S^{2}MoE: Shared-Subspace Mixture of Sparse Experts for Model Merging

Feihong He[1,2] **Anke Tang**[3] **Enneng Yang**[1] **Hao Jiang**[4] **Guojie Zhu**[4]
Gang Li[5] **Xiaochun Cao**[1] **Li Shen**[1,2*]

[1] Shenzhen Campus of Sun Yat-sen University
[2] Shenzhen Loop Area Institute
[3] Nanyang Technological University
[4] Huawei Technologies Ltd.
[5] University of Chinese Academy of Sciences
hefh3@mail2.sysu.edu.cn, anke.tang@ntu.edu.sg, yangenn@mail.sysu.edu.cn,
jianghao66@huawei.com, zhuguojie2@huawei.com, caoxiaochun@mail.sysu.edu.cn,
shenli6@mail.sysu.edu.cn

Abstract

Model merging offers a training-free paradigm for integrating multiple specialized models into a unified multi-task system, enabling efficient knowledge reuse from models. However, parameter interference remains a critical challenge, limiting the scalability and efficacy of merged models. SVD-based merging techniques represent the current state-of-the-art, excelling in alleviating interference through task-specific knowledge extraction and subspace orthogonalization. Despite these advances, they inevitably incur performance degradation due to aggressive parameter truncation, which discards valuable information and hinders inference accuracy. Our analysis reveals this truncation-induced knowledge loss as the fundamental bottleneck, motivating the need for mechanisms that preserve and recover discarded expertise without retraining. To address this, we introduce S^{2}MoE, a novel framework that augments SVD-based merging with a mixture of sparse experts to refine the truncation gap. S^{2}MoE comprises three key components: (1) a shared module that aggregates multi-task knowledge, (2) sparse experts that selectively recover truncation knowledge, and (3) a subspace mapping router to achieve precise training-free routing. Extensive experiments across vision and language benchmarks demonstrate up to 1.6% and 1.2% performance gains over SOTA methods, effectively validating robust and scalable model merging.

1 Introduction

The pretrain-finetune paradigm has become the dominant approach for applying deep learning models to downstream tasks. While this paradigm enables the development of highly specialized models for individual tasks, it also leads to the proliferation of multiple fine-tuned models, each optimized for a specific task or domain. This situation motivates the study of model merging techniques [1, 2, 3], which aim to integrate knowledge from multiple task-specific models into a single model, improving deployment efficiency and enabling multi-task capabilities without extensive retraining. In this context, merging task-specific models into a unified model emerges as an effective and scalable solution for enabling knowledge transfer and facilitating multi-task learning [4].

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Task arithmetic [5], as a prominent model merging approach, introduces the concept of task vectors to represent the parameter-space shift induced by a specific task. Based on this formulation, performing appropriate operations on task vectors enables functional editing of the model. Through task arithmetic, knowledge from multiple downstream models can be extracted and injected into a unified model. This approach offers a cost-effective solution for model merging. It is particularly useful for building multi-task systems when training data is scarce or restricted by privacy constraints.

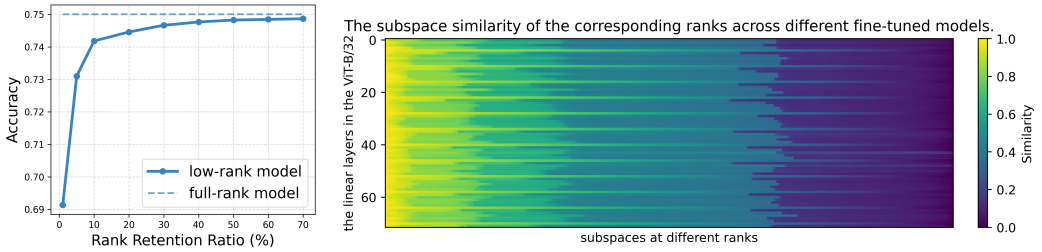


Figure 1: Experiment on the low-rank property of models (ViT-B/32). The left figure shows model performance after applying SVD to the weights and retaining different fractions of ranks. The right figure reports the subspace similarity of linear layers between two fine-tuned models (Sun397 and Stanford-cars). We argue that the rank-one component with lower similarity capture more task-specific discriminative knowledge. The results indicate that such low-similarity subspaces (<0.3) account for more than 40% of the overall subspace.

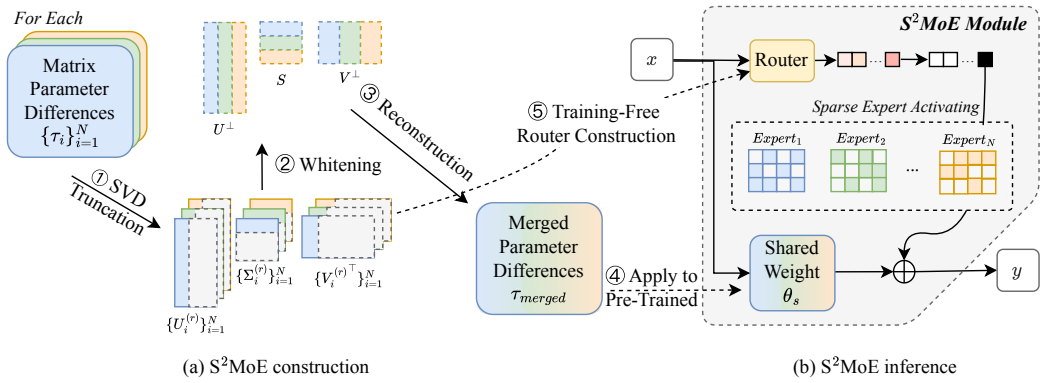
However, task arithmetic is prone to performance degradation due to parameter conflicts, and numerous studies [6, 7, 4] have been proposed to address this challenge. Singular Value Decomposition based merging [8, 9, 10, 11] represents the current state-of-the-art, outperforming a wide array of existing methods by leveraging subspace orthogonalization or alignment to mitigate parameter interference. Among them, Task Singular Vector Merge (TSVM) [8] demonstrates that applying singular value decomposition to task vectors and truncating them can retain high performance with only a small number of parameters. Based on this, it constructs a unified multi-task model by concatenating the decomposed matrices from multiple downstream models, thereby significantly alleviating parameter conflicts. Nevertheless, SVD-based merging methods suffer from performance degradation caused by parameter discarding during matrix truncation or alignment.

To address these challenges, we propose Shared-Subspace Mixture of Sparse Experts (S^2MoE), a training-free sparse mixture-of-experts approach that leverages the advantages of model merging. Our method transforms task-specific models into a mixture-of-experts architecture through model merging. First, we construct a shared module via model merging to absorb extensive multi-task knowledge, and then compress the expert modules through pruning to retain only the most important parameters. Second, during the routing construction, we retain and reuse the singular matrices obtained in the shared module construction process, and design a subspace-mapping-based routing mechanism to realize a training-free mixture of experts model. Finally, the introduced MoE architecture compensates for the loss of multi-task knowledge during core merging, thereby mitigating the performance degradation caused by model merging.

We conduct extensive experiments on both vision models and large language models, covering a wide range of scales from 80M to 7B parameters. Results show that S^2MoE consistently outperforms existing model merging methods on both vision and language benchmarks, with particularly strong performance in scenarios involving a large number of diverse downstream tasks. Notably, S^2MoE achieves significant improvements in alleviating parameter conflicts and mitigating the performance degradation caused by parameter discarding. With the aid of model merging, S^2MoE even surpasses fine-tuned models in certain cases due to cross-task knowledge transfer, while requiring neither additional training nor access to the original training data.

In summary, our main contributions are as follows:

- We present a method that leverages the structural properties of the Mixture-of-Experts architecture to alleviate performance degradation from parameter dropping in model merging, leading to consistent improvements across multiple tasks.
- We merge models to integrate diverse expert knowledge into a shared module, enabling more aggressive expert weight sparsification and lowering active parameters at inference.



(a) S^{2}MoE construction (b) S^{2}MoE inference

Figure 2: S^{2}MoE Overview. The left panel illustrates shared module construction, while the right presents expert module sparsification and the S^{2}MoE framework's workflow.

- Our method requires no access to data and incurs no additional retraining or fine-tuning. We validate its effectiveness on diverse vision and language tasks, consistently outperforming existing merging methods.

2 Understanding Information Loss in SVD-Based Merging Techniques

To investigate the performance impact of extensive parameter discarding in SVD-based merging, we employ SVD to truncate the task vector of a fine-tuned model at different levels by rank and evaluate the performance, i.e. $\text{Acc}(\theta_{pre} + \text{RankTrunc}(\mathcal{T}))$ vs. $\text{Acc}(\theta_{pre} + \mathcal{T})$. As shown in Figure 1, when retaining 20% of the rank, the low-rank model achieves performance comparable to the original fine-tuned model. However, as the retained rank ratio continues to decrease, the low-rank model inevitably suffers a sharp performance decline. For existing SVD-based merging techniques such as TSVM, the rank retention ratio is fixed to $1/N$, where N denotes the number of tasks being merged. Consequently, when merging models from eight tasks, only 12.5% of the singular components are preserved. As N increases, the retention rank decreases, leading to progressively more aggressive information loss and inherently limiting the scalability to the number of tasks.

On the other hand, the right panel of Figure 1 visualizes the subspace similarity across all linear layers of ViT-B/32 between two fine-tuned models, compared at different ranks after applying singular value decomposition. The visualization reveals two distinct patterns: The left region corresponding to singular components with high singular values exhibits high similarity values, indicating that these singular components capture shared, task-agnostic knowledge common to both fine-tuned models, whereas the right region components encode task-specific discriminative features. Notably, over 40% of the subspaces demonstrate low similarity, serving as the carriers of task-specific knowledge. During multi-task merging with increasing task numbers, existing SVD-based methods often just discard these low-similarity components, which explains the performance degradation observed in the left panel of Figure 1 and highlight the fundamental limitation of aggressive rank truncation in SVD-based merging methods.

