MergeME: Model Merging Techniques for Homogeneous and Heterogeneous MoEs

Yuhang Zhou [1] **Giannis Karamanolakis** [2] **Victor Soto** [2] **Anna Rumshisky** [2]
Mayank Kulkarni [2] **Furong Huang** [1] **Wei Ai** [1] **Jianhua Lu** [2]
[1] University of Maryland, College Park
[2] Amazon AGI
{tonyzhou, aiwei, furongh}@umd.edu Anna_Rumshisky@uml.edu
{karamai, nvmartin, maykul, jianhual}@amazon.com

Abstract

The recent success of specialized Large Language Models (LLMs) in domains such as mathematical reasoning and coding has led to growing interest in methods for merging these expert LLMs into a unified Mixture-of-Experts (MoE) model, with the goal of enhancing performance in each domain while retaining effectiveness on general tasks. However, the effective merging of expert models remains an open challenge, especially for models with highly divergent weight parameters or different architectures. State-of-the-art MoE merging methods only work with homogeneous model architectures and rely on simple unweighted averaging to merge expert layers, which does not address parameter interference and requires extensive fine-tuning of the merged MoE to restore performance. To address these limitations, this paper introduces new MoE merging techniques, including strategies to mitigate parameter interference, routing heuristics to reduce the need for MoE fine-tuning, and a novel method for merging experts with different architectures. Extensive experiments across multiple domains demonstrate the effectiveness of our proposed methods, reducing fine-tuning costs, improving performance over state-of-the-art methods, and expanding the applicability of MoE merging.

1 Introduction

Large language models (LLMs) pretrained on a wide-variety of corpora have achieved notable success in multiple tasks (Touvron et al., 2023; OpenAI, 2023; Brown et al., 2020; Liu et al., 2024a). With significant progress, there is increasing interest in how to continuously improve the performance of LLMs in new domains, including math (Yu et al., 2023), code (Roziere et al., 2023), Wikipedia knowledge (Shao et al., 2024), or legal domains (Cui et al., 2023). One straightforward approach is through continual pretraining (CPT) on domain-specific data, which, however, is challenging for multiple target domains, as it can cause catastrophic forgetting on previously learned tasks (Luo et al., 2023).

An alternative approach is Mixture-of-Experts (MoE) merging, where dense experts are first CPT-ed in parallel for each domain and then merged into a unified MoE model, usually by keeping feed-forward neural network (FFN) layers separate and averaging non-FFN layers (Sukhbaatar et al., 2024; Kang et al., 2024). Compared with dense models of similar size, the MoE model uses just a subset of parameters during inference by learning to route tokens to the top few experts, thus reducing inference costs. Unlike training an MoE model from scratch, MoE merging offers modularity, as individual experts are domain-specialized, and is substantially less expensive, as CPT-ing experts in parallel requires less compute than training the entire MoE on large datasets from the beginning (Sukhbaatar et al., 2024).

In this paper, we investigate how to effectively merge different domain expert models into a unified MoE model. The current state-of-the-art (SoTA) MoE merging approach, such as Branch-Train-Mix (BTX) (Sukhbaatar et al., 2024) assumes experts are branched from the same ancestor model and merges experts by simply unweighted averaging the non-FFN layers. However, as experts diverge in the parameter space, for example by branching from different ancestors or by training on aggressively different data, unweighted averaging may not effectively handle parameter interference such as sign conflicts (Yu et al., 2024; Yadav et al., 2024). As a result, the merged MoE may underperform and will require a significant amount of additional fine-tuning to recover in performance, which is both expensive and could be impractical when the experts' training data is not publicly available. Furthermore, existing MoE merging methods cannot be directly used to merge heterogeneous experts with different architectures, which could be the case in practice, as increasingly more experts

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 2315–2328
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

are provided by separate teams, such as CodeLlama (Roziere et al., 2023) and Olmo (Groeneveld et al., 2024). Therefore, it is still an open question how to effectively merge homogeneous and heterogeneous experts into an MoE combining the benefits of each.

To enable the use of diverse expert models, our work addresses the above limitations via new MoE merging methodologies for both homogeneous and heterogeneous experts. In summary, our work introduces three main contributions:

- We utilize advanced merging methods that address parameter interference, demonstrating their superiority over unweighted averaging in homogeneous expert merging, particularly in scenarios with limited resources for post-merging MoE fine-tuning.
- We propose a perplexity-based heuristic for routing token sequences to domain-specific experts in low-resource environments where MoE fine-tuning is not feasible.
- We develop a novel approach to merge experts with different architectures into a single MoE, which learns to route token sequences dynamically to the appropriate expert.

Through extensive experiments and ablation studies across benchmarks in mathematical reasoning, programming, and general knowledge, we show that our proposed methodologies outperform previous state-of-the-art methods and extend the practical applications of MoE merging.

2 Background and Related Work

2.1 Dense Model Merging

Dense merging methods combine multiple dense models into one to achieve diverse capabilities (Wortsman et al., 2022; Ilharco et al., 2022; Goddard et al., 2024; Jin et al., 2022; Matena and Raffel, 2022; Roberts et al., 2024). Most approaches focus on merging homogeneous dense models into another dense model. For example, average merging (Wortsman et al., 2022) averages model parameters, while task vector merging (Ilharco et al., 2022) adds the unweighted sum of task vectors (the difference between base and expert parameters) back to the dense model with scaling. Other work determines task vector weights instead of using an unweighted sum (Jin et al., 2022; Matena and Raffel, 2022). SoTA methods like Dare and Ties (Yadav et al., 2024; Yu et al., 2024) trim the task vector to resolve parameter interference: Dare trims the task vector randomly and rescales, while Ties sets vector parameters to zero by magnitude and adjusts signs to reduce conflicts.

In addition to homogeneous model merging, Roberts et al. (2024) propose merging heterogeneous models into a dense model using projectors, while Wan et al. (2024) apply knowledge distillation to fuse heterogeneous models. In this work, we introduce a more efficient method for merging experts with limited or no further fine-tuning and, unlike previous work focusing on dense models, we explore merging homogeneous and heterogeneous experts into an MoE model.

2.2 MoE Training and Merging

MoE architectures enable quicker inference with a certain parameter count by introducing Sparse MoE layers, where a router mechanism assigns tokens to the top-K expert FFNs (usually 1 or 2) in parallel (Fedus et al., 2022; Shazeer et al., 2017; Zhang et al., 2022). Most MoE training approaches, known as upcycling, train the entire model from scratch to handle multiple tasks (Komatsuzaki et al., 2022; Jiang et al., 2024; Dou et al., 2024; Dai et al., 2024). These methods first initialize the MoE model from a pretrained base model and then train it on the entire dataset. However, due to the costly communication between GPUs, the upcycling method introduces significant computational overhead (Sukhbaatar et al., 2024; Li et al., 2024b). To address this, methods like Branch-Train-Merge (BTM) (Gururangan et al., 2023; Li et al., 2022) average model outputs from different experts, while Branch-Train-Mix (BTX) (Sukhbaatar et al., 2024) branches the base model, trains each on different domains, and merges them into a unified MoE. BTX is shown to be more effective than BTM as well as dense CPT and MoE upcycling baselines. Another recent approach, Self-MoE (Kang et al., 2024), uses low-rank adaptation (LoRA) (Hu et al., 2021) to fine-tune experts on generated synthetic data (Liu et al., 2024b) and combines trained adapters into an MoE. To our knowledge, we are the first to introduce a framework for merging heterogeneous models into an MoE.

