The Thirty-Ninth AAAI Conference on Artificial Intelligence (AAAI-25)

Channel Merging: Preserving Specialization for Merged Experts

Mingyang Zhang[1], **Jing Liu**[2], **Ganggui Ding**[1], **Linlin Ou**[3], **Xinyi Yu**[3],
Bohan Zhuang[1*]
[1] Zhejiang University
[2] Monash University
[3] Zhejiang University of Technology

Abstract

Lately, the practice of utilizing task-specific fine-tuning has been implemented to improve the performance of large language models (LLM) in subsequent tasks. Through the integration of diverse LLMs, the overall competency of LLMs is significantly boosted. Nevertheless, traditional ensemble methods are notably memory-intensive, necessitating the simultaneous loading of all specialized models into GPU memory. To address the inefficiency, model merging strategies have emerged, merging all LLMs into one model to reduce the memory footprint during inference. Despite these advances, model merging often leads to parameter conflicts and performance decline as the number of experts increases. Previous methods to mitigate these conflicts include post-pruning and partial merging. However, both approaches have limitations, particularly in terms of performance and storage efficiency when merged experts increase. To address these challenges, we introduce Channel Merging, a novel strategy designed to minimize parameter conflicts while enhancing storage efficiency. This method clusters and merges channel parameters based on their similarity to form several groups offline. By ensuring that only highly similar parameters are merged within each group, it significantly reduces parameter conflicts. During inference, we can instantly look up the expert parameters from the merged groups, preserving specialized knowledge. Our experiments demonstrate that Channel Merging consistently delivers high performance, matching unmerged models in tasks like English and Chinese reasoning, mathematical reasoning, and code generation. Moreover, it obtains results comparable to model ensemble with just 53% parameters when used with a task-specific router.

Introduction

Recent advancements in large language models (LLMs) such as LLaMA (Touvron et al. 2023) and Mistral (Jiang et al. 2023a) have significantly pushed the boundaries of artificial intelligence, achieving near-human performance across various general tasks. Despite these achievements, a notable performance gap remains in specialized domains such as coding and mathematics. Addressing this gap, many LLMs undergo task-specific fine-tuning to enhance their capabilities within these targeted areas (Zhou and Yuqi 2024; Yu et al. 2023; Wu et al. 2023). In multi-task scenarios, it is common to ensemble LLMs specialized in different tasks to optimize performance. However, as shown in Figure 1 (a), traditional ensembling methods (Tang et al. 2024; Lu et al. 2023) require loading all specialized models into GPU memory, which is highly storage-intensive.

Method	Performance	Efficiency
One-size-fit-all (Yu et al. 2024)	×	✓
Partial Merging (Jiang et al. 2023b)	✓	×
Channel Merging (Ours)	✓	✓

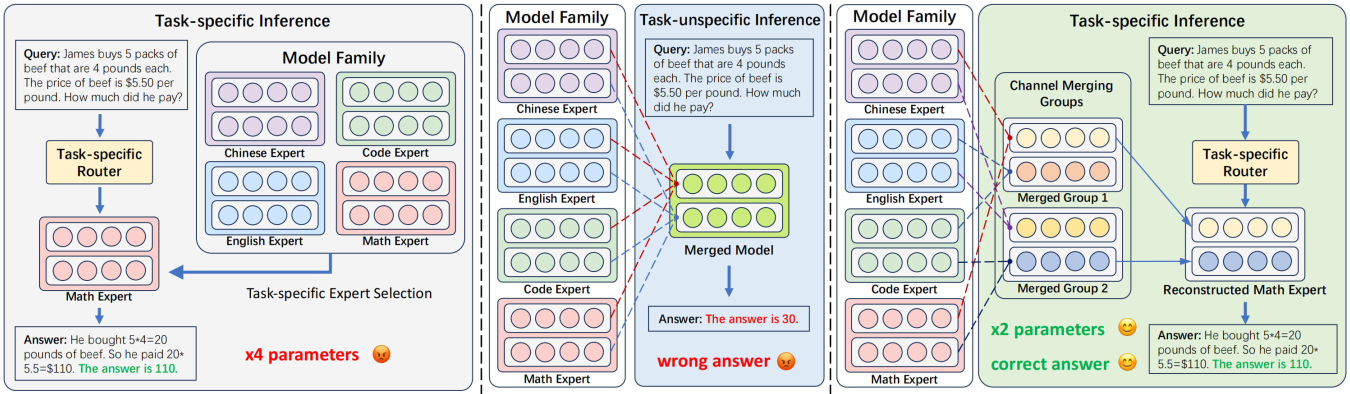
Table 1: This table compares the scalability of different merging approaches in terms of maintaining performance and efficiency as the number of experts increases.