These observations motivate a key question: *how can we preserve task-specific knowledge while still benefiting from the shared representations captured by SVD-based merging?* Rather than choosing between scalability and performance, we need a mechanism that selectively retains the task-specific components discarded during rank truncation, while maintaining the memory efficiency and shared knowledge extraction benefits of SVD-based approaches.

3 Methodology: S^{2}MoE

Based on the above analysis, we propose Shared-Subspace Mixture of Sparse Experts (S^{2}MoE), combining a shared backbone merging with a training-free subspace projection router.

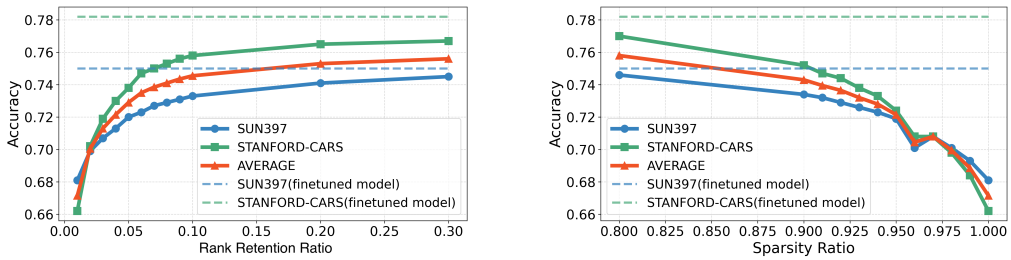


Figure 3: Ablation study on rank retention ratio and expert sparsity. (LHS) The left panel shows the performance of shared modules constructed with different rank retention ratios, where a higher rank retention ratio leads to improved performance. (RHS) Right panel presents performance under different expert sparsity levels, where increasing sparsity results in more performance degradation.

3.1 Preliminary

The Mixture-of-Experts (MoE) architecture effectively scales model capacity without proportionally increasing computation. A MoE model is composed of two key components: a set of N modular experts, and a router. The parameters can be formally expressed as follows:

$$\theta_{\text{MoE}} = \{\theta_1, \ldots, \theta_N; \theta_r\}. \tag{1}$$

Each expert θ_i specializes in one task or a subset of tasks, while the centralized router θ_r dynamically assigns token-level inputs to appropriate experts.

In this work, we also adopt the concept of task vectors from Task Arithmetic [5], where the difference between a pretrained model and its fine-tuned counterpart encodes the task-specific adaptation, i.e., $\mathcal{T}_i = \Theta_i - \Theta_{\text{pre}}, \forall i \in [N]$, where Θ_{pre} is the pre-trained weight and Θ_i is the weight of the i-th fine-tuned model. This is also referred to as the parameter differences or deltas.

3.2 Overall Architecture of S^{2}MoE

We propose using a MoE architecture with scalable experts to support a large number of tasks. This design helps mitigate the performance drop caused by discarding a significant portion of parameters during model merging. Unlike many existing MoE approaches [12, 13] that require training the router, S^{2}MoE adopts a training-free routing mechanism, which will be detailed in Section 3.5. Compared to conventional MoE architectures, S^{2}MoE achieves lower memory overhead during construction and inference, thereby enabling better scalability.

The complete architecture of S^{2}MoE is illustrated in Figure 2. In this work, we adopt mainstream Transformer-based architectures [14], including both vision and large language models [15, 16, 17], as backbone. As shown in Figure 2(a), the construction of S^{2}MoE proceeds through steps ① to ④ to build a shared module for maximal multi-task knowledge sharing, and step ⑤ constructing a training-free router from the right singular vectors. Figure 2(b) illustrates the inference process, where the router dynamically activates the selected sparse experts, and the outputs are combined with that of the shared part. Formally, the parameter set and the forward propagation process of S^{2}MoE can be briefly expressed as follows:

$$\theta_{\text{S}^2\text{MoE}} = \{\theta_s; \tilde{\theta}_1, \ldots, \tilde{\theta}_N; \theta_r\}, f(x) = f_{\theta_s}(x) + f_{Expert}(x) \text{ and } f_{Expert}(x) = \sum_{i=1}^{N} \tilde{\alpha}_i \cdot f_{\tilde{\theta}_i}(x). \tag{2}$$

where x denotes the input sample, $f_{\theta_s}(\cdot)$ denotes the shared module constructed from matrix weights of linear layers, while $f_{\tilde{\theta}_i}(\cdot)$ and $\tilde{\alpha}_i$ represent the i-th sparse expert and the routing weight of the router, respectively. This design allows shared module and sparse experts to complement each other, with the router dynamically selecting relevant experts for each input.

3.3 Shared Module Construction

Due to the low-rank property of matrix parameters from linear layers, we apply SVD to decompose the weight matrix and retain only the low-rank components with high singular values, allowing us

to capture the majority of the model's knowledge with fewer parameters. Mathematically, we first perform SVD on the task vectors: $\mathcal{T}_i = U_i S_i V_i^\top,\ \forall i \in [N]$, where U_i, S_i, and V_i are the left singular vectors, singular values, and right singular vectors, respectively. Subsequently, we truncate and concatenate the low-rank components from each task to construct a unified $USV^\top$. The truncation and concatenation process is described as:

$$U = \left[U_1^{(r)} \mid U_2^{(r)} \mid \cdots \mid U_N^{(r)}\right] \in \mathbb{R}^{w \times \mathrm{Tr}}, \tag{3}$$

$$S = \mathrm{diag}\left(\left[S_1^{(r)} \mid S_2^{(r)} \mid \cdots \mid S_N^{(r)}\right]\right) \in \mathbb{R}^{\mathrm{Tr} \times \mathrm{Tr}}, \tag{4}$$

$$V = \left[V_1^{(r)} \mid V_2^{(r)} \mid \cdots \mid V_N^{(r)}\right] \in \mathbb{R}^{h \times \mathrm{Tr}}. \tag{5}$$

Here r denotes the truncation rank for each task vector and $U_i^{(r)}$, $S_i^{(r)}$, $V_i^{(r)}$ represent the resulting truncated left singular matrix, diagonal singular values matrix, and right singular matrix, respectively. The term $\mathrm{Tr} = rN$ denotes the total number of rank-1 components preserved across the set of N experts, and the operator $[\cdot|\cdot]$ denotes the concatenation function. However, directly concatenating these components may lead to parameter conflicts due to direction non-orthogonality and task-specific variations. To address this, we apply SVD-based whitening to orthogonalize them:

$$\mathrm{SVD}(U) = P_U D_U Q_U^\top, \quad \mathrm{SVD}(V) = P_V D_V Q_V^\top, \tag{6}$$

$$U^\perp = P_U Q_U^\top, \quad V^\perp = P_V Q_V^\top. \tag{7}$$

Finally, the merged task vector is constructed using the following formulation: $\mathcal{T}_{merged} = U^\perp S V^{\perp\top}$, Here, $U^\perp$ and $V^\perp$ denote the whitened versions of U and V. We further apply the merged task vector to the pre-trained weight to obtain the final share module weight as follows:

$$\theta_s = \Theta_{pre} + \mathcal{T}_{merged}. \tag{8}$$

The left panel of Figure 3 shows the model accuracy across different rank retention ratios, comparing the Top-1 accuracy on two tasks after merging parameters from models fine-tuned on Sun397 and Cars. The results indicate that model merging effectively integrates knowledge from multiple tasks into a single model, and the overall performance improves as the proportion of merged parameters increases. This highlights the importance of consolidating multi-task knowledge in the shared module to enable richer integration.

3.4 Experts Construction and Sparsification

As discussed in Section 2, despite the effectiveness of SVD-based merging in consolidating multi-task knowledge, there remains a performance gap between the merged model and the individual fine-tuned models due to the necessary discarding of low-similarity task-specific components. To address this limitation, we introduce sparse expert modules that preserve the task-specific knowledge lost during the merging process. Since the shared module already captures most of the common knowledge across tasks, the experts only need to retain the complementary task-specific information, allowing us to apply aggressive sparsification without significant performance degradation.

We denote the parameters of an expert module as $\theta_i = \Theta_i - \theta_s = (\theta_{i,1}, \theta_{i,2}, \ldots, \theta_{i,n}) \in \mathbb{R}^n$. Our objective is to prune this parameter vector by retaining only k nonzero entries $\tilde{\theta}_i$ subject to $\|\tilde{\theta}_i\|_0 \leq k$. To minimize the perturbation introduced to the network, we minimize the squared ℓ_2 error between the original and pruned parameters: $D(\theta_i, \tilde{\theta}_i) := \|\theta_i - \tilde{\theta}_i\|_2^2$. Under a strict k-sparsity constraint, the optimization objective is given by:

$$\min_{\|\tilde{\theta}_i\|_0 \leq k} \sum_{j=1}^{n} (\theta_{i,j} - \tilde{\theta}_{i,j})^2. \tag{9}$$

Since each $\tilde{\theta}_{i,j} \in \{\theta_{i,j}, 0\}$, the distortion simplifies to:

$$\|\theta_i - \tilde{\theta}_i\|_2^2 = \sum_{j \in \mathcal{S}_{\text{pruned}}} \theta_{i,j}^2, \tag{10}$$

where $\mathcal{S}_{\text{pruned}}$ denotes the pruned parameter indices. Minimizing it requires identifying and retaining the k largest-magnitude entries:

$$\mathcal{S}^*_{\text{keep}} = \arg\max_{\mathcal{S}_{\text{keep}}} \sum_{j \in \mathcal{S}_{\text{keep}}} \theta^2_{i,j} \quad s.t. \quad |\mathcal{S}_{\text{keep}}| = k, \tag{11}$$

yielding the optimal pruned expert:

$$\tilde{\theta}_{i,j} = \begin{cases} \theta_{i,j}, & \text{if } j \in \text{Top-}k(\theta^2_{i,j}), \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

The right panel of Figure 3 validates this approach across different sparsity levels. As expected, increasing sparsity degrades performance. However, since the shared module already captures most common knowledge, the experts only need to preserve complementary task-specific information. This enables aggressive sparsification, avoiding the sharp performance drops typical of sparse models.