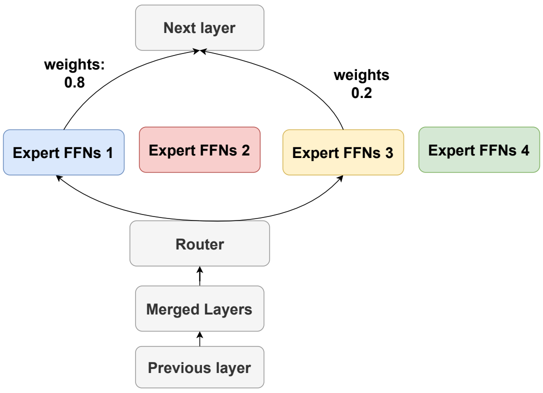


Figure 1: **Overview of the proposed MoE framework for homogeneous model merging**. We replace averaging with Dare or Ties merging to reduce parameter interference. Additionally, we introduce novel routing heuristics to enhance performance without fine-tuning.

3 Methodology

We define our research problem as follows: Given l dense expert models with parameters $[\theta_1, \theta_2, \ldots, \theta_l]$, each pretrained on different domains, we aim to propose an efficient merging method to combine these dense models into an MoE with parameters $\theta_m = \text{Merge}(\theta_1, \theta_2, \ldots, \theta_l)$, optimizing performance across all domains.

We now present our approach for MoE merging with homogeneous and heterogeneous expert models. First, for MoE merging with homogeneous experts (Section 3.1), we propose replacing existing averaging with more advanced merging methods to deal with parameter interference, and introduce sequence-level routing heuristics to enhance MoE performance without post-merge fine-tuning. Second, we introduce a novel framework for MoE merging with heterogeneous experts (Section 3.2), which uses projectors to unify expert inputs and outputs, and a sequence-level router.

3.1 Homogeneous Model Merging

First, we describe the basic merging setup (Section 3.1.1) and then summarize our extensions to resolve parameter interference (Section 3.1.2) and address the need for MoE fine-tuning (Section 3.1.3). The overall pipeline is visualized in Figure 1.

3.1.1 Merging Setup

Our merging setup is similar to the BTX (Sukhbaatar et al., 2024), where it merges all non-FFN layers (embedding, attention, normalization, and head) of experts by unweighted averaging and keeps the FFNs separate. As in standard MoE architectures, a router network, implemented as a Multilayer Perceptron (MLP), is inserted between the attention and FFN layers for token-level routing, selecting the top K (usually 1 or 2) experts for each layer among all l experts. The output of FFN layers $\text{FF}_{MoE}(v)$ of token embedding v is formulated as:

$$\text{FF}_{MoE}(v) = \sum_{i=1}^{K} \text{SoftMax}(\text{top-K}(\theta_r v))\text{FF}_i(v)$$

where θ_r is the parameter of the router network and $\text{FF}_i(v)$ is the output of each FFN experts for token v. After merging experts into a single MoE, BTX fine-tunes all parameters, including the router parameters on a mix of training data from all experts.

3.1.2 Addressing Parameter Interference

The major pitfall of the unweighted merging is that there exists parameter interference, as explored in the previous work on dense model merging (Yu et al., 2024; Yadav et al., 2024). As suggested in Figure 2, when influential parameters (large magnitude parameters) in the task vector merge with redundant parameters (small magnitude parameters) or parameters with sign conflict, simple averaging will output a small magnitude parameter, which may reduce the effect of the original task vector.

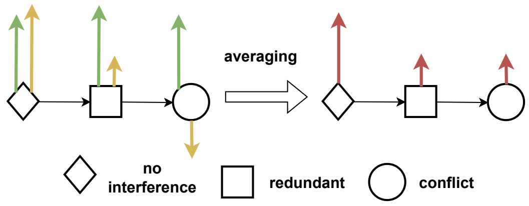


Figure 2: Different types of parameter interference and merged outputs produced by simple averaging.

In contrast to BTX, we mitigate model interference by employing previous SoTA methods in this MoE setup, namely Dare and Ties. First, we calculate the task vector $\tau_i = \theta_b - \theta_i$ with the base model parameter θ_b and the parameter θ_i for the model CPTed on domain i. For Ties merging, we first drop the bottom $(100 - p)\%$ of the redundant parameters (smallest magnitude) by resetting them to 0. For each parameter, we determine the sign with the highest total magnitude in all task vectors and sum all task vectors together to τ_m but only by keeping the parameter values whose signs are the same as the determined sign. For Dare merging, we randomly drop the $(100 - p)\%$ parameters. We rescale each task vector with $\tau_i = \frac{\tau_i}{0.01p}$. We sum all task vectors to τ_m. Finally, we add the summed task vector back to the base model with the scaling term λ and obtain the merged layer parameters: $\theta_m = \theta_b + \lambda \cdot \tau_m$. We expect that the drop operation in both methods will address the parameter interference issue, as revealed in dense model merging, and produce a consistent performance boost (Yu et al., 2024; Yadav et al., 2024).

Similar to BTX, after combining each expert model into an MoE, we fine-tune all parameters in the MoE in the fine-tuning stage. By addressing parameter interference, our approach achieves performance improvements over BTX especially in earlier stages of fine-tuning. Next, we describe how to further reduce the fine-tuning needs.

3.1.3 Reducing Fine-Tuning Needs

Fine-tuning MoEs is expensive due to the communication cost between GPUs (Sukhbaatar et al., 2024). Previous MoE merging methods require substantial fine-tuning of the MoE parameters to train the router network. In this section, we propose two techniques to reduce reliance on MoE fine-tuning, namely a perplexity-based routing and separating the attention layers.

The overall MoE pipeline after merging is illustrated in Figure 1, but we replace the router network with our routing heuristic to determine the expert selection. Additionally, we separate attention layers without merging them. For each input, the routing heuristic selects the appropriate experts and assigns their weights. The input is then processed by the chosen experts, and their outputs are combined using weights.

Routing Heuristics Our goal is to develop routing heuristics that replace the routing network without accessing the training data. We propose a sequence-level heuristics: perplexity (PPL) routing with only access to the inference sentence.