To combat the inefficiency of the model ensemble, model merging strategies (Yadav et al. 2024; Yang et al. 2024) have been introduced, where delta parameters (Ilharco et al. 2023) from each expert are merged into the pre-trained parameters, allowing the system to load weights equivalent to a single expert during inference. Nevertheless, as shown in Figure 1 (b), this one-size-fits-all merging can exacerbate parameter conflicts as the number of experts increases, often leading to a decline in downstream performance. To mitigate parameter conflicts, previous approaches can be divided into two main strategies: (1) Post pruning, which prunes delta parameters—the alteration of the model parameters before and after fine-tuning (Yu et al. 2024; Yadav et al. 2024). However, performance degradation still occurs as the number of merged experts increases. (2) Implementing partial merging (Jiang et al. 2023b), which merges task-agnostic parameters and maintains task-specific ones. Nonetheless, this strategy becomes less storage-efficient with the escalation in the number of experts, as it necessitates the retention of more separate parameters.

To effectively mitigate parameter conflicts while enhancing storage efficiency, we specifically analyze the layer-by-layer channel similarities between several experts and highlight that merging with finer granularity can further reduce parameter conflicts. Based on our analysis, we introduce a novel strategy called Channel Merging. As illustrated in Figure 1(c), this approach first clusters channel parameters that are across various experts into several groups based on their

---

[*] BZ is the corresponding author.



(a) Model Ensemble (b) Model Merging (c) Channel Merging

Figure 1: This diagram contrasts various methods of handling multiple experts in LLMs. Panel (a) illustrates the conventional model ensemble approach, which requires loading all expert models into memory, leading to significant storage inefficiency. Panel (b) depicts the model merging strategy that simplifies the memory load but results in performance degradation due to parameter conflicts. Panel (c) presents our proposed Channel Merging method, which clusters and merges channel parameters, retaining each expert's unique features and ensuring efficient and effective performance.

similarities, merging only those within identical groups to minimize conflicts. Subsequently, during inference, Channel Merging adaptively looks up the task-specific parameters required for each expert. Compared with model merging and model ensemble methods, Channel Merging mitigates performance degradation by retaining the unique knowledge of each expert model while reducing the total parameters loaded to GPU memory.

Our experiments, including specialized tasks like English reasoning, mathematic reasoning, code generation, and Chinese reasoning demonstrate that Channel Merging achieves performance on par with unmerged models. Additionally, we utilize a task-specific router to optimize expert selection for each query, enhancing the model's overall effectiveness. Our findings from general tasks show that Channel Merging, combined with the router, not only outperforms the Chinese-Mistral-7B-Instruct-v0.1 by 1.26% on the AGIEval benchmark and achieves a 2.13% improvement on the combined MMLU+CMMLU benchmarks but also requires only 53% of the parameter load compared to traditional ensemble methods, clearly demonstrating its efficacy and versatility in various applications. Our contributions are mainly three-fold:

- Through a detailed analysis of layer-by-layer channel similarities between different experts, we highlight the limitations of traditional model-level merging methods and demonstrate the necessity of finer-grained merging.
- Based on our analysis, we introduce Channel Merging, a novel strategy that merges parameters at the channel level. This method mitigates parameter interference and maintains parameter efficiency as the number of experts increases.
- The effectiveness of Channel Merging is demonstrated through extensive experimental results. For example, it shows minimal performance loss compared to one-size-fits-all and partial merging methods across various downstream tasks, such as English reasoning, mathematical reasoning, code generation, and Chinese reasoning. Additionally, when combined with a task-specific router, it achieves performance comparable to the model ensemble method on general tasks while requiring only 53% of the parameters.