3.5 Subspace Mapping-Based Routing

In this section, we construct a routing mechanism based on subspace residual projection, leveraging the right singular matrix defined in Eq. (5).

The right singular matrix $V_i^{(r)}$ constitutes an orthonormal basis that spans the subspace of task i within the input space. Geometrically, the right singular vectors in $V_i^{(r)}$ characterize the most significant directions of parameter updates during the fine-tuning process for task i [18, 19]. Given an input representation x, its orthogonal projection onto this subspace is defined as:

$$\hat{x}^{(i)} = V_i^{(r)} V_i^{(r)\top} x, \tag{13}$$

where $P^{(i)} = V_i^{(r)} V_i^{(r)\top}$ denotes the orthogonal projection matrix. The corresponding residual vector is $x - \hat{x}^{(i)}$, with its L_2 norm distance given by:

$$e_i = ||x - V_i^{(r)} V_i^{(r)\top} x||_2. \tag{14}$$

A marginal residual e_i indicates that the primary components of x are highly aligned with the basis vectors in $V_i^{(r)}$. Since $V_i^{(r)}$ captures the directions of most intensive weight updates during task-specific fine-tuning, such alignment suggests that the features carried by x reside within the specialized domain of the i-th expert. Consequently, this enables the construction of a precise routing mechanism based on subspace mapping. Based on the aforementioned geometric intuition, we formalize the routing process as a mapping from subspace residuals to activation weights. The procedure is delineated as follows:

To transform physical distances into a probability distribution, we apply the Softmax function to the negative residuals. For a set of N experts, the initial routing weight α_i for the i-th expert is:

$$\alpha_i = \frac{\exp(-e_i)}{\sum_{j=1}^{N} \exp(-e_j)}, \quad \text{for } i \in [N]. \tag{15}$$

To further incentivize expert specialization and mitigate inter-expert interference, we implement a Top-K selection strategy. Upon obtaining α_i, we retain only the top K scoring experts and mask the remaining weights to zero. Let $\mathcal{M}_i \in \{0, 1\}$ denote the binary mask. The filtered weights $\tilde{\alpha}_i$ are subsequently normalized to maintain a unit sum:

$$\mathcal{M}_i = \begin{cases} 1, & \text{if } \alpha_i \in \text{Top-}K(\{\alpha_j\}_{j=1}^{N}) \\ 0, & \text{otherwise} \end{cases}, \tag{16}$$

This step guarantees that the model activates only the most relevant subspace experts for a given input during inference. The final output of the expert module is as follows:

$$f_{Expert}(x) = \sum_{i=1}^{N} \tilde{\alpha}_i \cdot f_{\tilde{\theta}_i}(x), \text{ where } \tilde{\alpha}_i = \frac{\mathcal{M}_i}{\sum_{j=1}^{N} \mathcal{M}_j}, \tag{17}$$

where $f_{\tilde{\theta}_i}(\cdot)$ represents the operation defined by the i-th sparse expert.

3.6 Computational and Memory Efficiency.

Importantly, routing decisions are determined entirely by the geometric structure of the subspaces, requiring no parameter training or data access. During inference, the routing computation involves only matrix multiplications to project inputs onto task subspaces and compute residuals, which is negligible compared to the forward pass through the shared module. Furthermore, the Top-K selection ensures that only K out of N experts are activated per input, significantly reducing computational overhead compared to activating all experts.

In terms of memory efficiency, S^{2}MoE stores one shared module plus N sparse experts, which is substantially more efficient than storing N full fine-tuned models. This design enables S^{2}MoE to scale effectively to a large number of tasks while maintaining practical memory requirements.

4 Experiments and Analysis

In this section, we describe the experimental setup and present the results and analysis across vision and language tasks using models of different scales. Additional results are provided in the appendix D.

4.1 Experiment Setup

Models and Datasets: Following the mainstream experimental settings in recent model merging research for vision tasks, we perform comparative experiments on eight image classification datasets: SUN397 [20], Stanford-cars [21], RESISC45 [22], EuroSAT [23], SVHN [24], GTSRB [25], MNIST [26], and DTD [27]. In natural language tasks, there is currently no unified benchmark that serves as a standard for evaluation. To address this, we construct two broad sets of natural language tasks for a comprehensive experimental comparison. These two sets cover a wide range of capabilities, including mathematical reasoning (GSM8K-CoT [28]), multilingual understanding (ARC [29], HELLASWAG [30], MMLU [31]), instruction following (IFEVAL [32]), code generation and comprehension (HUMANEVAL-PLUS [33], MBPP-PLUS [34]), safety evaluation (TOXIGEN [35], TRUTHFULQA [36], WINOGENDER [37]), sentiment classification (SST2 [38]), natural language inference (MNLI [38]), sentence-pair similarity (MRPC [38]), and commonsense understanding (HELLASWAG [30]). In our experiments, we select ViT-B/32 [15], LLaMA-3.2-3B [17], and LLaMA-2-7B [16] as the base models, covering parameter scales from approximately 80M to 7B.

Baselines and Metric: Our baselines fall into three categories: (1) individually fine-tuned models, serving as the theoretical upper bound of performance, (2) model merging methods, and (3) training-free MoE-based methods.Specifically, we include: Single Task Learning, Weight Averaging [39], Fisher Merging [40], RegMean [41], Task Arithmetic [5], Ties Merging [42], AdaMerging [43], Dare [44], TSVM [8], and SMILE [18]. We report per-task accuracy and their arithmetic mean.

Hardware: All experiments are performed on a server equipped with an Intel(R) Xeon(R) Platinum 8160 CPU and eight NVIDIA RTX A6000 48 GB GPUs for computational acceleration.

4.2 Experimental Results

Vision Tasks: In the vision tasks, we evaluated the ViT-B/32 model on eight datasets: SUN397, Stanford-Cars, RESISC45, EuroSAT, SVHN, GTSRB, MNIST, and DTD, and summarized the results in Table 1. On these tasks, we compared our proposed method with various existing model merging approaches and MoE-based methods. The baselines cover a wide range of strategies, including traditional parameter averaging, importance-weighted merging, parameter editing using task vectors, conflict resolution optimization, and adaptive merging at task or layer granularity. In addition, we reported Single Task Learning results on the same tasks as the theoretical upper bound.

From Table 1, Weight Averaging performs worse than Single Task Learning on all tasks, with 0.528 on GTSRB, a 46.6% drop from 0.989. Fisher Merging, which is based on parameter importance, yields certain improvements in some tasks such as SVHN, but overall remains unstable. RegMean and Task Arithmetic outperform simple averaging on some medium-scale tasks such as RESISC45 and EuroSAT, but show noticeable degradation on tasks with larger distribution shifts such as DTD. Ties Merging provides slight improvements over Task Arithmetic in most tasks, which indicates that conflict resolution can bring marginal gains. AdaMerging in the layer-level mode significantly outperforms its task-level variant. For instance, it reaches 0.695 on Stanford-Cars compared with

Table 1: Performance comparison on ViT-B/32 across multiple tasks. The best results, excluding the Single Task Learning for the corresponding tasks, are highlighted in **bold**.

Method / Task	Sun397	Stanford-cars	Resisc45	Eurosat	Svhn	Gtsrb	Mnist	Dtd	AVG.
Single Task Learning	0.750	0.783	0.952	0.990	0.973	0.989	0.996	0.797	0.903
Weight Averaging [39]	0.653	0.633	0.714	0.736	0.642	0.528	0.875	0.501	0.660
Fisher Merging [40]	0.686	0.692	0.707	0.664	0.729	0.511	0.879	0.599	0.683
RegMean [41]	0.653	0.635	0.756	0.786	0.781	0.674	0.937	0.520	0.718
Task Arithmetic [5]	0.553	0.549	0.667	0.774	0.802	0.697	0.973	0.501	0.690
Ties Merging [42]	0.650	0.643	0.747	0.768	0.813	0.694	0.965	0.543	0.728
Iso-C ($\lambda = 1.5$) [9]	0.698	0.729	0.860	0.898	0.876	0.924	0.985	0.655	0.828
Iso-CTS ($\lambda = 1.5$) [9]	0.711	0.746	0.865	0.890	0.834	0.904	0.981	0.685	0.827
AdaMerging (Task) [43]	0.583	0.532	0.718	0.801	0.816	0.844	0.934	0.427	0.707
AdaMerging (Layer) [43]	0.642	0.695	0.824	0.925	0.865	0.937	0.977	0.611	0.809
TSVM [8]	0.676	0.716	0.846	0.934	0.918	0.925	0.988	0.638	0.830
SMILE [18]	0.715	0.746	0.898	0.980	0.966	0.971	0.995	0.767	0.880
S^2MoE (Ours)	**0.742**	**0.780**	**0.936**	**0.983**	**0.971**	**0.987**	**0.996**	**0.769**	**0.896**

Table 2: Performance comparison on LLaMA-3.2-3B across multiple natural language tasks. The best results, excluding the Single Task Learning for the corresponding tasks, are highlighted in **bold**.

