2024a) as the default calibration corpus. We randomly sample 2,400 sequences for Qwen1.5-MoE and 2,080 sequences for Mixtral, each of length 2048 tokens from the training split, which serve as the calibration set $\mathcal{D}$ for saliency estimation, output-distance clustering, and output-guided fitting. For the multi-seed results in Section 5.2, we repeat the complete compression pipeline three times, varying the calibration-sampling and optimization seeds jointly. For the calibration dataset experiment in Section 5.5, we additionally evaluate ResCode with C4 (Raffel et al., 2020), SlimPajama (Soboleva et al., 2023), English Wikipedia (Wikimedia Foundation, 2024), and StarCoder (Li et al., 2023) as alternative calibration sources, drawing the same

number of sequences from each dataset.

**Evaluation tasks.** We measure zero-shot accuracy on the eight standard reasoning and language-understanding benchmarks reported in Table 3: ARC-Challenge, ARC-Easy (Clark et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021), OpenBookQA (Mihaylov et al., 2018), RTE (Bentivogli et al., 2009), and WinoGrande (Sakaguchi et al., 2021). All evaluations are performed with the `lm-evaluation-harness` (Gao et al., 2023) under the default zero-shot configuration for each task, without any task-specific prompt tuning or in-context demonstrations. The average accuracy column in Table 3 is the unweighted mean over these eight tasks.

**Compression ratios.** We report results under two compression ratios—25% and 50%—defined as the relative reduction in expert parameter memory $1 - |\widehat{\Theta}_E|/|\Theta_E|$. Following Properties 2 and 3, the per-layer and per-expert effective budgets are not constrained to be uniform; RESCODE governs the per-expert allocation through cluster membership and salient-singleton selection, while the per-layer residual rank is fixed across layers.

**Competitors.** We summarize the competitors of RESCODE as follows:

- **S-prune** (He et al., 2025) prunes the least salient experts under a frequency- and norm-based importance score.
- **REAP** (Lasby et al., 2026) prunes experts using a routing-aware importance metric that accounts for each expert's contribution to the layer output.
- **M-SMoE** (Li et al., 2024b) merges experts via uniform-weighted averaging within routing-similarity groups.
- **HC-SMoE** (Chen et al., 2025a) hierarchically clusters experts by output similarity on calibration data and merges each cluster into a single representative expert.
- **ResMoE** (Ai et al., 2025) decomposes each expert into a shared common expert and a low-rank residual. Among ResMoE's two residual-compression variants, unstructured pruning and SVD, we adopt SVD. The SVD variant stores low-rank residual components in the same form as RESCODE and isolates the effect of RESCODE's design, while the unstructured-pruning variant produces irregular sparse residuals that are impractical to deploy (Appendix D). The residual rank of ResMoE-SVD is chosen to match the serialized memory of RESCODE (rank 2,545 for Mixtral and 558 for Qwen at the 50% setting).

We follow the official implementations of all competitors whenever available, and match their calibration data selection and ratio definitions to ours for fair comparison.

**Implementation details.** We implement RESCODE with PyTorch (Paszke et al., 2019) on top of HuggingFace Transformers (Wolf et al., 2020). Saliency $a_i$ and output distances $dist_{\text{out}}(i, j)$ are accumulated in a single forward pass over $\mathcal{D}$, and the cluster bases $\mathbf{S}_\mathcal{G}$ are computed in closed form from Eq. (3).

RESCODE uses a frequency-weighted residual decomposition to initialize the cluster-shared residual bases and expert-specific diagonal codes. For each expert $i \in \mathcal{G}$, let $\mathbf{U}_i = \mathbf{W}_i - \mathbf{S}_\mathcal{G}$, and let $\omega_i$ be the routing frequency of expert $E_i$ on $\mathcal{D}$. The goal is to initialize

$$\mathbf{U}_i \approx \mathbf{A}_\mathcal{G} \operatorname{diag}(\mathbf{z}_i) \mathbf{B}_\mathcal{G}.$$

Intuitively, the shared bases $\mathbf{A}_\mathcal{G}$ and $\mathbf{B}_\mathcal{G}$ should capture residual directions that appear consistently across experts in the same cluster, while reflecting the routing distribution within the cluster. We therefore estimate weighted residual covariances within each cluster, so that their top eigenvectors capture dominant output and input subspaces of the cluster residuals, with each expert weighted by its routing contribution.