Related Work

Model merging. As large pre-trained models are a repository of extensive knowledge, fine-tuning them for new tasks has become a prevalent method (Dodge et al. 2020). Model merging, involving merging task-specific models fine-tuned from the same pre-trained model, has been increasingly recognized as an effective strategy to enhance generalization and multi-task capabilities in LLMs (Yadav et al. 2024; Yang et al. 2024; Yu et al. 2024; Matena and Raffel 2022a; Ilharco et al. 2023; Ainsworth, Hayase, and Srinivasa 2023; Entezari et al. 2022). Although model merging provides enhanced flexibility and utility (Daheim et al. 2024; Matena and Raffel 2022b), straightforward techniques like model averaging (Wortsman et al. 2022) often lead to substantial performance degradation across multiple tasks due to parameter conflicts. To address these conflicts, strategies such as TIES Merging (Yadav et al. 2024) and DARE (Yu et al. 2024) have been proposed, which involve pruning some delta parameters before the merging process. Nonetheless, parameter conflicts can still be stringent as the number of merged models increases. Another mitigation strategy is partial merging (Stoica et al. 2024; Kim et al. 2023; Jiang et al. 2023b), which involves merging only a part of the parameters while preserving others independently. For example, ZipIT (Stoica et al. 2024) selectively unmerges certain layers, effectively creating a multi-head model. Passthrough (Kim et al. 2023) concatenates layers from different LLMs, producing a deeper model. BYOM (Jiang et al. 2023b) preserves some task-specific parameters according to magnitude. However, these partial merging methods become storage-inefficient as the number of merged experts increases. In contrast,

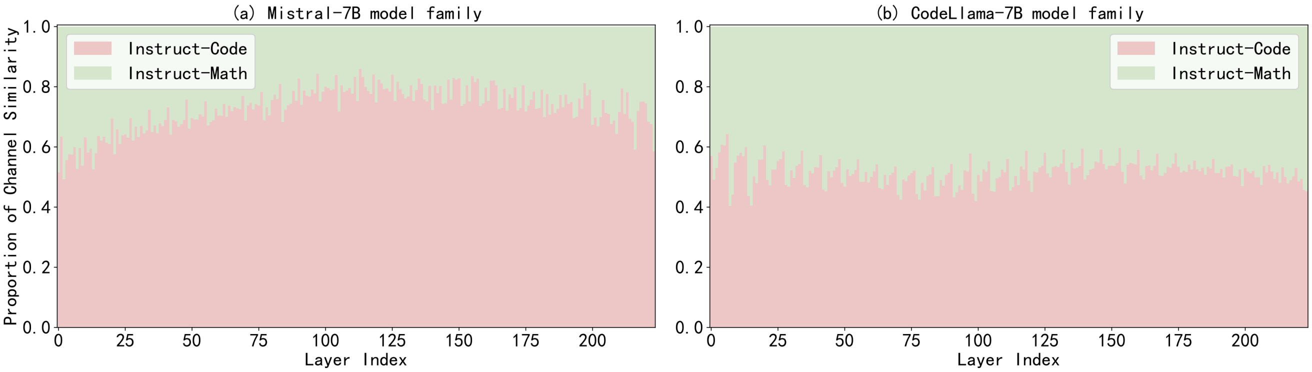


Figure 2: Layer-wise proportion of channel similarity between Instruction expert and other experts in (a) Mistral-7B model and (b) CodeLLaMA-7B families. The blue portions represent the proportion of channels in the Instruction expert that are more similar to the Code expert, while the red portions indicate the proportion of channels more similar to the Math expert.

our Channel Merging approach merges experts into fixed groups, thereby retaining storage efficiency even as the number of merged experts grows. To mitigate parameter conflicts, Channel Merging operates at the channel level and considers the similarity between channels of different experts. As shown in Table 1, Channel Merging preserves both performance and efficiency as the number of merged experts increases.

Large language model ensemble. LLM ensembling aims to combine off-the-shelf large language models (LLMs) to consistently improve performance across a variety of downstream tasks. LLM-BLENDER (Jiang, Ren, and Lin 2023) infers outputs from all candidate models and then ranks them using a reward function, which introduces significant computational overhead. To mitigate this, FrugalGPT (Chen, Zaharia, and Zou 2023) adopts a sequential inference process that stops as soon as it generates a response of sufficient quality, thereby reducing the need to infer from all models. Additionally, to cut down on computation further, several router-based methods (Tang et al. 2024; Lu et al. 2023; Shnitzer et al. 2023) have been developed, which employs a trained routing function that accurately directs each query to the LLM best suited for it. Consequently, only one expert LLM is activated during each inference cycle. For instance, (Shnitzer et al. 2023) shows the utility and limitations of learning model routers from various benchmark datasets. Zooter (Lu et al. 2023) distills a reward model to a task-specific router, assigning queries to experts more accurately. However, these methods often require preserving all expert models' weights in GPU memory, which can lead to memory inefficiencies. In contrast, our approach leverages the channel similarities between experts by merging multiple expert parameters into a few clusters, significantly reducing the parameter storage requirements during inference. Moreover, our method dynamically activates and reconstructs different experts based on the incoming query, maintaining expert diversity while minimizing the total parameters.