Method / Task	Mathematics	Multilingual understanding	Instruction-following	Coding	Safety	AVG.
Single Task Learning	0.571	0.391	0.553	0.579	0.482	0.515
Weight Averaging [39]	0.378	0.242	0.192	0.431	0.398	0.391
Task Arithmetic [5]	0.655	0.397	0.513	0.543	0.482	0.518
Dare [44]	0.615	0.380	0.490	0.500	0.477	0.492
TSVM [8]	**0.720**	0.398	0.543	0.533	0.476	0.534
SMILE [18]	0.658	0.400	0.562	0.529	**0.483**	0.526
S^2MoE (Ours)	0.682	**0.405**	**0.588**	**0.586**	0.474	**0.546**

0.532 for the task-level variant, which demonstrates stronger fine-grained parameter adaptation capabilities. However, AdaMerging requires unlabeled data for training, which introduces additional computational cost. TSVM achieves robust improvements across multiple tasks, attaining an average accuracy of 0.830 in a training-free setting. In contrast, our proposed S^2MoE achieves the best results on all eight tasks, reaching 0.742 on SUN397, 0.780 on Stanford-Cars, and 0.936 on RESISC45. It achieves an average accuracy of 0.896, which surpasses the second-best method SMILE (0.880) by 1.6 percentage points. The performance is already very close to the theoretical upper bound of single-task learning, with only a 0.007 drop in average accuracy compared to Single Task Learning. These results demonstrate that S^2MoE effectively mitigates inter-task interference in multi-task vision merging while alleviating performance degradation caused by weight truncation.

Natural Language Tasks: Table 2 and Table 3 present a comparative analysis of S^2MoE against other model merging and MoE methods on natural language tasks. Table 2 shows that Weight Averaging reduces performance on most tasks. TSVM achieves 0.720 on math reasoning but performs weakly on multilingual and coding tasks. Task Arithmetic and Dare yield stable yet limited results, while SMILE excels in instruction following (0.562) but not in math or multilingual tasks. In contrast, S^2MoE achieves the best results in multilingual understanding with 0.405, instruction following with 0.588, and coding with 0.586. Its average accuracy reaches 0.546, which is higher than the second-best method TSVM with 0.534. This demonstrates that recovering the pruned weights can further improve the overall performance of the model. Table 3 shows results on LLaMA-2-7B. TSVM obtains the

Table 3: Performance comparison on LLaMA2-7B. Best results (excluding single-task baselines) are in **bold**.

Method / Task	SST2	MNLI	Hellaswag	MRPC	AVG.
Single Task Learning	0.955	0.615	0.629	0.683	0.721
Weight Averaging	0.934	0.587	0.596	0.669	0.696
Task Arithmetic	0.941	0.594	0.587	0.676	0.700
Dare	0.938	**0.595**	0.586	0.661	0.695
TSVM	0.955	0.593	0.592	0.701	0.710
SMILE	0.957	0.594	0.596	0.705	0.713
S^2MoE (Ours)	**0.959**	0.594	**0.607**	**0.716**	**0.718**

highest score on MRPC with 0.701. Dare performs slightly better on MNLI with 0.595. SMILE maintains balanced results across tasks with an average of 0.713. S^2MoE achieves the best results on SST2 with 0.959, Hellaswag with 0.607, and MRPC with 0.716. Its average reaches 0.718, which is higher than SMILE with 0.713 and TSVM with 0.710.

Overall, the results in Table 2 and Table 3 show that S^2MoE can achieve advantages across multiple task types in natural language settings. It also maintains a consistent improvement trend across large language models of different scales. These results indicate that S^2MoE has strong generality and scalability in multi-task scenarios.

Cross-dataset Evaluation: Table 4 reports the cross-dataset evaluation results of LLaMA2-7B, fine-tuned on GSM8K-RFT, CodeAlpaca, and ShareGPT, and tested on GSM8K, IFEval, and MBPP. First, Weight Averaging achieves the best performance on MBPP (0.294) but performs poorly on GSM8K (0.300) and IFEval (0.254), indicating that simple averaging struggles to balance knowledge across heterogeneous tasks. Task Arithmetic and Dare remain stable across tasks but fail to achieve top performance, resulting in a moderate average. In contrast, S^2MoE consistently achieves competitive or superior performance across all three benchmarks. It obtains the best results on GSM8K (0.365) and IFEval (0.280), while maintaining comparable accuracy on MBPP (0.264). In terms of average accuracy, S^2MoE achieves the highest average accuracy of 0.303. This demonstrates its effectiveness in mitigating parameter conflicts and knowledge loss through a shared-sparse design.

Table 4: Cross-dataset evaluation of LLaMA2-7B. Best results are in **bold**.

Method / Task	GSM8K	IFEval	MBPP	AVG.
Single Task Learning	0.382	0.286	0.254	0.307
Weight Averaging	0.300	0.254	**0.294**	0.282
Task Arithmetic	0.294	0.273	0.278	0.282
Dare	0.288	0.269	0.272	0.276
TSVM	0.362	0.275	0.264	0.300
SMILE	0.348	0.256	0.264	0.289
S^2MoE (Ours)	**0.365**	**0.280**	0.264	**0.303**

Statistical Significance Analysis In this section, we conduct experiments on statistical significance. Since S^2MoE is training free and involves no stochastic optimization, parameter initialization, or router training, repeating the complete model construction and evaluation procedure across multiple random seeds produces identical results. Table 5 reports the mean and standard deviation across seeds, where the zero standard deviation confirms that the reported results are not affected by seed-dependent randomness.

Table 5: Mean and standard deviation of the average accuracy across multiple random seeds.

Setting	**Mean $\pm$ standard deviation**
ViT-B/32	0.8962 ± 0.0000
LLaMA-3.2-3B	0.5460 ± 0.0000

Seed consistency alone does not quantify uncertainty arising from finite test sets. We therefore perform a paired bootstrap analysis on the ViT-B/32 eight-task benchmark. Within each of the eight vision tasks, we resample test examples with replacement and apply the same sampled indices to S^2MoE and each baseline. We then compute the performance difference for each task and take the macro average over the eight tasks, using 10000 bootstrap replicates with seed 2026. As shown in Table 6, both paired confidence intervals are strictly above zero, and no non-positive improvement is observed within the resolution of the 10000-replicate bootstrap. We additionally compute absolute bootstrap confidence intervals for the macro average of every task, obtaining 0.896 ± 0.0063 for ViT-B/32 and 0.546 ± 0.0125 for LLaMA-3.2-3B. Together, these results demonstrate that the improvements of S^2MoE over SMILE and TSVM are not attributable to seed-dependent randomness or finite test-sample variability.

5 Related Works

Model Merging [1, 45, 46, 47] edits models by directly manipulating their parameters. Its low cost and simplicity have led to wide adoption in computer vision [39, 42, 48, 49] and natural language processing [13, 40]. Model Soup [39] improves generalization by averaging weights of models

Table 6: Paired bootstrap significance analysis (10,000 replicates) on the ViT-B/32 eight-task benchmark. Gain is measured in percentage points (pp).

Comparison	S^2MoE	Baseline	Gain (pp)	Paired 95% CI	Bootstrap Pr($\Delta \leq 0$)
S^2MoE vs. SMILE	0.8962	0.8801	+1.61	[+1.40, +1.82]	$< 10^{-4}$
S^2MoE vs. TSVM	0.8962	0.8306	+6.56	[+6.28, +6.84]	$< 10^{-4}$

trained on different tasks or epochs. Task arithmetic [5] introduced task vectors—the parameter differences between fine-tuned and pretrained models—as fundamental units for model merging. This concept has driven extensive follow-up research [43, 42, 50, 51, 52], primarily addressing two challenges: balancing merging coefficients and resolving parameter conflicts. Task Singular Vector Merge (TSVM) [8] applies SVD with truncation, concatenation, and whitening to mitigate parameter conflicts. However, discarding low-rank components inevitably causes knowledge loss, a fundamental limitation of SVD-based methods that we address in this work.

Multi-task Learning [53, 54, 55, 56, 57] aims to train a single model to handle multiple related tasks efficiently, maximizing parameter efficiency and enabling cross-task knowledge transfer. Recent advances include representation learning methods that model task relationships through subspace constraints or attention mechanisms [58], and parameter-efficient techniques like adapters that insert lightweight task-specific modules into shared backbones [59]. The MoE architecture [60, 61] has emerged as a promising approach, using dynamic routing to allocate computation efficiently, share knowledge among related tasks, and isolate unrelated tasks. This enables improved scalability and performance with controlled computational costs. While these MoE approaches are primarily data-driven, our method distinguishes itself by operating directly on model parameters. This approach bypasses data dependencies and provides a novel perspective for multi-task MoE frameworks.

6 Conclusion and Limitations

We present S^2MoE, a scalable and training-free model merging framework that integrates task-specific parameter subspaces with subspace-mapping-based selective expert activation. This design effectively eliminates the performance drop typical of traditional model merging while preserving deployment efficiency. Across diverse vision and language benchmarks, S^2MoE achieves the highest average accuracy, surpassing state-of-the-art merging and MoE methods, and shows strong generalization from vision models to large-scale language models. While S^2MoE demonstrates significant strengths, its current construction primarily targets architectures where linear layers predominate. Since the subspace mapping process relies on linear dependencies, extending this approach to networks with highly non-linear structural components remains a challenge. Future work will focus on generalizing the subspace selection mechanism to accommodate more complex non-linear architectures, further broadening the applicability of S^2MoE.