Our approach assesses the confidence of expert models by utilizing perplexity (PPL) to estimate their uncertainty. We then select the experts with the lowest PPL values, indicating higher confidence (Jelinek et al., 1977). Formally, with the inference input x_{inf} with t tokens and the expert parameter θ_i for expert i, we compute the PPL value $\text{PPL}(x_{inf}, \theta_i)$ as below:

$$\text{PPL}(x_{inf} \mid \theta_i) = \exp\left(-\frac{1}{t}\sum_{j=1}^{t} \log P(x_j \mid x_{<j}, \theta_i)\right)$$

where $P(x_j \mid x_{<j}, \theta_i)$ is the probability assigned by model θ_i on j-th token, given previous tokens.

Since a higher PPL indicates greater uncertainty, we use the reciprocal of PPL values to represent the model's confidence. With the top-K routing, the selected experts and their weights α can be computed as follows:

$$\alpha = \text{SoftMax}(\text{top-K}(\tfrac{1}{\text{PPL}(x_{inf}|\theta_1)}, \ldots, \tfrac{1}{\text{PPL}(x_{inf}|\theta_l)}))$$

Additionally, we also propose another routing heuristic based on the task vector and we present the details of this heuristic in Appendix C. With the routing heuristics and the corresponding computed weights from the heuristic, we will present the detailed merging process to form the MoE without further fine-tuning.

Separating attention layers We hypothesize that by merging attention layers, BTX creates inconsistency between the attention and FFN outputs. Specifically, the merged attention layers are influenced by all l task vectors from the dense experts, while the top-k routing method limits the FFN output to only k task vectors, leading to mismatched outputs. To address this, we consider keeping experts' attention layers as separate, similar to FFN. This ensures that both the attention and FFN layers come from the same expert, eliminating discrepancies from inconsistent task vector counts.

3.2 Heterogeneous Model Merging

This section describes how to merge models with different architectures into a unified MoE. Previous MoE merging techniques cannot be directly used in this setting, as it is not possible to merge non-FFN networks layer by layer when experts have different numbers of layers or different layer shapes. To resolve this challenge, we propose a new merging method, which introduces projector layers and sequence-level routing as shown in Figure 3.

First, we denote the hidden dimension of all l experts as $d_1, d_2 \ldots, d_l$, and the maximum dimension among them is d_m. Suppose that we have a vocabulary $\mathcal{V}$ and an input sentence with tokens $[v_1, v_2 \ldots, v_t]$. For the shared embedding layer $\mathcal{M}_e$, it maps the token v_i in the sentence to embedding $e_i \in \mathbb{R}^{d_m}$ and the shared head layer is the network $\mathcal{M}_h : \mathbb{R}^{d_m} \to \mathbb{R}^{|\mathcal{V}|}$, which maps the weighted sum of projectors back to the probability distribution of tokens in the vocabulary. The

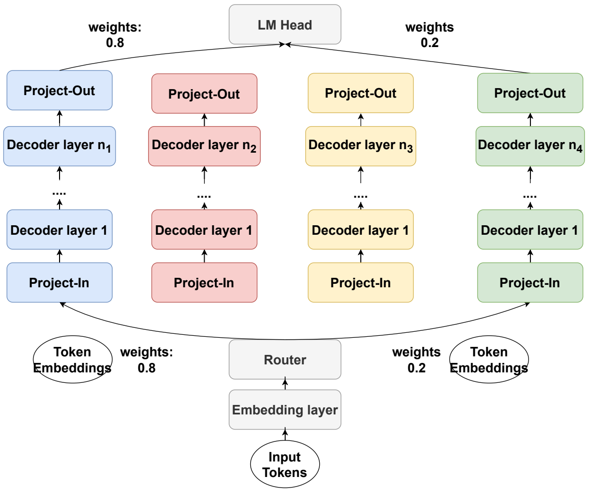


Figure 3: **Overview of the proposed MoE framework for heterogeneous experts.** Each color represents one heterogeneous expert. $n_1, \cdots, n_4$ refers to the number of layers in each expert.

embedding and head layer parameters are initialized from an averaging of the embedding and head layers of each expert. For experts with a hidden dimension less than d_m, we add padding zeros for their embedding and head layers before averaging.

Since we do not merge attention layers due to heterogeneous experts, all tokens must be routed to the same expert. Otherwise, the attention layers cannot perform self-attention, as they require access to every token. Hence, we average the token embeddings and use the router to perform the sequence-level routing. Formally, for top-K routing with router parameters θ_r, the router computes the model weights as follows:

$$\alpha = \text{SoftMax}(\text{top-K}(\theta_r \text{avg}(e_1, e_2, \ldots, e_t)))$$

For projectors: Proj-in and Proj-out, for each expert, randomly initialized MLP layers, they project the embedding outputs to the dimension of each expert, and project the expert output back to the maximum dimension. For i-th expert, we define:

$$\text{Proj-in layer} : \mathbb{R}^{d_m} \rightarrow \mathbb{R}^{d_i}, \quad \text{Proj-out layer} : \mathbb{R}^{d_i} \rightarrow \mathbb{R}^{d_m}$$

After using the selected K experts to process the input sequences and translating their outputs to the representation r_i via the Proj-out layer (with dimension d_m), we combine the representations using the router's weights: $\sum_{i=1}^{K} \alpha_i r_i$. The combined representation is then fed into the head layer to obtain the token probabilities.

After merging the heterogeneous experts into the MoE model, we choose an arbitrary tokenizer from one expert, following previous work (Roberts et al., 2024) and fine-tune all parameters.

4 Experiments Setup and Model Analysis

Through our extensive empirical analysis, we aim to evaluate our frameworks in the settting of homogeneous experts and heterogeneous experts.

4.1 Evaluation Dataset

We evaluate our proposed methodology on 6 datasets from three domains, as in the previous work (Sukhbaatar et al., 2024). For math reasoning, we choose GSM8K (8-shot) and MATH (4-shot) (Cobbe et al., 2021; Hendrycks et al., 2021). For code generation, we choose MBPP (0-shot) and HumanEval (0-shot) (Chen et al., 2021; Austin et al., 2021). For world knowledge, we choose Natural Questions (NQ, 5-shot) and TriviaQA (5-shot) (Kwiatkowski et al., 2019; Joshi et al., 2017).

4.2 Model Configuration

This section describes the base model and experts discussed in our experiments:

- **Base Model (Base-1B)**: This is our base model with 1B parameters and Llama-like architecture. We pretrain Base-1B from scratch with 250 billion (250B) tokens from the following datasets from the RedPajama dataset (Together Computer, 2023): Arxiv, CommonCrawl, C4, StackExchange data and the first half of the WikiPedia data in the RedPajama dataset.

- **Math Expert**: We CPT the Base model on the OpenWebMath data for 100B tokens (Paster et al., 2023).