Preliminaries

Formulation of model merging. Assuming a pretrained model, let $\boldsymbol{P} \in \mathbb{R}^{O \times I}$ represent the parameters of a specific layer, where O and I correspond to the output and input channel number, respectively. From this model, we derive a set of N task-specific models with parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}^{t_1}, \boldsymbol{\theta}^{t_2}, ..., \boldsymbol{\theta}^{t_N}\} \in \mathbb{R}^{N \times O \times I}$, each fine-tuned for a distinct task. Model merging is the process of integrating the modifications of all task-specific models back into a single model. This is achieved by first calculating the delta parameters for each task-specific model, which represent the changes made during the fine-tuning process relative to the pretrained model. These delta parameters are defined as $\boldsymbol{\delta}^{t_n} = \boldsymbol{\theta}^{t_n} - \boldsymbol{P}$ for each task n. Using the task arithmetic method (Ilharco et al. 2023), the merged parameters, $\boldsymbol{\theta^*} \in \mathbb{R}^{O \times I}$, are computed as follows:

$$\boldsymbol{\theta^*} = \boldsymbol{P} + \lambda \sum_{t_n} \boldsymbol{\delta}^{t_n}, \tag{1}$$

where λ is a scaling factor that adjusts the influence of the delta parameters on the merged model.

Task-specific routing. To address computational efficiency concerns within LLM ensembles, task-specific routing is employed. This technique strategically selects a single LLM expert, denoted as m^*, that is best suited to respond to a given query q. This selection is determined through a scoring function that evaluates the appropriateness of each expert for the query, formalized as follows:

$$m^* = \underset{m \in M}{\operatorname{argmax}}\, \mathcal{Z}(q, m), \tag{2}$$

where M represents the set of all available LLM experts, and $\mathcal{Z}(\cdot)$ is a function that scores each expert m based on its predicted effectiveness in responding to query q. The expert with the highest score is chosen for the task, optimizing the ensemble's computational resources by activating only the most relevant model.

Method

In this section, we begin by analyzing the inconsistency in channel similarity across different expert models, illus-

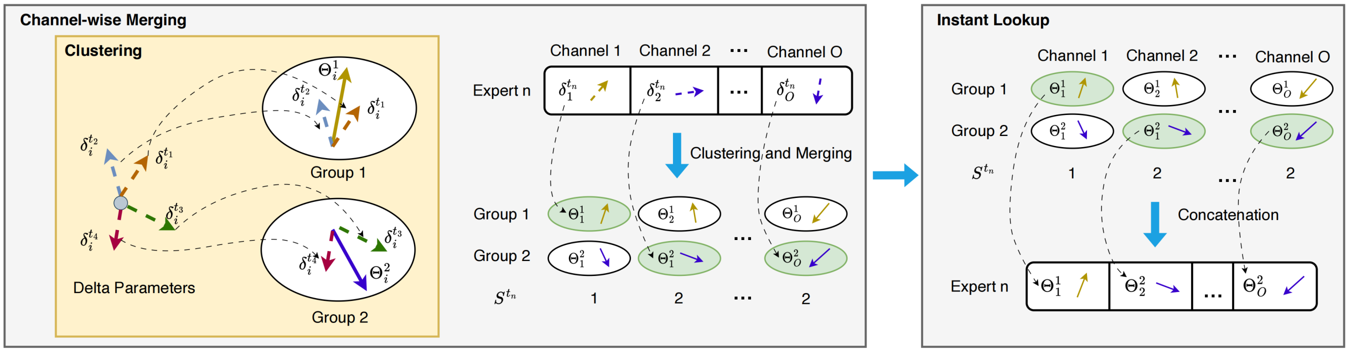


Figure 3: An illustration of our Channel Merging method. The process involves two core parts: Channel-wise Merging and Instant Lookup. In the channel-wise Merging stage, for each channel, delta parameters from each expert $\delta_i^{t_n}$ are clustered into different groups, and parameters within the same group are merged into $\mathbf{\Theta}_i^k$. S^{t_n} records the group index that each expert's parameters have been merged into (shown in ForestGreen). During inference, the activated expert can instantly look up its parameters from the groups based on S^{t_n}.

trating how simplistic merging strategies—such as merging models at the model level—can be sub-optimal. Subsequently, we introduce Channel Merging, a channel-wise merging strategy tailored to optimize both parameter compression and model performance. As illustrated in Figure 1 (c), Channel Merging groups parameters from different experts in the model family according to their channel similarities. During the inference, the parameters of the activated expert are instantly looked up from these offline merged groups, ensuring efficient and effective performance.