Impact Statement

We propose a training-free framework for efficient multi-task knowledge aggregation. This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

Acknowledgement

This work is supported by National Key R&D Program of China (Grant No.2024YFC3307100), NSFC Grant (No. 62576364, No. U2541229), GuangDong Basic and Applied Basic Research Foundation (2026B1515020071), Shenzhen Science and Technology Program (NO. JCYJ20241202124430041), Shenzhen Science and Technology Program (NO.SYSRD20250529113401002), Postdoctoral Fellowship Program and China Postdoctoral Science Foundation under Grant (No. BX2026301), the China Postdoctoral Science Foundation under Grant (No. 2025M781535).

References

[1] Anke Tang, Li Shen, Yong Luo, Han Hu, Bo Du, and Dacheng Tao. Fusionbench: A comprehensive benchmark of deep model fusion. *arXiv preprint arXiv:2406.03280*, 2024.

[2] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model Merging in LLMs, MLLMs, and Beyond: Methods, Theories, Applications and Opportunities.

[3] Prateek Yadav, Colin Raffel, Mohammed Muqeeth, Lucas Caccia, Haokun Liu, Tianlong Chen, Mohit Bansal, Leshem Choshen, and Alessandro Sordoni. A Survey on Model MoErging: Recycling and Routing Among Specialized Experts for Collaborative Learning.

[4] Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless Knowledge Fusion by Merging Weights of Language Models. In *The Eleventh International Conference on Learning Representations*, 2023.

[5] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. 2023.

[6] Guodong Du, Junlin Lee, Jing Li, Runhua Jiang, Yifei Guo, Shuyang Yu, Hanting Liu, Sim K Goh, Ho-Kin Tang, Daojing He, et al. Parameter competition balancing for model merging. *Advances in Neural Information Processing Systems*, 37:84746–84776, 2024.

[7] Daniel Marczak, Simone Magistri, Sebastian Cygert, Bartłomiej Twardowski, Andrew D. Bagdanov, and Joost van de Weijer. No Task Left Behind: Isotropic Model Merging with Common and Task-Specific Subspaces. In *Forty-Second International Conference on Machine Learning*, June 2025.

[8] Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodola. Task singular vectors: Reducing task interference in model merging. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 18695–18705, 2025.

[9] Daniel Marczak, Simone Magistri, Sebastian Cygert, Bartłomiej Twardowski, Andrew D Bagdanov, and Joost van de Weijer. No task left behind: Isotropic model merging with common and task-specific subspaces. *arXiv preprint arXiv:2502.04959*, 2025.

[10] George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with svd to tie the knots. *arXiv preprint arXiv:2410.19735*, 2024.

[11] Kuangpu Guo, Yuhe Ding, Jian Liang, Zilei Wang, and Ran He. Stay unique, stay efficient: Preserving model personality in multi-task merging. *arXiv preprint arXiv:2512.01461*, 2025.

[12] Anke Tang, Li Shen, Yong Luo, Nan Yin, Lefei Zhang, and Dacheng Tao. Merging Multi-Task Models via Weight-Ensembling Mixture of Experts. In *Proceedings of the 41st International Conference on Machine Learning*, pages 47778–47799. PMLR, July 2024.

[13] Zhenyi Lu, Chenghao Fan, Wei Wei, Xiaoye Qu, Dangyang Chen, and Yu Cheng. Twin-merging: Dynamic integration of modular expertise in model merging. *Advances in Neural Information Processing Systems*, 37:78905–78935, 2024.

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

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2021.

[16] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[17] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407, 2024.

[18] Anke Tang, Li Shen, Yong Luo, Shuai Xie, Han Hu, Lefei Zhang, Bo Du, and Dacheng Tao. Zero-shot sparse mixture of low-rank experts construction from pre-trained foundation models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[19] Donato Crisostomi, Alessandro Zirilli, Antonio Andrea Gargiulo, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, Iacopo Masi, and Emanuele Rodolà. MASS: MoErging through Adaptive Subspace Selection.

[20] Jianxiong Xiao, Krista A Ehinger, James Hays, Antonio Torralba, and Aude Oliva. Sun database: Exploring a large collection of scene categories. *International Journal of Computer Vision*, 119(1):3–22, 2016.

[21] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013.

[22] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

[23] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

[24] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 7. Granada, 2011.

[25] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks*, pages 1453–1460. IEEE, 2011.

[26] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[27] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014.

[28] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[29] Viet Dac Lai, Chien Van Nguyen, Nghia Trung Ngo, Thuat Nguyen, Franck Dernoncourt, Ryan A Rossi, and Thien Huu Nguyen. Okapi: Instruction-tuned large language models in multiple languages with reinforcement learning from human feedback. *arXiv preprint arXiv:2307.16039*, 2023.

[30] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a Machine Really Finish Your Sentence? In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics.

[31] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*, October 2020.

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

[33] Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Lei Shen, Zihan Wang, Andi Wang, Yang Li, et al. Codegeex: A pre-trained model for code generation with multilingual benchmarking on humaneval-x. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5673–5684, 2023.

[34] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[35] Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. *arXiv preprint arXiv:2203.09509*, 2022.

[36] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022. *URL https://arxiv. org/abs/2109.07958*, 1, 2021.

[37] Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. Gender bias in coreference resolution. *arXiv preprint arXiv:1804.09301*, 2018.

[38] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[39] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022.

[40] Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

[41] Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*, 2022.

[42] Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36:7093–7115, 2023.

[43] Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. AdaMerging: Adaptive Model Merging for Multi-Task Learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[44] Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*, 2024.

[45] Kunda Yan, Min Zhang, Sen Cui, Qu Zikun, Bo Jiang, Feng Liu, and Changshui Zhang. Calm: Consensus-aware localized merging for multi-task learning. In *Forty-second International Conference on Machine Learning*.

[46] Gabriel J Perin, Xuxi Chen, Shusen Liu, Bhavya Kailkhura, Zhangyang Wang, Brian Gallagher, et al. Rankmean: Module-level importance score for merging fine-tuned large language models. *FINDINGS OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: ACL 2024*, page 7, 2024.

[47] Yuyan Zhou, Liang Song, Bingning Wang, and Weipeng Chen. Metagpt: Merging large language models using model exclusive task arithmetic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1724, 2024.

[48] Benjamin Biggs, Arjun Seshadri, Yang Zou, Achin Jain, Aditya Golatkar, Yusheng Xie, Alessandro Achille, Ashwin Swaminathan, and Stefano Soatto. Diffusion soup: Model merging for text-to-image diffusion models. In *European Conference on Computer Vision*, pages 257–274. Springer, 2024.

[49] Xiao Liu, Xiaoliu Guan, Yu Wu, and Jiaxu Miao. Iterative ensemble training with anti-gradient control for mitigating memorization in diffusion models. In *European Conference on Computer Vision*, pages 108–123. Springer, 2024.

[50] Hyo Seo Kim, Dongyoon Han, and Junsuk Choe. Negmerge: Sign-consensual weight merging for machine unlearning. In *Forty-second International Conference on Machine Learning*.

[51] Yu-Ang Lee, Ching-Yun Ko, Tejaswini Pedapati, I Chung, Mi-Yen Yeh, Pin-Yu Chen, et al. Star: Spectral truncation and rescale for model merging. *arXiv preprint arXiv:2502.10339*, 2025.

[52] Rui Dai, Sile Hu, Xu Shen, Yonggang Zhang, Xinmei Tian, and Jieping Ye. Leveraging submodule linearity enhances task arithmetic performance in llms. In *ICLR*, 2025.

[53] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[54] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[55] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016.

[56] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Latent multi-task architecture learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4822–4829, 2019.

[57] Yangyang Xu, Yibo Yang, and Lefei Zhang. Multi-Task Learning with Knowledge Distillation for Dense Prediction. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 21493–21502, Paris, France, October 2023. IEEE.

[58] Yongxin Yang and Timothy Hospedales. Deep multi-task representation learning: A tensor factorisation approach. *arXiv preprint arXiv:1605.06391*, 2016.

[59] Alexandre Audibert, Massih Amini, Konstantin Usevich, and Marianne Clausel. Low-rank updates of pre-trained weights for multi-task learning. In *Findings of the Association for Computational Linguistics, ACL 2023*, pages 7544–7554. Association for Computational Linguistics, 2023.

[60] Siyuan Mu and Sen Lin. A comprehensive survey of mixture-of-experts: Algorithms, theory, and applications. *arXiv preprint arXiv:2503.07137*, 2025.

[61] Luozheng Qin, Jia Gong, Qian Qiao, Tianjiao Li, Li Xu, Haoyu Pan, Chao Qu, Zhiyu Tan, and Hao Li. Uni-vigu: Towards unified video generation and understanding via a diffusion-based video generator. *arXiv preprint arXiv:2604.08121*, 2026.

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

A Pseudocode

In this section, we present the pseudocode of S^{2}MoE, covering the overall workflow of S^{2}MoE 1, the construction of the shared module 2, the sparsification of expert modules 3, and the subspace-mapping-based routing mechanism 4.

Algorithm 1 summarizes the overall workflow of S^{2}MoE, where Ensemble$(\cdot)$ and Forward$(\cdot)$ correspond to model construction and inference, respectively.