- **Code Expert**: We use the GitHub data in RedPajama to CPT the Base model for 100B tokens.

- **Knowledge Expert**: We CPT the Base-1B model on the second half of the Wikipedia data in the RedPajama dataset for 100B tokens.

- **Math TinyLlama and Math Olmo**: We CPT the TinyLlama-1.1B model (Zhang et al., 2024) and Olmo-1B model (Groeneveld et al., 2024) on the same data mixture of the Math Expert.

- **Mixture of Experts (MoE)**: For homogeneous model merging, we combine three experts (Math Expert, Code Expert, Knowledge Expert) and one base model (Base-1B) into an MoE. For heterogeneous merging, we combine Code Expert, Knowledge Expert, Base-1B, and either Math TinyLlama or Math Olmo. MoE fine-tuning is performed on all data sources from the base and expert models, using an additional 40B tokens. Detailed sampling ratios for pretraining and fine-tuning are provided in Appendix B.

We present the details of model architecture for each expert in Appendix A.

4.3 Baseline Methods

To demonstrate the effectiveness of our methodology, we compare the performance of the merged 4-expert MoE models with several other baselines.

- **Base & Experts**: The dense base and expert models in Section 4.2.
- **BTX** (Sukhbaatar et al., 2024): The MoE model derived from the BTX pipeline with average merging and post-merge fine-tuning.
- **Random Routing**: The average merged MoE with randomly initialized router.
- **Router Fine-tuning**: The MoE model derived from the BTX pipeline but only fine-tune the parameters in the router network.
- **3-expert MoE**: To demonstrate the functionality of Math Olmo or TinyLlama in heterogeneous expert merging, we prepare 3-expert MoE models (Base, Knowledge Expert, Code Expert), fine-tuned either on the full data source (including math) or only on code- and knowledge-related data. We merge these models using the BTX method, naming them **3-expert MoE (same data)** and **3-expert MoE (w/o math)**.
- **Dare Dense** (Yu et al., 2024), **Ties Dense** (Yadav et al., 2024): Advanced dense model merging method. We apply Dare or Ties to merge four LMs to one dense model.

The details of the model configuration of the baseline methods are included in Appendix A.

4.4 Similarity of Model Parameters

Before presenting the performance of our proposed methodology, we first analyze the similarities in model parameters across different experts to demonstrate the necessity for alternatives to average merging. Previous work assumes that parameters in attention layers are less domain-specialized, leading to the use of simple averaging when combining non-FFN layers (Sukhbaatar et al., 2024). Our analysis aims to verify whether this assumption holds true for experts trained on different domains.

To quantify the degree of domain specialization in the model layers, we first extract the task vectors for each layer from our Math and Code Expert models. We then concatenate the task vectors from the attention layers and FFNs into two long vectors. Next, we calculate the cosine similarity between the two concatenated task vectors. The cosine similarity for the task vectors of the FFNs and self-attention layers is visualized separately in Figure 4.

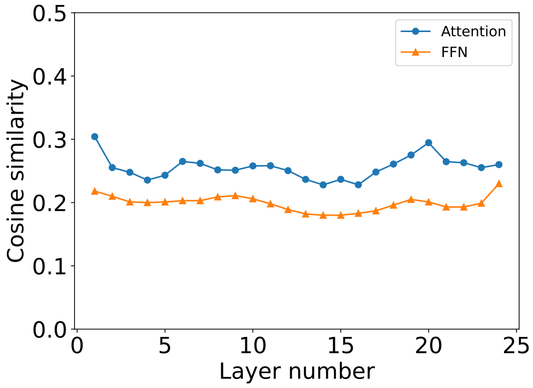


Figure 4: **Similarity of task vector for attention and FFNs layers for Math and Code Expert experts.** We average the similarity of attentions or FFNs in one decoder layers as the overall similarity for each layer.

We observe that the task vectors from both layers exhibit low similarity, suggesting that the assumption of similar attention layers does not consistently hold and parameter interference may occur. This analysis demonstrates the need for more advanced merging methods, rather than averaging, for homogeneous model merging.

5 Results

5.1 Homogeneous Model Merging

5.1.1 Averaging vs. Dare / Ties

Replacing simple averaging with Dare or Ties merging obtains better performance. In this section, we demonstrate the superiority of our proposed Ties and Dare merging MoE over the BTX merging method. We present the performance of MoE models with **Dare merging** or **Ties merging** on non-FFN layers and other baselines in Table 1.

The details of training cost for each method are presented in Table 6 in Appendix.

Method	MBPP	HumanEval	MATH	GSM8K	NQ	TriviaQA	Avg.
Dense Model							
Base-1B	4.60	3.04	2.42	1.44	**6.61**	26.72	7.47
Code Expert	**10.2**	**8.53**	2.42	2.57	3.11	16.70	7.26
Math Expert	9.80	6.71	**7.81**	**6.36**	5.48	19.86	**9.34**
Knowledge Expert	3.60	4.26	2.62	2.04	5.65	**28.71**	7.81
MoE Merging							
Random Routing	4.00	6.10	2.78	2.05	4.86	21.75	6.92
Router Fine-tuning	3.60	6.71	2.42	2.96	5.82	25.98	7.92
BTX merging	12.40	11.58	6.74	7.73	**6.78**	25.10	11.72
Ties merging	14.20	**11.98**	6.74	7.81	6.72	27.66	12.52
Dare merging	**14.20**	10.98	**6.82**	**7.96**	6.50	**30.68**	**12.86**
MoE from Scratch							
MoE Upcycling	18.40	12.20	7.80	12.21	8.37	37.33	16.05

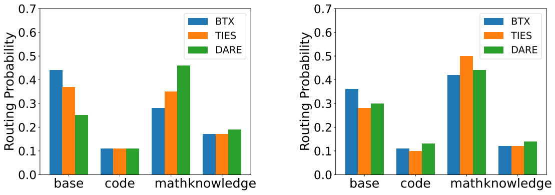
Table 1: **Performance of proposed Dare and Ties merged MoE and other baselines across six datasets.** The best performance of Dense and MoE model is marked in bold. Results of Dare and Ties merged MoE outperform the BTX MoE and other baseline methods.

From Table 1, we see that individual experts generally achieve the best performance in their respective domains, as expected. However, CPTed Expert models experience catastrophic forgetting. For instance, both Code and Math Expert perform worse than Base-1B on the TriviaQA and NQ datasets.

The results in Table 1 show that using Ties or Dare merging significantly improves MoE performance over the BTX pipeline across almost all datasets, with a relative improvement of 6.94% and 9.72% in average performance. This suggests that advanced merging methods reduce weight interference and enhance performance.