Similarity Inconsistency

Before we delve into Channel Merging, it's essential to address a fundamental question: Why merge at the channel level instead of the model level, as was common in previous methods? To answer this, we specifically analyze the layer-by-layer channel similarities between the Instruction expert and the other two experts—Code and Math. This analysis involves quantifying how many channels in the Instruction expert are more similar to the Code expert compared to the Math expert based on their cosine similarities. We employ cosine similarity to measure the similarity between parameters since high cosine similarity between neural network parameters correlates with similar activations produced by the layers (Mason-Williams and Dahlqvist 2024; Klabunde et al. 2023). The details of LLM candidates used for comparison can be found in **Appendix**.

In the Mistral-7B model family (Figure 2(a)), overall, the Instruction expert channels show closer similarity to the Code expert channels, with the proportion of more similar channels typically being higher. However, even in the layers where the similarity proportion peaks, more than 20% of Instruction channels are closer to the Math expert, highlighting significant similarity inconsistency between those experts. Moreover, the CodeLLaMA-7B model family (Figure 2(b)) exhibits more random variations in similarity. Approximately half of the channels in all layers show greater similarity to the Code expert, while the other half are more aligned with the Math expert. These experiments highlight a crucial insight: **Finer granularity, such as channel-level merging, can further reduce parameter conflicts arising from insufficiently similar parameters.**

Merging with Channel Similarity

To effectively merge multiple expert LLMs while accommodating significant variations in channel similarities, we introduce a novel method termed Channel Merging. As illustrated in Figure 3, our method unfolds in two parts: Channel-wise Merging and Instant Lookup.

Channel-wise merging. In this stage, we cluster the delta parameters for each output channel from different experts based on cosine similarity. Specifically, for the i-th output channel, delta parameters across all experts $\boldsymbol{\delta}_i = \{\boldsymbol{\delta}_i^{t_1}, \boldsymbol{\delta}_i^{t_2}, ..., \boldsymbol{\delta}_i^{t_N}\} \in \mathbb{R}^{N \times I}$ are grouped using the K-Means clustering algorithm into K clusters $\boldsymbol{C}_i = \{\boldsymbol{C}_i^1, \boldsymbol{C}_i^2, ..., \boldsymbol{C}_i^K\}$ $(K < N)$. This clustering ensures that each group contains parameters with high similarity, thus reducing conflicts during merging. These similar parameters within each cluster k are then merged with the task arithmetic (Ilharco et al. 2023):

$$\mathbf{\Theta}_i^k = \boldsymbol{P}_i + \lambda \sum_{\boldsymbol{\delta}_i^{t_n} \in \boldsymbol{C}_i^k} \boldsymbol{\delta}_i^{t_n}. \tag{3}$$

Repeating the clustering and merging for all channels in the layer, we can obtain K groups new parameters $\mathbf{\Theta} \in \mathbb{R}^{K \times O \times I}$, where the k-th group parameter $\mathbf{\Theta}^k$ can be represented as:

$$\mathbf{\Theta}^k \in \mathbb{R}^{O \times I} = \{\mathbf{\Theta}_1^k, \mathbf{\Theta}_2^k, ..., \mathbf{\Theta}_O^k\}. \tag{4}$$

Additionally, an index set $S^{t_n} = \{S_1^{t_n}, ..., S_O^{t_n}\} \in \mathbb{R}^{1 \times O}$, where $S_i^{t_n} \in \{1, ..., K\}$, is maintained for each expert t_n, indicating which group their channel parameters are merged into. It is worth noting that the Channel-wise Merging stage is executed offline, thus imposing no additional computational overhead during inference.