Algorithm 1 S^{2}MoE

Require: Pretrained Θ_{pre}, fine-tuned $\{\Theta_i\}_{i=1}^N$, rank r, sparsity k (ratio ρ), router Top-K, the l-th layer input $x^{(l)}$

Function Ensemble($\Theta_{\text{pre}}, \{\Theta_i\}_{i=1}^N, r, k, K$)
Shared Module Construction: $\theta_s, \{V_i^{(r),(l)}\} \leftarrow \textsc{BuildShared}(\Theta_{\text{pre}}, \{\Theta_i\}_{i=1}^N, r)$
Sparsifying the Expert Modules:
$\{\tilde{\theta}_i\}_{i=1}^N \leftarrow \textsc{PruneExperts}(\{\theta_i\}_{i=1}^N, k)$
$(\rho \Rightarrow k = \lfloor \rho n \rfloor)$
Setup router: θ_r using $\{V_i^{(r),(l)}\}$
return θ_{MoE}
End Function
Function Forward($x;\ \theta_{\text{MoE}}$)
for each routed layer ℓ **do**
    Shared output: $y_s^{(l)} \leftarrow f_{\theta_s^{(l)}}(x^{(l)})$
    Hierarchical Expert Routing:
    $\tilde{\alpha}^{(\ell)} \leftarrow \textsc{Router}(x^{(l)}, \theta_r, K)$
    Expert aggregation:
    $y_{Expert}^{(\ell)} \leftarrow \sum_{i=1}^N \tilde{\alpha}_i^{(\ell)} f_{\tilde{\theta}_i}^{(\ell)}(x^{(\ell)})$
    Hierarchical Aggregation:
    $x^{(l+1)} \leftarrow y_s^{(l)} + y_{Expert}^{(\ell)}$
end for
End Function

As shown in Algorithm 2, the shared module is constructed by extracting residual updates from fine-tuned models, applying low-rank SVD decomposition, and compressing the concatenated subspaces into orthonormal bases. The merged update is then added back to the pretrained backbone to form shared parameters θ_s.

As shown in Algorithm 3, we sparsify the expert modules by pruning each fine-tuned model θ_i. For each prunable tensor W, only the top-k entries with the largest magnitudes are preserved while all others are zeroed out, resulting in a sparse tensor $\tilde{W}$. The pruned tensors are then assembled into sparse experts $\{\tilde{\theta}_i\}_{i=1}^N$. When a sparsity ratio ρ is used, the pruning threshold k is determined as $k = \lfloor \rho n \rfloor$, where n denotes the tensor size. This procedure effectively reduces storage and computation while retaining the most influential parameters.

Algorithm 4 outlines the subspace mapping routing mechanism, which determines expert activation by evaluating the alignment of the input $x^{(\ell)}$ with each expert subspace. The projection error onto $V_i^{(r),\ell}$ is first computed and transformed into routing scores α_i through an exponential mapping. To achieve sparsity and efficiency, only the Top-K scores are retained, with the remaining entries set to zero, followed by normalization to yield the final routing weights $\tilde{\alpha}^{(\ell)}$. This process adaptively selects the most relevant experts without requiring additional training.

Algorithm 2 BUILDSHARED

Require: Pre-trained weights Θ_{pre}, fine-tuned models $\{\Theta_i\}_{i=1}^N$, truncation rank r

Ensure: Shared parameters θ_s, truncated right singular matrices $\{V_i^{(r),\ell}\}_{i,\ell}$

Function BuildShared($\Theta_{\text{pre}}, \{\Theta_i\}, r$)
for each layer ℓ **do**
  for $i = 1$ **to** N **do**
    Compute task vector: $\mathcal{T}_i \leftarrow \Theta_i^{(\ell)} - \Theta_{\text{pre}}^{(\ell)}$
    Perform SVD: $\mathcal{T}_i = U_i S_i V_i^\top$
    Retain rank-r components: $(U_i^{(r)}, S_i^{(r)}, V_i^{(r)})$
  end for
  Parameter Concatenation:
  $U \leftarrow [U_1^{(r)} \mid \cdots \mid U_N^{(r)}], \quad V \leftarrow [V_1^{(r)} \mid \cdots \mid V_N^{(r)}]$
  $S \leftarrow \text{diag}([S_1^{(r)} \mid \cdots \mid S_N^{(r)}])$ {Dimension: $\mathbb{R}^{Tr \times Tr}$}
  SVD-based Whitening to Mitigate Conflict:
  Decompose $U = P_U D_U Q_U^\top$ and $V = P_V D_V Q_V^\top$
  Construct whitened bases: $U^\perp \leftarrow P_U Q_U^\top, \quad V^\perp \leftarrow P_V Q_V^\top$
  Module Assembly:
  $\mathcal{T}_{\text{merged}}^{(\ell)} \leftarrow U^\perp S (V^\perp)^\top$
  $\theta_s^{(\ell)} \leftarrow \Theta_{\text{pre}}^{(\ell)} + \mathcal{T}_{\text{merged}}^{(\ell)}$
  Store task-specific bases $\{V_i^{(r),\ell}\}_{i=1}^N$ for routing
end for
return $\theta_s, \{V_i^{(r),\ell}\}$
End Function

Algorithm 3 PRUNEEXPERTS

Require: expert modules $\{\theta_i\}_{i=1}^N$, pruning size k (or ratio ρ)

Ensure: Sparse experts $\{\tilde{\theta}_i\}_{i=1}^N$

Function PruneExperts($\{\theta_i\}, k$)
for $i = 1 \ldots N$ **do**
  for each prunable tensor $W \subset \theta_i$ **do**
    Keep top-k entries in $\text{vec}(W)$ by magnitude, zero out others $\Rightarrow \tilde{W}$
  end for
  Construct $\tilde{\theta}_i$ by replacing W with $\tilde{W}$
end for
return $\{\tilde{\theta}_i\}_{i=1}^N$
End Function
Note: if a sparsity ratio ρ is used, set $k = \lfloor \rho n \rfloor$ where n is the tensor size.

B Theoretical Justification: Shared-Then-Experts vs. Direct Expert Compression

Let Θ_{pre} denote the pretrained model parameters and $\{\Theta_i\}_{i=1}^N$ represent N fine-tuned models for different tasks. We define the task vectors as $\mathcal{T}_i = \Theta_i - \Theta_{\text{pre}}$, which capture the parameter updates for the i-th downstream task.

Most existing MoE architectures [62, 18] approximate each task vector $\mathcal{T}_i$ with capacity-limited experts and combine them via routing at inference. We advocate first learning a shared displacement C through model merging (forming the shared model $\theta_s = \Theta_{\text{pre}} + C$), and then letting experts approximate only the residuals as follows:

$$R_i = \mathcal{T}_i - C. \tag{18}$$

Here the shared module captures *cross-task commonalities*, while experts handle *task-specific differences*.

Algorithm 4 ROUTER

Require: Layer input $x^{(\ell)}$, truncated right singular matrices $\{V_i^{(r),\ell}\}_{i=1}^N$, selection parameter K
Ensure: Sparse and normalized routing weights $\tilde{\alpha}^{(\ell)}$

for $i = 1 \dots N$ **do**
  Compute projection residual: $e_i \leftarrow \|x^{(\ell)} - V_i^{(r),\ell}(V_i^{(r),\ell})^\top x^{(\ell)}\|_2$
end for
for $i = 1 \dots N$ **do**
  Apply negative exponential mapping: $\alpha_i \leftarrow \frac{\exp(-e_i)}{\sum_{j=1}^N \exp(-e_j)}$
end for
Identify top-K indices: $\mathcal{I} \leftarrow \text{Top-}K(\{\alpha_j\}_{j=1}^N)$
for $i = 1 \dots N$ **do**
  Apply binary mask: $\mathcal{M}_i \leftarrow \mathbb{I}(i \in \mathcal{I})$
  Compute filtered weights: $\tilde{\alpha}_i \leftarrow \frac{\mathcal{M}_i \cdot \alpha_i}{\sum_{j=1}^N \mathcal{M}_j \cdot \alpha_j}$
end for
return $\tilde{\alpha}^{(\ell)} = [\tilde{\alpha}_1, \dots, \tilde{\alpha}_N]^\top$

To analyze the advantage, we compare two regimes under the same per-expert compression budget K (in terms of parameters, rank, or sparsity) and the same inference Top-k routing as follows:

- *Direct expert compression*: Directly compresses each task vector $\mathcal{T}_i$ into expert $\widehat{\mathcal{T}}_i$, with final model $\Theta_{\text{pre}} + \sum_i \alpha_i \widehat{\mathcal{T}}_i$.
- *Shared-Then-Experts*: First constructs a shared displacement C via model merging to form shared backbone $\theta_s = \Theta_{\text{pre}} + C$, then lets experts approximate only the residuals $R_i = \mathcal{T}_i - C$, with final model $\Theta_{\text{pre}} + C + \sum_i \alpha_i \widehat{R}_i$.

Let $\mathcal{F}(K)$ denote the hypothesis class of compressed representations with capacity at most K (e.g., rank-K matrices or K-sparse tensors). The best total approximation error achievable by direct expert compression would be

$$\mathcal{E}_{\text{direct}}(K) = \sum_{i=1}^{N} \inf_{\widehat{\mathcal{T}}_i \in \mathcal{F}(K)} \|\mathcal{T}_i - \widehat{\mathcal{T}}_i\|_F^2, \tag{19}$$

while the best total error with shared-then-experts is

$$\mathcal{E}_{\text{shared}}(K) = \inf_{C} \sum_{i=1}^{N} \inf_{\widehat{R}_i \in \mathcal{F}(K)} \|(\mathcal{T}_i - C) - \widehat{R}_i\|_F^2, \tag{20}$$

where the shared displacement C is optimized to minimize the total residual approximation error.

Taking $C = 0$ shows immediately that $\mathcal{E}_{\text{shared}}(K) \le \mathcal{E}_{\text{direct}}(K)$, since the feasible set of the shared-then-experts regime contains that of the direct expert compression. Moreover, if there exists an optimal shared displacement $C^\star$ that reduces the sum of squared residual norms,

$$\sum_{i=1}^{N} \|\mathcal{T}_i - C^\star\|_F^2 \;<\; \sum_{i=1}^{N} \|\mathcal{T}_i\|_F^2, \tag{21}$$

then the shared-then-experts regime achieves strictly better performance: $\mathcal{E}_{\text{shared}}(K) < \mathcal{E}_{\text{direct}}(K)$.