As a reference, we include the results of MoE sparse upcycling (Komatsuzaki et al., 2022) in the last row of Table 1. This approach initializes the MoE model by creating four identical copies of the FFN layers from the base model and then CPT on the same 340B tokens used in our pipeline. However, we do not directly compare our results with the upcycling method, as it involves pretraining the entire MoE on all data, incurring significantly higher costs. We also visualize the average performance for each merging method with different fine-tuning token numbers in Figure 10 in Appendix D. In Figure 10, we observe that the Dare and Ties merging MoE models consistently outperform the BTX merging MoE throughout fine-tuning, especially in the earlier stages of fine-tuning.

MoE with Dare or Ties merging routes more tokens to domain experts. To further explore the effectiveness of Dare and Ties merging MoE, we evaluate MoEs on multiple benchmarks and calculate the routing probability averaged from each layer and token. We visualize the routing probability of each method of two math datasets (MATH and GSM8K) in Figure 5 and for other datasets, we put the results in Figure 7 in Appendix D.



(a) GSM8K

(b) MATH

Figure 5: Routing probability of experts on GSM8K and MATH for different merging methods.

Compared to MoEs with BTX merging, where the base model accepts the most routing decisions, the Dare and Ties merging method routes tokens to domain experts more frequently, as suggested in Figure 5. For example, for the GSM8K dataset, the routing probability for math expert increases from 0.28 to 0.35 or 0.46 when replacing simple averaging with the Ties or Dare merging. This finding suggests that the more effective MoE with the more advanced merging method should be attributed to more optimized routing decisions.

5.1.2 Merging without Fine-tuning

In this part, we will evaluate our proposed routing heuristics in Section 3.1.3 for MoE without fine-tuning. Before we evaluate the overall performance of each benchmark, we will first examine the routing decision with our proposed heuristics. We present the routing probability for PPL routing heuristics for each dataset in Table 2.

Benchmark	Base	Code	Math	Knowledge
GSM8K	23%	2%	**43%**	32%
MATH	22%	2%	**49%**	27%
MBPP	19%	22%	**44%**	15%
HumanEval	5%	43%	**45%**	7%
NQ	43%	4%	10%	**43%**
TriviaQA	50%	0%	0%	**50%**

Table 2: **Routing probability of PPL routing for each dataset.** The largest probability are in bold, and the second-largest are underlined.

Routing heuristic effectively assigns tokens to the corresponding experts. Table 2 demonstrates that PPL routing generally achieves the desired routing patterns, effectively directing inputs from a specific domain to the specialized expert models, except in the case of the MBPP dataset. Since our heuristics rely solely on inference inputs without fine-tuning, they can be considered reliable strategies. We also visualize the routing probability for both PPL and task vector routing heuristics for each dataset in Figure 9 in Appendix D. We find that PPL routing consistently produces better results than the task vector routing.

Next, we evaluate the performance on each dataset with different combinations of merging methods and routing heuristics, compared to the baseline methods. We prepare three dense fine-tuning baselines: **Dare Dense**, **Ties Dense** and **Random Routing** (details in Section 4.3). We also evaluate the ablation methods: merging attention layers without separation and task vector routing. We present the results of each method across datasets in Table 3. The details of training cost for each method are presented in Table 7 in Appendix.

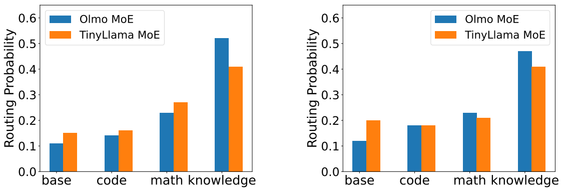
Merging	Routing	MBPP	HumanEval	MATH	GSM8K	NQ	TriviaQA	Avg.
				Dense Merging				
Dare	N/A	6.20	6.70	2.22	2.27	4.80	20.45	7.11
Ties	N/A	6.00	6.70	2.48	2.19	3.62	20.86	6.98
				MoE Merging				
Merge attention	random	4.00	6.10	2.78	2.05	4.86	21.75	6.92
Merge attention	task vector	6.60	4.87	3.06	1.44	**6.05**	21.39	7.24
Merge attention	PPL	6.40	4.87	2.86	1.13	5.93	22.71	7.32
Separate attention	task vector	4.00	7.32	**2.98**	2.5	5.37	20.11	7.05
Separate attention	PPL	**6.80**	**7.92**	2.88	**2.95**	4.74	**23.21**	**8.08**

Table 3: **Performance of proposed merging and routing methods for MoE without substantial fine-tuning and other baselines across six datasets.** Separating attention layers and perplexity routing heuristics get the best average performance.

Proposed MoE method without fine-tuning outperforms the dense merging baseline. From Table 3, we observe that using the PPL routing heuristic and separating attention layers achieves the best average results among all baseline methods. Compared to Random Routing and the SoTA dense merging method (Dare), our best method - PPL routing + separating attention layers - yields relative improvements of 16.8% and 13.6%, respectively. The superior performance of PPL routing aligns with Figure 9 in Appendix D, where PPL routing more accurately directs input to the appropriate experts. Moreover, the better results of separating attention layers support our expectation that this approach resolves the inconsistency of task vector counts, as discussed in Section 3.1.3.

5.2 Heterogeneous Model Merging

MoE merged with heterogeneous models outperforms the corresponding experts. After showing the superiority of our homogeneous model merging method, our next question is whether the proposed heterogeneous expert merging is also effective. We present the performance of the dense, MoE and baseline methods in Table 4. The details of training cost for each method are presented in Table 8 in Appendix.



(a) GSM8K

(b) MATH

Figure 6: Routing probability of experts on GSM8K and MATH for the MoE w/ Olmo and MoE w/ TinyLlama.

Method	MBPP	HumanEval	MATH	GSM8K	NQ	TriviaQA	Avg.
			Dense Model				
Base-1B	4.60	3.04	2.42	1.44	6.61	26.72	7.47
Base TinyLlama	5.40	5.27	2.26	2.2	8.53	34.27	9.66
Base Olmo	2.80	2.64	2.46	2.42	6.16	29.21	7.62
Code Expert	10.20	8.53	2.42	2.57	3.11	16.7	7.26
Math TinyLlama	15.60	9.76	4.18	5.91	6.05	21.12	10.44
Math Olmo	0.00	0.00	4.82	5.08	3.61	11.25	4.13
Knowledge Expert	3.60	4.26	2.62	2.04	5.65	28.71	7.81
			Homogeneous Expert Merging				
3-expert MoE (same data)	9.14	10.8	4.42	5.16	6.95	26.78	10.54
3-expert MoE (w/o math)	12.00	9.76	2.38	1.74	6.22	**33.20**	10.88
			Heterogeneous Expert merging				
(Ours) MoE w/ Math Olmo	13.60	10.98	4.86	6.14	5.43	26.01	11.17
(Ours) MoE w/ Math TinyLlama	**15.80**	**11.59**	**5.42**	**6.29**	**8.25**	32.71	**13.34**

Table 4: **Performance of proposed heterogeneous merged MoE and other baselines.** The merged MoE is comparable or outperform the dense or 3-expert baselines on the benchmark from the corresponding domain.