Instant lookup. Channel Merging merges different channels from each expert into distinct groups, thereby preserving the

Method	Instruction Expert (%) ↑			Math Expert (%) ↑			Code Expert (%) ↑			Chinese Expert (%) ↑		
	CommonSenseQA	TriviaQA	Avg.	GSM8K	Math	Avg.	HumanEval	MBPP	Avg.	CMMLU	CEVAL	Avg.
Baseline	75.86	58.39	67.12	73.92	20.62	47.27	45.36	43.20	44.28	47.52	47.50	47.51
BYOM	75.67	63.86	69.77	65.08	16.95	41.02	43.62	40.23	41.93	48.15	47.83	48.00
TIES	70.62	50.06	60.34	63.51	10.84	37.17	30.26	35.80	33.03	43.05	45.80	44.42
DARE	73.85	51.93	62.89	63.02	10.38	36.70	31.58	36.26	33.92	44.32	45.96	45.14
TIES-CM (Ours)	73.61	60.85	67.23	68.14	18.26	43.2	43.28	41.81	42.55	47.93	47.38	47.66
DARE-CM (Ours)	**75.27**	**64.49**	**69.88**	**70.05**	**19.89**	**44.95**	**45.13**	**43.00**	**44.06**	**48.41**	**47.69**	**48.05**
DARE-CM + router (Ours)	75.27	64.49	69.88	70.01	19.85	44.93	45.12	43.00	44.06	48.41	47.69	48.05

Table 2: Comparison on downstream tasks. 'Baseline' refers to the performance metrics of experts when unmerged. TIES-CM and DARE-CM represent TIES and DARE methods combined with Channel Merging, respectively.

unique characteristics of each expert compared to one-size-fits-all merging. During inference, we can selectively activate the expert that is the most suitable to respond to the input query. Since the merged parameter groups and expert index are constructed offline in the Channel-wise Merging stage, the parameter of the activated expert can be instantly looked up and concatenated from the corresponding groups according to the index stored in S^{t_n}. Specifically, the layer-level parameters of activated experts $\hat{\boldsymbol{\theta}}^{t_n} \in \mathbb{R}^{O \times I}$ can be formally written as:

$$\hat{\boldsymbol{\theta}}^{t_n} = \bigoplus_{S_i^{t_n} \in S^{t_n}} \boldsymbol{\Theta}_i^{S_i^{t_n}}, \tag{5}$$

where $\bigoplus$ denotes the concatenation operation, which aligns the channel parameters from selected groups to reconstruct the full parameter set for each layer of the t_n-th expert. This concatenation ensures that the structural and functional characteristics of each expert's layers are maintained, while only the parameters of the relevant expert are activated.

Model size reduction analysis. Assuming each expert in a model zoo has Ψ parameters, managing N distinct experts would require storing $N\Psi$ parameters separately. In contrast, our Channel Merging method organizes each expert's parameters by clustering them along the output channel dimension into K distinct categories, each containing a full set of Ψ parameters. Consequently, the total storage required for parameters is reduced to $K\Psi$. Although each expert maintains an index S^{t_n}, the size of this index is considerably smaller than Ψ—equivalent to the total number of channels—therefore, it can be considered negligible in the overall parameter count. With the implementation of Channel Merging, the total number of parameters necessary is effectively diminished to $\frac{K\Psi}{N\Psi} = \frac{K}{N}$ $(K < N)$. Since K is a constant, unlike partial merging, Channel Merging can maintain storage efficiency even as the number of merged experts increases.

Task-specific Routing

As shown in Figure 1(c), following the paradigm of previous LLM ensemble methods (Liu et al. 2024), we employ a task-specific router to determine which expert to activate and reconstruct for a given query. To train this router efficiently, we operate under the assumption that an expert will perform optimally on queries that originate from its fine-tuning dataset. To implement this, we sample a set of queries Q from the datasets of various tasks, using the originating task classes (e.g., code, math, instruction, Chinese) as the label Y. The optimization process for training the router is then defined as follows:

$$\mathcal{Z}^* = \operatorname*{argmin}_{\mathcal{Z}} \sum_{(q,y) \in (Q,Y)} -y \cdot log(\mathcal{Z}(q,m)) \tag{6}$$

In this equation, $\mathcal{Z}(q,m)$ calculates the probability that expert m is the most suitable for handling query q, based on the learned task-specific affinities. Once this model is trained, the arrival of a new query triggers the task-specific router, which employs the optimized function $\mathcal{Z}^*$ to determine which expert should be activated. This process ensures that each query is handled by the expert most likely to achieve the best performance, thereby enhancing overall efficiency and effectiveness.

Experiments

Experimental Setting

Test datasets. To evaluate the performance of merging, we report accuracy on several benchmarks across different domains: CommonSenseQA (Talmor et al. 2019) and TriviaQA (Joshi et al. 2017) for the instruction, GSM8K (Cobbe et al. 2021) and MATH (Hendrycks et al. 2021b) for the mathematics, HumanEval (Chen et al. 2021) and MBPP (Austin et al. 2021) for the code, and CEval (Huang et al. 2024) and CMMLU (Li et al. 2023) for the Chinese. Besides, we evaluate the merged model with the task-specific router on several general task benchmarks: MMLU (Hendrycks et al. 2021a), CMMLU, and AGIEval (Zhong et al. 2024). We use the OpenCompass toolbox (Contributors 2023) to evaluate all datasets.