Connection to S^{2}MoE: In our method, the shared displacement C is constructed through subspace-based model merging (Section 3.3), which extracts common task knowledge while preserving orthogonal subspace information for routing. The expert modules then approximate the remaining task-specific residuals through sparsification (Section 3.4).

Overall, the strategy of first constructing a shared module through model merging and then adding sparse expert modules achieves superior performance compared to directly building sparse experts, as it leverages cross-task commonalities more effectively.

C Experimental Details

In this section, we provide additional details on the datasets used, experimental settings, and implementation details to ensure the reproducibility of our experiments.

C.1 Dataset Description

SUN397 [20]: The SUN397 dataset is a large-scale benchmark for scene recognition, comprising approximately 108000 images across 397 categories. It spans a wide range of indoor and outdoor environments, making it a challenging and representative benchmark for evaluating models on scene understanding and generalization ability.

Stanford-cars [21]: The Stanford Cars dataset is a fine-grained image classification benchmark containing around 16000 images across 196 car categories. The categories span various car makes, models, and years, providing a challenging testbed for evaluating models on fine-grained recognition and visual discrimination.

RESISC45 [22]: The RESISC45 dataset is a benchmark for remote sensing image scene classification, consisting of 31500 images evenly distributed across 45 categories. Each class contains 700 images with a spatial resolution of 256×256 pixels, collected under diverse geographic regions, seasons, weather, and illumination conditions. It poses significant challenges in scene understanding and cross-domain generalization.

EUROSAT [23]: The EUROSAT dataset is a benchmark for remote sensing scene classification, built upon Sentinel-2 satellite imagery. It consists of 27000 images categorized into 10 land-use and land-cover classes, such as residential areas, forests, farmlands, and industrial zones. Each image has a spatial resolution of 64×64 pixels, collected from diverse regions and seasons, making it a widely used dataset for evaluating models in land-use classification and remote sensing analysis.

SVHN [24]: The Street View House Numbers (SVHN) dataset is a real-world digit recognition benchmark derived from Google Street View imagery. It contains over 600000 digit images spanning 10 classes (0–9), divided into training, testing, and extra sets. Compared to MNIST, SVHN presents a more challenging setting with cluttered backgrounds, varying illumination, and multi-digit arrangements, making it widely used for evaluating models on digit recognition in natural scenes and robustness.

GTSRB [25]: The German Traffic Sign Recognition Benchmark (GTSRB) is a large-scale dataset for traffic sign recognition, comprising over 50000 images across 43 categories. The dataset exhibits significant variability in terms of resolution, illumination, weather conditions, occlusions, and rotations, closely reflecting the complexity of real-world driving environments. It has become a standard benchmark for evaluating models on object recognition, robustness, and generalization, particularly in the context of autonomous driving and intelligent transportation systems.

MNIST [26]: The MNIST dataset is a seminal benchmark for handwritten digit recognition. It contains 70000 grayscale images of digits (0-9), with 60000 for training and 10000 for testing. Each image is 28×28 pixels, centered and size-normalized. As one of the earliest and most widely used benchmarks in deep learning, MNIST serves as a standard testbed for evaluating models on image classification, feature learning, and pattern recognition.

DTD [27]: The Describable Textures Dataset (DTD) is a benchmark for texture recognition and attribute classification, consisting of 5640 images organized into 47 texture categories. Each category is defined by human-describable attributes such as banded, striped, or zigzagged. The images cover a broad range of natural and man-made textures, offering substantial diversity. DTD is widely employed to evaluate models on texture understanding, attribute recognition, and visual representation learning.

GSM8K-CoT [28]: The GSM8K (Grade School Math 8K) dataset is a benchmark for mathematical reasoning and problem solving, consisting of approximately 8500 grade-school level word problems spanning arithmetic, algebra, geometry, and applied math. Each problem is paired with a gold-standard solution. The extended GSM8K-CoT version additionally provides step-by-step reasoning annotations (Chain-of-Thought, CoT), enabling models to learn multi-step reasoning strategies. It is widely used to evaluate language models on numerical reasoning, logical inference, and multi-hop problem solving.

ARC [29]: The AI2 Reasoning Challenge (ARC) is a benchmark for commonsense reasoning and science question answering, consisting of multiple-choice questions drawn from U.S. elementary and middle school science exams. The dataset contains about 7800 questions, divided into Easy and Challenge subsets. Unlike traditional QA datasets, ARC requires background knowledge, logical reasoning, and multi-step inference to answer correctly, making it a widely used benchmark for evaluating models on knowledge reasoning, scientific understanding, and multi-hop QA.

HELLASWAG [30]: The dataset is a large-scale benchmark for commonsense reasoning and natural language understanding, comprising around 70000 multiple-choice questions. Each instance provides a context followed by several candidate endings, where the task is to select the most plausible continuation. The dataset is constructed with adversarial filtering, making distractors superficially similar to the correct choice, thus preventing models from relying on shallow statistical cues. HellaSwag is widely used to evaluate models on commonsense reasoning, contextual understanding, and narrative completion.

MMLU [31]: The Massive Multitask Language Understanding (MMLU) benchmark is a comprehensive evaluation suite spanning 57 diverse subject areas, including STEM, humanities, social sciences, and other professional domains. It consists of over 15000 multiple-choice questions designed to assess models' knowledge breadth, cross-domain understanding, and reasoning skills. Covering problems from high-school level to professional expertise, MMLU is widely regarded as a primary benchmark for measuring the general knowledge and reasoning capabilities of large language models.

IFEVAL [32]: The IFEval dataset is designed to assess large language models on their ability to follow natural language instructions. It covers a wide range of task types, including logical reasoning, text generation, format control, and knowledge-based question answering. Beyond correctness, IFEval emphasizes whether models adhere to the explicit constraints and output requirements specified in the instructions. As such, it serves as a benchmark for evaluating models on reliability, controllability, and practical usability in real-world instruction-following scenarios.

HUMANEVAL-PLUS [33]: The HumanEval-Plus dataset is an extension of the OpenAI HumanEval benchmark, designed to evaluate large language models on code generation and program synthesis. Compared to the original HumanEval, HumanEval-Plus introduces more diverse and rigorous test cases, covering complex function logic and edge conditions to prevent models from achieving high scores through memorization or pattern matching. It has become a widely used benchmark for assessing models on code generation, semantic correctness, and robustness, making it a critical resource for programming and automated reasoning evaluation.

MBPP-PLUS [34]: The MBPP-Plus (Mostly Basic Python Problems Plus) dataset is an enhanced extension of the original MBPP benchmark, designed to evaluate large language models on Python programming and code generation. While the original MBPP consists of roughly 1000 short coding problems, MBPP-Plus provides a larger and more diverse set of tasks along with more rigorous and comprehensive test cases, covering a wider range of programming skills and edge conditions. By focusing on functional correctness, robustness, and generalization, MBPP-Plus serves as a crucial benchmark for assessing the practical coding capabilities of language models.

TOXIGEN [35]: The dataset is a benchmark for toxic language detection, designed to evaluate large language models on hate speech, discriminatory content, and offensive expressions. It consists of over 270000 sentences spanning sensitive topics such as race, gender, and religion. Unlike traditional toxicity datasets, ToxiGen is constructed via a language-model–driven generation pipeline with human validation, ensuring both diversity and contextual plausibility. It is widely used for assessing models in toxicity detection, bias evaluation, and safety analysis.

TRUTHFULQA [36]: The dataset is a benchmark for evaluating large language models on factual accuracy and resistance to misinformation. It comprises around 817 open-ended questions spanning diverse domains such as commonsense, science, history, and popular culture. Designed to test whether models produce truthful and reliable answers to misleading or misconception-based queries, TruthfulQA serves as a critical benchmark for assessing models on truthfulness, robustness, and knowledge alignment.

WINOGENDER [37]: The WINOGENDER dataset is designed to evaluate gender bias in language models, inspired by the Winograd Schema Challenge. It consists of 720 coreference resolution sentences containing gendered pronouns across professions, activities, and social roles. Each instance is constructed in three variants—neutral, male, and female—to assess whether models exhibit

systematic errors due to gender stereotypes. WinoGender is widely used as a benchmark for analyzing models' performance in bias detection, fairness, and social alignment.

SST2 [38]: The Stanford Sentiment Treebank 2 (SST-2) dataset is a widely used benchmark for sentiment classification, developed at Stanford University. It consists of sentences extracted from movie reviews, annotated with binary labels (positive or negative). Unlike the original SST, SST-2 removes neutral labels, resulting in a two-class classification task. As part of the GLUE benchmark, SST-2 is commonly employed to evaluate models on sentiment analysis, text classification, and semantic understanding.

MNLI [38]: The Multi-Genre Natural Language Inference (MNLI) dataset is a large-scale benchmark for natural language inference, consisting of approximately 433000 sentence pairs. Each instance includes a premise and a hypothesis, and the task is to determine whether the relationship is entailment, contradiction, or neutral. A key feature of MNLI is its coverage of multiple genres, providing a broad evaluation of models' cross-domain reasoning, natural language understanding, and generalization capabilities. It serves as one of the central tasks in the GLUE benchmark.

MRPC [38]: The Microsoft Research Paraphrase Corpus (MRPC) is a benchmark dataset for semantic textual similarity, developed by Microsoft Research. It contains 5801 sentence pairs drawn primarily from news sources, each annotated as either paraphrase or non-paraphrase. Since many pairs exhibit significant surface-level differences while remaining semantically equivalent, MRPC provides a challenging testbed for evaluating models on semantic understanding, paraphrase detection, and textual similarity. It is one of the core tasks in the GLUE benchmark.