Table 4 shows that our merged MoE models are comparable to or outperform the domain expert models in their respective domains. For instance, the MoE merged with Math Olmo and Math TinyLlama achieves 6.14% and 6.29% accuracy on GSM8K, compared to 5.91% and 5.08% for their dense counterparts. On average, our MoEs with Olmo and TinyLlama improves performance by 43.02% and 27.78% relative to the best dense experts, respectively. Both MoEs with heterogeneous experts also outperform the 3-expert MoE baseline, particularly in math, highlighting the effectiveness of including math experts in the pipeline.

MoE merged with heterogeneous experts show the desired routing patterns in most cases. We also perform a similar routing analysis as described in Section 5.1.1. We visualize the routing probability of two MoEs when evaluating on GSM8K and MATH datasets in Figure 6 and for other datasets, we visualize the results in Figure 8 in Appendix D.

As shown in Figures 6 and 8, most tokens in the coding and knowledge datasets are routed to the corresponding experts. However, unlike homogeneous model merging where the math expert has the highest routing probability for math datasets, Math Olmo or Math TinyLlama ranks second. This discrepancy is likely due to the difference in embedding outputs between the MoE and expert models. Since the MoE's embedding layer is merged from 3 Expert models and 1 other model, its output is closer to that of the Expert models, making the router more likely to select them. Adding a load balancing loss is a possible solution to address this issue (Sukhbaatar et al., 2024; Fedus et al., 2022), ensuring a more uniform routing distribution. We leave this for future exploration

6 Conclusion

In this paper, we propose novel methods to address challenges in the current MoE merging literature. For homogeneous experts, we replace average merging in non-FFN layers with more advanced methods to reduce parameter interference. We also explore merging models into an MoE without post-merge fine-tuning. For heterogeneous experts, we introduce a method using projectors and sequence-level routing networks to combine models with different architectures. Extensive empirical evaluations show that our approach significantly improves MoE performance across multiple datasets.

7 Limitation

One of the limitations of the proposed merging methods with heterogeneous experts is that the merged MoE model has more parameters when the BTX merging, since we do not merge the attention layers. For example, for our 4 × 1B Expert MoE, the total parameter number is about 3.7 billion due to the non-FFNs layer merging but the total parameter number of the MoE after the heterogeneous merging method is near 4 billion. More parameters represent more costly fine-tuning and inference.

For our homogeneous merging method, we replace simple averaging with a more advanced merging method: Dare and Ties and fine-tune MoE models. There are still other merging methods, such as fisher merging (Matena and Raffel, 2022) or Regmean (Jin et al., 2022) methods. However, in the Ties and Dare paper (Yadav et al., 2024; Yu et al., 2024), they have demonstrated the superiority of proposed merging methods over Regmean and fisher merging, so we leave the exploration of other merging methods to future work.

Moreover, using routing heuristics to process the input sequence introduces additional inference costs, as we first need to use the expert model to calculate the perplexity (PPL) or gradient. However, our routing heuristic requires only one additional forward pass, and considering the multiple forward passes during inference (forward pass number = the generate token number), the computational overhead for our method to enhance MoE performance without fine-tuning is minimal.

For all MoE fine-tuning, we utilize only the cross-entropy loss to do the auto-regression on the training data. Previous works showed that the load-balancing loss (Fedus et al., 2022; Sukhbaatar et al., 2024) may be beneficial to resolve the "dead" experts. From our routing analysis for the merged MoEs, we observe that merging with homogeneous experts gets the desirable patterns, where most tokens in one specific domain are gated to the corresponding expert. However, for heterogeneous experts, due to the different architecture and tokenizer of the math expert, the math expert does not get the highest routing probability in evaluating on GSM8K and MATH datasets. For the next step, we may need to add the load balancing loss for the fine-tuning of MoE with heterogeneous experts to develop more robust models (Zhou et al., 2024a) and observe whether the routing patterns are more efficient.

Due to limitations of computation resources, we only experimented with three domains and 1b LLMs. Incorporating larger models and more domains, such as legal, medical, or multilingual, can benefit future studies. Furthermore, our method can be extended to multimodal MoE by incorporating vision audio or graph experts (Wang et al., 2024b,a; Li et al., 2024a; Zhu et al., 2024).

In addition to directly merging models with different architectures with additional projectors, there is another direction to first distill the knowledge of experts to student models with the same architecture (Wan et al., 2024; Zhou and Ai, 2024; Li et al., 2025; Zhou et al., 2023, 2024b) and merge student models together to an MoE. We leave the exploration of this direction to future work.

Acknowledgments

We would like to thank the anonymous reviewers as well as Saleh Soltan, He Xie, Venkatesh Elango, Wael Hamza, Paiheng Xu, and Xiyao Wang for providing helpful comments and suggestions.

References

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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

Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. 2023. Chatlaw: Open-source legal large language model with integrated external knowledge bases. *arXiv preprint arXiv:2306.16092*.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, et al. 2024. Loramoe: Alleviating world knowledge forgetting in large language models via moe-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1932–1945.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vlad Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024. Arcee's mergekit: A toolkit for merging large language models. *arXiv preprint arXiv:2403.13257*.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. 2024. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*.

Suchin Gururangan, Margaret Li, Mike Lewis, Weijia Shi, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. 2023. Scaling expert language models with unsupervised domain discovery. *arXiv preprint arXiv:2303.14177*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

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. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

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

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Junmo Kang, Leonid Karlinsky, Hongyin Luo, Zhen Wang, Jacob Hansen, James Glass, David Cox, Rameswar Panda, Rogerio Feris, and Alan Ritter. 2024. Self-moe: Towards compositional large language models with self-specialized experts. *arXiv preprint arXiv:2406.12034*.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. 2022. Sparse upcycling: Training mixture-of-experts from dense checkpoints. *arXiv preprint arXiv:2212.05055*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. 2022. Branch-train-merge: Embarrassingly parallel training of expert language models. *arXiv preprint arXiv:2208.03306*.

Yunxin Li, Shenyuan Jiang, Baotian Hu, Longyue Wang, Wanqi Zhong, Wenhan Luo, Lin Ma, and Min Zhang. 2024a. Uni-moe: Scaling unified multimodal llms with mixture of experts. *arXiv preprint arXiv:2405.11273*.

Zongxia Li, Ishani Mondal, Huy Nghiem, Yijun Liang, and Jordan Lee Boyd-Graber. 2024b. PEDANTS: Cheap but effective and interpretable answer equivalence. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9373–9398, Miami, Florida, USA. Association for Computational Linguistics.