Implementation details. For model merging, we cluster the expert weights into several groups. Subsequently, we use the commonly used model merging algorithms from MergeKit (Goddard et al. 2024) to merge the parameters in the same group: (1) **DARE-CM**, we randomly prune 30% of the delta parameters for each expert before merging. (2) **TIES-CM**, we prune 30% of the delta parameters based on their magnitude and sign for each expert before merging. λ in Eq. (3)

Model	Total Param. (B)	Activate Param. (B)	AGIEval (%) ↑				MMLU+CMMLU (%) ↑					
			Chinese	English	Gaokao	Avg.	Humanities	Social	Stem	Other	Chinese	Avg.
Dolphin-2.2.1-Mistral-7B	6.7	6.7	32.30	39.21	34.84	35.45	56.26	59.61	44.63	57.76	41.80	52.01
MetaMath-Mistral-7B	6.7	6.7	31.65	37.63	34.53	34.60	54.91	58.615	43.675	57.35	40.3	50.97
Speechless-Code-Mistral-7b-V1.0	6.7	6.7	32.53	39.16	35.89	35.86	55.67	59.67	44.36	57.89	40.30	51.58
Chinese-Mistral-7B-Instruct-v0.1	6.7	6.7	36.10	37.09	37.30	36.75	56.30	49.95	51.65	56.75	**46.80**	52.29
BYOM + router	14.7	6.7	35.51	37.93	39.41	37.62	57.86	61.83	50.03	55.97	46.08	54.35
Model Ensemble + router	26.8	6.7	**36.10**	**39.21**	39.44	**38.25**	**58.30**	60.32	**51.72**	57.04	44.80	**54.43**
DARE-CM + router (Ours)	**14.3**	6.7	35.99	38.49	**39.57**	38.01	58.29	**62.51**	46.14	**59.49**	45.69	54.42

Table 3: Comparison of general tasks across multiple domains. "Total Param." and "Activate Param." refer to the total number and the activated number of parameters, respectively.

Granularity	Instruction Expert (%) ↑			Math Expert (%) ↑			Code Expert (%) ↑			Chinese Expert (%) ↑		
	CommonSenseQA	TriviaQA	Avg.	GSM8K	Math	Avg.	HumanEval	MBPP	Avg.	CMMLU	CEVAL	Avg.
Channel	75.27	**64.49**	**69.88**	**70.05**	**19.89**	**44.95**	**45.13**	**43.00**	**44.01**	48.41	**47.69**	**48.05**
Layer	74.85	63.80	69.33	67.70	18.26	42.98	43.28	41.33	42.31	**48.45**	47.53	47.99
Model	**75.61**	61.20	68.41	66.20	15.20	40.7	40.25	40.20	40.23	47.38	47.20	47.29

Table 4: Experimental results on different merging granularities.

is set to 0.5. Unless otherwise specified, we define the number of groups as two. The detail of model candidates can be found in **Appendix**. The merging experiments can be done on only a single A100 GPU. To fairly assess the performance loss due to model merging, we only activate the corresponding expert for each task during downstream task testing. For general tasks, we use a task-specific router to activate different experts based on the task requirements. We show the detail of the task-specific router in **Appendix**.

Contenders. For the one-size-fit-all merging strategy, we compare with DARE (Yu et al. 2024) and TIES (Yadav et al. 2024) which use the same post-pruning strategy as we mentioned in **implementation details** and merge all experts into one model. For the partial merging strategy, we compare our method with BYOM (Jiang et al. 2023b), which retains the top 30% of parameters by magnitude for each expert and merges the remaining parameters into one group.