Specifically, we initialize $\mathbf{A}_\mathcal{G} \in \mathbb{R}^{m \times r}$ as the top-$r$ eigenvectors of

$$\sum_{i \in \mathcal{G}} \omega_i \mathbf{U}_i \mathbf{U}_i^\top,$$

which captures output directions most commonly affected by the cluster residuals. Similarly, we initialize $\mathbf{B}_\mathcal{G} \in \mathbb{R}^{r \times n}$ as the transpose of the top-$r$ eigenvectors of

$$\sum_{i \in \mathcal{G}} \omega_i \mathbf{U}_i^\top \mathbf{U}_i,$$

which captures input directions to which the cluster residuals are the most sensitive.

Given these shared bases, we then initialize each expert-specific code by projecting the expert residual onto the paired basis directions:

$$\mathbf{z}_i = \operatorname{diag}\left( \mathbf{A}_\mathcal{G}^\top \mathbf{U}_i \mathbf{B}_\mathcal{G}^\top \right).$$

Table 9: Zero-shot accuracy of compressed DeepSeek-MoE-16B (Dai et al., 2024) at 50% compression across benchmark tasks. Bold numbers denote the best result among compressed methods. RESCODE outperforms existing methods on the majority of tasks.

| Method | Type* | ARC-c | ARC-e | BoolQ | HSwag | MMLU | OBQA | RTE | WinoG | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|
| HC-SMoE (2025a) | M | 0.3225 | 0.6271 | 0.6807 | 0.4433 | 0.2480 | 0.2140 | **0.6390** | 0.6511 | 0.4782 |
| REAP (2026) | P | 0.3328 | 0.6313 | 0.6401 | 0.5112 | 0.2324 | **0.2840** | 0.5957 | 0.6527 | 0.4850 |
| ResMoE (2025) | D | 0.3191 | 0.6431 | **0.7367** | 0.4213 | 0.2654 | 0.2340 | 0.5740 | 0.6709 | 0.4831 |
| **RESCODE (Proposed)** | D | **0.3532** | **0.6481** | 0.7254 | **0.5118** | **0.2751** | 0.2440 | 0.5848 | **0.6717** | **0.5018** |

* P: Pruning, M: Merging, D: Decomposition.

Table 10: Comparison of residual parameterizations with identical output-guided fitting (OGF) at 50% expert-memory reduction, where independent residuals correspond to ResMoE and shared codes to RESCODE.

| Model | Residual | One-shot | +OGF | Params | GPU-h |
|---|---|---|---|---|---|
| Mixtral | Independent | 0.4475 | 0.5847 | 17.35B | 14.4 |
| | **Shared codes** | 0.5885 | **0.6096** | 0.43B | 0.95 |
| Qwen | Independent | 0.4200 | 0.5434 | 6.07B | 2.44 |
| | **Shared codes** | 0.5059 | **0.5466** | 0.052B | 0.64 |

For output-guided fitting, we train only the residual parameters $\{\mathbf{A}_\mathcal{G}, \mathbf{B}_\mathcal{G}, \mathbf{z}_i\}$ with AdamW (learning rate 1.5e-4, weight decay 0.01) for $n_{ep} = 3$ epochs over $\mathcal{D}$, using gradient accumulation of 8 sequences per optimizer step. We set $\delta = 10^{-3}$ for numerical stability in Eq. (3). The residual rank $r$ is set uniformly across layers as a hyperparameter. The singleton ratio is fixed at $\rho = 0.15$, i.e., the top-$\lceil \rho N \rceil$ experts per layer by saliency are protected as singleton clusters and the remaining experts are partitioned into similarity-based clusters.

While $r$ and $\rho$ are uniform hyperparameters, the resulting per-layer and per-expert effective budgets are non-uniform: the layer-specific cluster count $C_\ell$ depends on the saliency distribution at each layer (Property 2), and an expert's effective parameter budget within a layer depends on whether it is a singleton (full per-expert capacity) or shares a rank-$r$ residual basis with its cluster siblings (Property 3). This two-level non-uniformity is the mechanism visualized as effective bit-widths in Figure 5a (Section 5.4). All experiments are conducted on a single NVIDIA B200 180GB GPU. The reported compression runtime in Table 4 measures the end-to-end wall-clock time from loading the pretrained checkpoint to writing the compressed parameters.