Zongxia Li, Xiyang Wu, Hongyang Du, Huy Nghiem, and Guangyao Shi. 2025. Benchmark evaluations, applications, and challenges of large vision language models: A survey.

Xiaoyu Liu, Paiheng Xu, Junda Wu, Jiaxin Yuan, Yifan Yang, Yuhang Zhou, Fuxiao Liu, Tianrui Guan, Haoliang Wang, Tong Yu, et al. 2024a. Large language models and causal inference in collaboration: A comprehensive survey. *arXiv preprint arXiv:2403.09606*.

Xiaoyu Liu, Jiaxin Yuan, Yuhang Zhou, Jingling Li, Furong Huang, and Wei Ai. 2024b. Csrec: Rethinking sequential recommendation from a causal perspective. *arXiv preprint arXiv:2409.05872*.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*.

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

OpenAI. 2023. Gpt-4 technical report.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. 2023. Openwebmath: An open dataset of high-quality mathematical web text. *arXiv preprint arXiv:2310.06786*.

Nicholas Roberts, Samuel Guo, Zhiqi Gao, Satya Sai Srinath Namburi GNVV, Sonia Cromp, Chengjun Wu, Chengyu Duan, and Frederic Sala. 2024. Pretrained hybrids with mad skills. *arXiv preprint arXiv:2406.00894*.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Yijia Shao, Yucheng Jiang, Theodore A Kanell, Peter Xu, Omar Khattab, and Monica S Lam. 2024. Assisting in writing wikipedia-like articles from scratch with large language models. *arXiv preprint arXiv:2402.14207*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Rozière, Jacob Kahn, Daniel Li, Wen tau Yih, Jason Weston, and Xian Li. 2024. Branch-train-mix: Mixing expert llms into a mixture-of-experts llm.

Together Computer. 2023. Redpajama: an open dataset for training large language models.

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

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024. Knowledge fusion of large language models. *arXiv preprint arXiv:2401.10491*.

Xiyao Wang, Jiuhai Chen, Zhaoyang Wang, Yuhang Zhou, Yiyang Zhou, Huaxiu Yao, Tianyi Zhou, Tom Goldstein, Parminder Bhatia, Furong Huang, and Cao Xiao. 2024a. Enhancing visual-language modality alignment in large vision language models via self-improvement.

Xiyao Wang, Yuhang Zhou, Xiaoyu Liu, Hongjin Lu, Yuancheng Xu, Feihong He, Jaehong Yoon, Taixi Lu, Gedas Bertasius, Mohit Bansal, et al. 2024b. Mementos: A comprehensive benchmark for multimodal large language model reasoning over image sequences. *arXiv preprint arXiv:2401.10529*.

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. 2022. 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.

Chengyue Wu, Yukang Gan, Yixiao Ge, Zeyu Lu, Jiahao Wang, Ye Feng, Ping Luo, and Ying Shan. 2024. Llama pro: Progressive llama with block expansion. *arXiv preprint arXiv:2401.02415*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2024. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36.

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

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

Xiaofeng Zhang, Yikang Shen, Zeyu Huang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2022. Mixture of attention heads: Selecting attention heads per token.

Yuhang Zhou and Wei Ai. 2024. Teaching-assistant-in-the-loop: Improving knowledge distillation from imperfect teacher models in low-budget scenarios. *arXiv preprint arXiv:2406.05322*.

Yuhang Zhou, Suraj Maharjan, and Beiye Liu. 2023. Scalable prompt generation for semi-supervised learning with language models. *arXiv preprint arXiv:2302.09236*.

Yuhang Zhou, Paiheng Xu, Xiaoyu Liu, Bang An, Wei Ai, and Furong Huang. 2024a. Explore spurious correlations at the concept level in language models for text classification.

Yuhang Zhou, Jing Zhu, Paiheng Xu, Xiaoyu Liu, Xiyao Wang, Danai Koutra, Wei Ai, and Furong Huang. 2024b. Multi-stage balanced distillation: Addressing long-tail challenges in sequence-level knowledge distillation. *arXiv preprint arXiv:2406.13114*.

Jing Zhu, Yuhang Zhou, Shengyi Qian, Zhongmou He, Tong Zhao, Neil Shah, and Danai Koutra. 2024. Multimodal graph benchmark. *arXiv preprint arXiv:2406.16321*.

A Implementation Details

For our Base-1B models, we utilize the Llama-2 architecture (Wu et al., 2024) with layer number 24 and hidden dimension 2048. The open-source TinyLlama-1.1B model contains 22 layers and the hidden dimension is 2048. For the open-source Olmo-1B model, it has 16 layers and the hiddn dimension is 2048.

In our experiments, we use top-2 routing for MoE models. For Dare-merging and Ties merging (both dense and MoE), we set the scaling term λ to $\frac{1}{3}$ and the retain ratio p of the model parameters of two methods are set to 80% to gain the optimal performance, according to our preliminary exploration. For inference, we set the temperature to 0.0 for greedy decoding, and the maximal number of generated tokens is 512. For CPT and fine-tuning of MoE and dense models, we set the learning rate to 1e-5 and the weight decay is 0.01.

B Data mixture

In Table 5, we present the data ratios to CPT or fine-tune the dense or MoE models. For fine-tuning the MoE model, we sample datasets that are used to train all experts and the base model with the same probabilities as described in Sukhbaatar et al. (2024).

	Base	Math	Code	Knowledge	Finetune MoE
Wiki1	0.85%	0.17%	0.17%	8.00%	1.11%
Wiki2	0.00%	0.00%	0.00%	8.00%	0.82%
Arxiv	9.37%	1.87%	1.87%	7.94%	3.94%
CommonCrawl	27.92%	5.58%	5.58%	23.65%	11.74%
C4	54.60%	10.93%	10.93%	46.26%	22.97%
StackExchange	7.26%	1.45%	1.45%	6.15%	3.05%
Open Web Math	0.00%	80.00%	0.00%	0.00%	24.13%
GitHub	0.00%	0.00%	80.00%	0.00%	32.25%

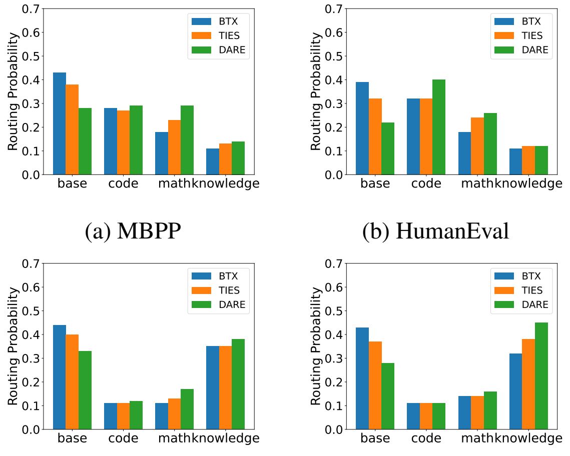
Table 5: Data source and weights for CPT or fine-tune MoE or dense models. Wiki1 represents the first half of Wikipedia data for pretraining the base model and Wiki2 represents the second half of Wikipedia data for CPT the knowledge expert.