C.2 Model Description

ViT-B/32 [15]: The ViT-B/32 (Vision Transformer, Base, Patch Size 32) is a standard vision transformer architecture introduced by Google. The input image is divided into 32×32 patches, each linearly embedded and processed by a 12-layer Transformer encoder with a hidden size of 768 and 12 attention heads, totaling approximately 86M parameters. As a widely adopted visual backbone, particularly in CLIP and other multimodal models, ViT-B/32 demonstrates strong generalization ability and efficiency across image classification, retrieval, and transfer learning tasks.

LLaMA-3.2-3B [17]: The LLaMA-3.2-3B is a medium-scale model in the LLaMA 3.2 family released by Meta, containing approximately 3 billion parameters with a decoder-only Transformer architecture. It incorporates advances in pretraining data coverage, extended context length, and optimized training strategies, enabling strong language understanding and generation capabilities under moderate computational budgets. LLaMA-3.2-3B is widely adopted as a lightweight backbone for task-specific fine-tuning (e.g., instruction following, reasoning, dialogue) and serves as a mid-sized benchmark in model merging and comparative studies.

LLaMA2-7B [16]: The LLaMA2-7B is a model in Meta's LLaMA 2 family, released in 2023, with approximately 7 billion parameters and a decoder-only Transformer architecture. Trained on a larger and more diverse corpus with improved optimization strategies and extended context length, it achieves strong performance across tasks such as language understanding, reasoning, and dialogue generation. Striking a balance between capability and efficiency, LLaMA2-7B is widely used as a backbone for task-specific fine-tuning and serves as a standard baseline in studies of model merging, parameter-efficient adaptation, and reasoning.

C.3 Experimental Setup

In the proposed S^2MoE framework, two key hyperparameters are involved: the truncation length of the singular matrices r and the sparsity ratio of the expert modules ρ. In both the vision experiments (ViT-B/32) and the language processing tasks (LLaMA-3.2-3B and LLaMA-2-7B), we uniformly set the truncation length of the singular matrices to $r = Tr/N$, where N is the number of fine-tuned models. This setting allows the decomposition to allocate equal capacity to each fine-tuned model, thereby maximizing the integration of task knowledge across multiple domains. Meanwhile, we set the sparsity of the expert modules uniformly to 0.2, meaning that only 20% of the parameters are retained while the remaining 80% are pruned, thereby enabling efficient sparse modeling.

Table 7: Performance comparison on ViT-B/32 backbone across 14-task and 20-task scenarios.

Setting	**SUN397**	**Cars**	**RESISC45**	**EuroSAT**	**SVHN**	**GTSRB**	**MNIST**	**DTD**	**Flowers**	**PCAM**	**FER2013**
14-Task Acc.	0.7324	0.7696	0.9257	0.9778	0.9692	0.9799	0.9964	0.7351	0.8257	0.8473	0.7040
20-Task Acc.	0.7269	0.7484	0.9090	0.9715	0.9672	0.9698	0.9908	0.7202	0.8151	0.8396	0.7014

Setting	**Pet**	**STL10**	**CIFAR100**	**CIFAR10**	**Food101**	**F-MNIST**	**E-Letters**	**KMNIST**	**SST2**	**AVERAGE**
14-Task Acc.	0.9133	0.9775	0.8172	-	-	-	-	-	-	**0.8694**
20-Task Acc.	0.9092	0.9737	0.7962	0.9647	0.8606	0.9096	0.8343	0.8947	0.7177	**0.8610**

Table 8: Ablation on the truncation length r, showing that incorporating more multi-task knowledge into the shared module improves performance.

Task/$\frac{r}{T_r}$	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1	0.2	0.3
SUN397	0.681	0.699	0.707	0.713	0.720	0.723	0.727	0.729	0.731	0.733	0.741	0.745
STANFORD-CARS	0.662	0.702	0.719	0.730	0.738	0.747	0.750	0.753	0.756	0.758	0.765	0.767

Table 9: Ablation on the sparsity parameter ρ, showing that excessive pruning of expert modules leads to performance degradation.

Task/ρ	0.80	0.90	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99	1.00
SUN397	0.746	0.734	0.732	0.729	0.726	0.723	0.719	0.701	0.708	0.701	0.693	0.681
STANFORD-CARS	0.770	0.752	0.747	0.744	0.738	0.733	0.724	0.708	0.708	0.698	0.684	0.662

D Ablation Studies

To further evaluate the scalability and robustness of our proposed method, we conduct performance tests using the ViT-B/32 backbone, extending the evaluation from a 14-task to a 20-task scenario. As reported in Table 7, the model demonstrates remarkable stability when the task load increases. For instance, performance on existing tasks such as SUN397 and Stanford-Cars remains highly consistent with only marginal fluctuations. Furthermore, the model maintains superior generalization capabilities on high-accuracy datasets like MNIST and EuroSAT, regardless of the task setting. Although the overall Average Accuracy slightly adjusts from 0.8694 to 0.8610 due to the increased complexity of the 20-task environment, the minimal performance decay underscores the framework's ability to effectively mitigate negative transfer and inter-task interference. These results highlight the model's efficacy in maintaining a high performance ceiling while accommodating a broader range of diverse downstream tasks.

In Table 8 and Table 9, we present ablation studies on two hyperparameters of S^{2}MoE. In Table 8, we prune the expert modules entirely (100%) and vary the truncation length r, in order to observe how injecting different amounts of downstream task knowledge into the shared module affects

Table 10: Comparison of inference memory usage on the ViT-B/32 backbone across 8 tasks.

Method	Inference Memory (MiB)	Memory Increment
Single fine-tuned model	930.2 MiB	-
Iso-CTS [9]	930.2 MiB	-
TSV-M [8]	930.2 MiB	-
SMILE [18]	5482.0 MiB	+4551.8 MiB
Ours (S^2MoE)	**932.2 MiB**	**+2.0 MiB**

Table 11: Peak GPU Memory Usage Comparison for 8-task experimental setup.

Model Configuration	**Peak Memory (MiB)**	**Overhead vs. Fine-tuned**
ViT-B/32 (Single Fine-tuned)	930.2	-
ViT-B/32 (S^2MoE)	**932.2**	**+0.21%**
ViT-L/32 (Single Fine-tuned)	1518.0	-
ViT-L/32 (S^2MoE)	**1520.0**	**+0.13%**

overall performance. The results in Table 8 show that incorporating more multi-task information into the shared module consistently improves model performance. In Table 9, we conduct an ablation study on the sparsity hyperparameter ρ of the expert modules. Here, we fix the truncation length at $r = 0.01$ and vary ρ to examine how the sparsity level of expert modules influences performance. The results demonstrate that performance decreases progressively as ρ increases, indicating that excessive pruning leads to a noticeable drop in accuracy.

To evaluate the resource requirements during deployment, we compare the peak GPU memory usage of different methods during the inference phase. As illustrated in Table 10 and Table 11, although S^2MoE incorporates a dynamic routing mechanism, its memory footprint remains remarkably low. On the ViT-B/32 backbone, our method incurs a marginal increase of only 2.0 MiB in inference memory compared to Iso-CTS and TSV-M. In stark contrast, SMILE requires a substantial 5482.0 MiB of memory, which results in a massive increment of 4551.8 MiB. When examining the percentage of memory overhead across different architectural scales, S^2MoE introduces an additional peak memory cost of only 0.21% for ViT-B/32 and a mere 0.13% for ViT-L/32 relative to the standard single fine-tuned models. These results provide strong evidence that S^2MoE is highly memory-efficient. It imposes negligible additional pressure on hardware resources during inference.

To demonstrate the efficiency of our proposed method in terms of model weight storage, we provide a detailed breakdown of storage consumption compared to the baseline approach of storing N full fine-tuned models. Taking the ViT-B/32 model evaluated across 8 tasks as a representative case, the baseline requires storing eight independent models. This results in a substantial storage requirement of 14.216 GB. In contrast, our S^2MoE requires only 6.400 GB of storage space. This significant reduction saves over 50% of the storage footprint. These results underline the efficiency of S^2MoE in integrating multiple task-specific capabilities into a single consolidated architecture.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The motivation, contributions, and scope of our work are explicitly detailed in the Abstract and Introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed the limitations of our work in the Conclusion and Limitations section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: Our work does not focus on theoretical contributions; therefore, this item is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper fully discloses the algorithmic steps, model architectures, and all necessary hyperparameters to ensure that our experimental results can be independently reproduced.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All models and datasets used in our work are publicly accessible, and we have provided comprehensive implementation details and algorithmic descriptions in the paper.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We have specified all essential experimental configurations, including data splits, hyperparameter settings, and optimizer types, in the main paper and Appendix. Detailed tables are provided to ensure a clear understanding of how the results were obtained.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Our method is training-free and deterministic. Given a specific input and set of hyperparameters, the algorithm produces the same output consistently, making standard error bars or statistical significance tests for training variance non-applicable.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide details regarding the hardware specifications in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This study does not involve any ethical issues.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have briefly discussed the potential impacts of our work in the Impact Statement.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: Our work focuses on foundational algorithmic improvements and architectural design. It does not involve the release of high-risk pre-trained models or sensitive datasets that require specific safeguards against misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing assets used in this work, including models, datasets, and codebases, are properly cited in the references. We have explicitly mentioned their original sources and confirmed that our use of these assets strictly adheres to their respective open-source licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer:[N/A]

    Justification: This paper does not introduce or release new datasets, software packages, or pre-trained model weights. The primary contributions are algorithmic improvements, which are fully documented within the manuscript.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: While Large Language Models are the primary subject of our research, they are used as standard experimental benchmarks. Our core methodological contributions do not rely on LLMs as a non-standard or original component of the method development itself.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.