# F Further Results

## F.1 Generalization to DeepSeek-MoE

We further evaluate RESCODE on DeepSeek-MoE-16B (Dai et al., 2024) to verify its generalization beyond the Mixtral and Qwen models. Table 9 reports zero-shot accuracy at 50% compression against the strongest baselines from each category. RESCODE achieves the highest average accuracy of 0.5018, outperforming REAP, the strongest baseline, by 1.68 pp (and HC-SMoE by 2.36 pp).

Table 11: Average accuracy of RESCODE across calibration sizes (sequences × epochs) on Mixtral-8×7B at 50% reduction.

| Seq. × ep. | 520×12 | 1,040×6 | 2,080×3 | 6,240×1 | 9,600×1 |
|---|---|---|---|---|---|
| Accuracy | 0.5978 | 0.6015 | **0.6076** | 0.5950 | 0.6000 |

Table 12: Average accuracy across compression ratios (values of Figure 1).

| Model | Method | 25% | 37.5% | 50% | 62.5% | 75% |
|---|---|---|---|---|---|---|
| Mixtral | RESCODE | 0.6476 | 0.6218 | 0.6076 | 0.5703 | 0.5379 |
| | ResMoE | 0.6068 | 0.5879 | 0.4475 | 0.3734 | 0.3491 |
| Qwen | RESCODE | 0.5793 | 0.5580 | 0.5424 | 0.5153 | 0.5056 |
| | ResMoE | 0.5711 | 0.5115 | 0.4200 | 0.4004 | 0.3689 |

## F.2 ResMoE with Output-guided Fitting

Table 10 compares residual parameterizations under identical calibration data and output-guided fitting (OGF). OGF substantially recovers the independent-residual parameterization (Base + I3 in Table 5), yet RESCODE surpasses it while training 40× fewer parameters at lower fitting cost.

## F.3 Calibration-size Scaling

We vary the number of DCLM calibration sequences for Mixtral-8×7B at 50% expert-memory reduction, keeping approximately 390 optimizer steps for the settings with 520–6,240 sequences by adjusting the number of epochs. Table 11 shows that about 2,000 sequences suffice. Increasing both the data and optimization budget with 9,600 sequences does not improve accuracy.

## F.4 Compression Ratios

Table 12 lists the exact values underlying Figure 1. These runs share the single fixed seed used in Sections 5.5–5.6, thus the 25% and 50% values differ

Table 13: Per-task zero-shot accuracy of RESCODE at 50% expert-memory reduction, averaged over three independent compression runs.

| **Task** | **Qwen1.5-MoE** | **Mixtral 8×7B** |
|---|---|---|
| ARC-c | 0.3632 | 0.4829 |
| ARC-e | 0.6396 | 0.7650 |
| BoolQ | 0.7557 | 0.8243 |
| HSwag | 0.4961 | 0.6005 |
| MMLU | 0.4678 | 0.5290 |
| OBQA | 0.2633 | 0.3033 |
| RTE | 0.7040 | 0.6450 |
| WinoG | 0.6672 | 0.7261 |
| Avg. | 0.5446 | 0.6095 |

Table 14: All evaluated settings of the singleton ratio $\rho$ under mild (top) and aggressive (bottom) compression.

| **Qwen, 25%** | $\rho = 0$ | $\rho = 0.10$ | $\rho = 0.15$ | $\rho = 0.20$ |
|---|---|---|---|---|
| Accuracy | 0.5831 | 0.5831 | 0.5839 | **0.5870** |
| **Mixtral, 50%** | $\rho = 0$ | $\rho = 0.125$ | $\rho = 0.25$ | $\rho = 0.375$ |
| Accuracy | **0.6098** | 0.6085 | 0.6032 | 0.6076 |

slightly from those of the independently seeded runs in Table 3.

### F.5 Multi-seed Results

Table 13 reports per-task zero-shot accuracy when the complete compression pipeline of RESCODE is repeated with three independent calibration-sampling and optimization seeds at 50% expert-memory reduction. The eight-task averages are $0.6095 \pm 0.0035$ on Mixtral-8×7B and $0.5446 \pm 0.0058$ on Qwen1.5-MoE.

### F.6 Singleton-ratio Sweeps

Table 14 reports all evaluated settings of the singleton ratio $\rho$ underlying Figure 7, including the settings omitted there ($\rho = 0$ for Qwen and $\rho = 0.375$ for Mixtral).