C Task Vector Routing Heuristic

Our second approach is to identify the input domain and assign the input to experts trained in that domain. The core idea is that an expert's task vector, defined as the difference between its parameters and the base model, represents the cumulative gradient of the base model on the expert's training data. For a given input, we first compute the base model's gradient on that input and compare it to the task vectors of each expert. A higher similarity between the gradient and a task vector suggests the input is closer to the expert's training data.

With the task vectors $\tau_1, \tau_2, \ldots, \tau_l$ for l experts and inference input x_{inf}, the loss function $\mathcal{L}$ and the base model parameters θ_b, we first compute the gradient (g_{inf}) of the loss function with respect to the base model parameters as: $g_{inf} = \nabla_{\theta_b}\mathcal{L}(x_{inf})$.

The routing heuristic decides the experts and



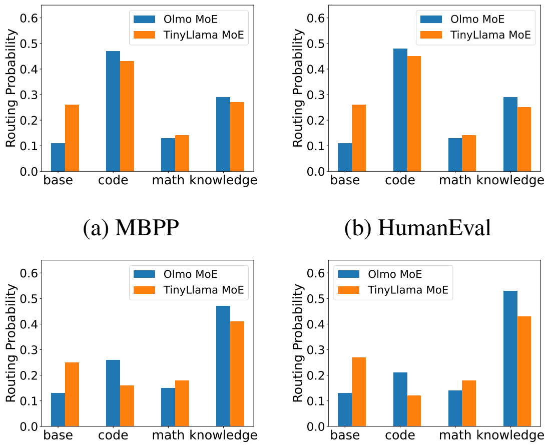
(a) MBPP

(b) HumanEval

(c) Natural Questions

(d) TriviaQA

Figure 7: Routing probability of experts on MBPP, HumanEval, Natural Questions and TriviaQA for different merging methods.



(a) MBPP

(b) HumanEval

(c) Natural Questions

(d) TriviaQA

Figure 8: Routing probability of experts on MBPP, HuamnEval, Natural Questions and TriviaQA for the MoE w/ Olmo and MoE w/ TinyLlama.

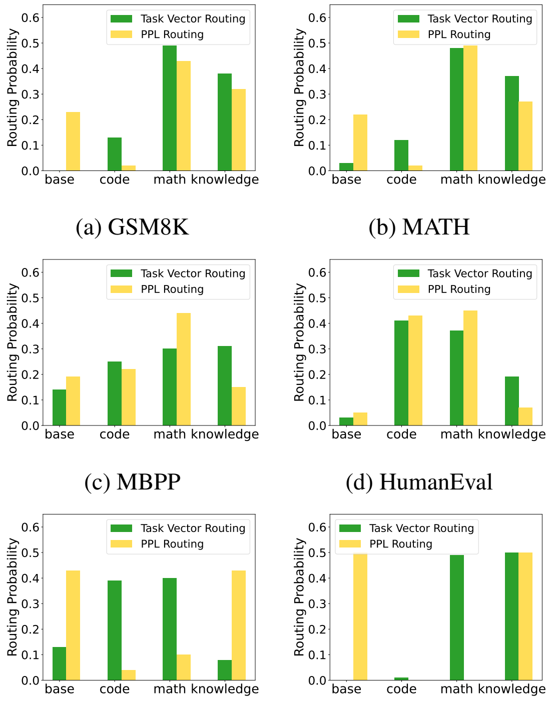
weights with the cosine similarity (Sim) as below:

$$\alpha = \text{SoftMax}(\text{top-K}(\text{Sim}(g_{inf}, \tau_1), \ldots, \text{Sim}(g_{inf}, \tau_l)))$$

D Supplementary Results

In this section, we present the supplementary analysis of the routing probability for each research question.

For the calculation of training cost for each method, we will use the product of the number of model parameters and the number of training tokens as a metric for training cost. We present the training costs for each method featured in Tables 1, 3, and 4.



(a) GSM8K

(b) MATH

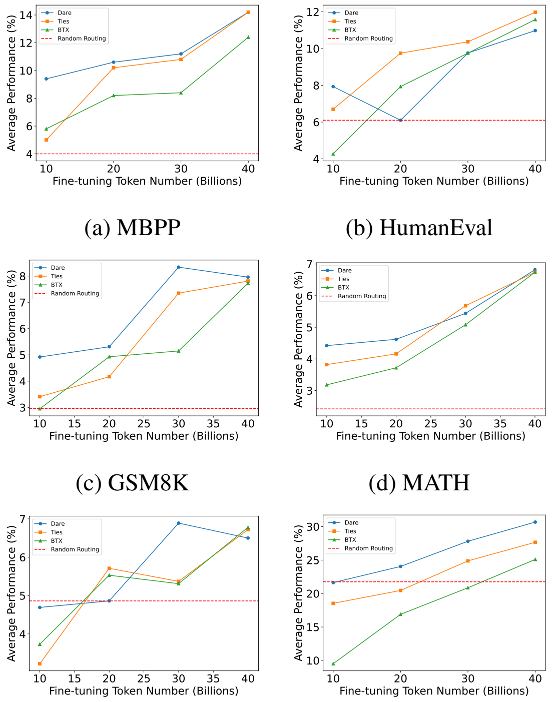
(c) MBPP

(d) HumanEval

(e) Natural Questions

(f) TriviaQA

Figure 9: Routing probability of tow routing heuristics for each dataset.



(a) MBPP

(b) HumanEval

(c) GSM8K

(d) MATH

(e) Natural Questions

(f) TriviaQA

Figure 10: Performance with varied fine-tuning token numbers across different datasets.

Method	Training Cost (# B parameters × # B tokens)
Base-1B	0
Code Expert	100
Math Expert	100
Knowledge Expert	100
Random Routing	300
Router Fine-Tuning	300
BTX Merging	448 (3 × 100 + 3.7 × 40)
Ties Merging	448
Dare Merging	448
Model Upcycling	1258 (3.7 × 340)

Table 6: Training cost of methods in Table 1

Method	Training Cost (# B parameters × # B tokens)
Dare	100
Ties	100
Merge Attention	100
Separate Attention	100

Table 7: Training cost of methods in Table 3

Method	Training Cost (# B parameters × # B tokens)
Base-1B	0
Base TinyLlama	0
Base Olmo	0
Code Expert	100
Math TinyLlama	100
Math Olmo	100
Knowledge Expert	100
3-expert MoE	312 (2 × 100 + 2.8 × 40)
(Ours) MoE w/ Math Olmo	448
(Ours) MoE w/ Math TinyLlama	448

Table 8: Training cost of methods in Table 4