Main Results

Downstream tasks. To assess the necessity of channel-level merging, we conduct experiments across various downstream tasks to compare different merging methods, with and without the incorporation of channel merging. We use unmerged-downstream models as our baseline to establish a clear performance benchmark. The results, as shown in Table 2, highlight the effectiveness of channel-level merging. For example, in the Instruction Expert tasks, DARE-CM improves performance significantly, achieving an average score of 69.88%, compared to 62.89% with the model-level DARE. This represents a substantial 6.99% increase in performance, underscoring the reduced performance degradation offered by our channel-level approach. Similarly, in the Code Expert tasks, DARE-CM scored 44.01% on average, outperforming DARE by 7.31%. When comparing DARE-CM to BYOM, we observe that DARE-CM consistently outperforms BYOM across various metrics. For instance, in the Math Expert tasks, DARE-CM achieves an average score of 44.95%, compared to BYOM's 41.02%. Notably, our DARE-CM yields the most optimal results, maintaining performance levels close to or even surpassing the baseline in specialized tasks such as Instruction and Chinese tasks. These findings demonstrate that our method not only mitigates the performance loss but can also enhance the model's effectiveness in handling domain-specific queries. We also compare the performance metrics with and without the use of the trained router on DARE-CM. The results in Table 2 clearly demonstrate that the router's deployment maintains the effectiveness of our model.

General tasks. Given that the goal of model merging is to achieve a more versatile model, we integrate our merged models DARE-CM with a task-specific router and compare their performance on general tasks against the unmerged baseline models. As indicated in Table 3, these benchmarks encompass a variety of tasks including those in Chinese, English, mathematics, and coding. Consequently, Channel Merging combined with the router consistently outperforms the separate unmerged models. For example, Channel Merging exhibited a 1.26% higher average accuracy over the Chinese-Mistral-7B-Instruct-v0.1 in the AGIEval benchmark, and a 2.13% improvement in the combined MMLU+CMMLU benchmarks. Moreover, we find that DARE-CM can surpass BYOM in several benchmarks. For instance, in the AGIEval and MMLU+CMMLU benchmarks, DARE-CM + router achieves an average score of 38.01% and 54.42%, notably higher than the 37.62% and 37.62% of BYOM + router, respectively. When compared to the traditional model ensemble with the router, Channel Merging achieves comparable outcomes across all bench-

Cluster Method	Instruction Expert (%) ↑			Math Expert (%) ↑			Code Expert (%) ↑			Chinese Expert (%) ↑		
	CommonSenseQA	TriviaQA	Avg.	GSM8K	Math	Avg.	HumanEval	MBPP	Avg.	CMMLU	CEVAL	Avg.
KMeans	75.27	64.49	69.88	**70.05**	**19.89**	**44.95**	**45.13**	**43.00**	**44.01**	**48.41**	47.69	**48.05**
Random	75.01	63.45	69.23	68.84	19.07	43.96	44.21	41.00	42.61	47.82	46.05	46.94
Sign	**75.93**	**64.96**	**70.45**	68.69	19.73	44.21	45.01	42.1	43.56	48.09	**47.93**	48.01

Table 5: Experimental results on different cluster methods.

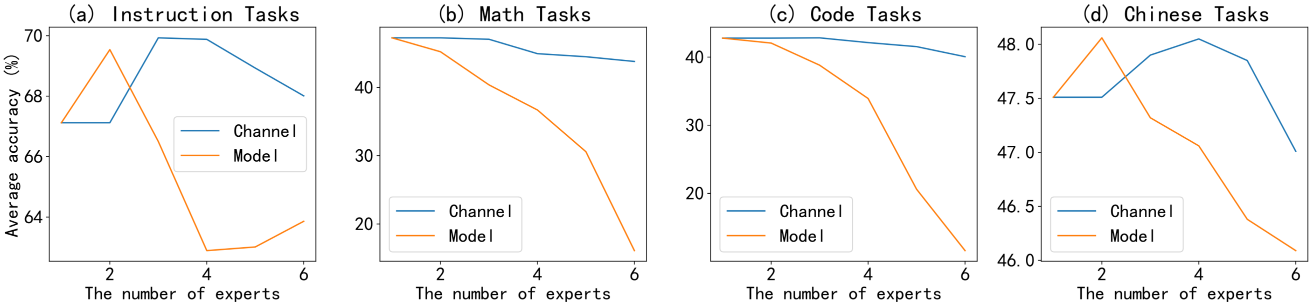


Figure 4: Experimental results on four different task categories as the number of experts varies from one to six. 'Channel' and 'Model' represent the accuracy achieved with channel-level and model-level merging, respectively.

marks, requiring only 53% of the parameter used by traditional methods.

Ablation Studies

Merging granularities. A key distinction of our Channel Merging approach compared to other merging methods lies in the granularity at which the merge is executed—specifically, at the channel level. To assess the impact of different merging granularities on model performance, we conducted an ablation study. This study involved merging operations at three different levels: Channel, Layer, and the entire Model, with subsequent performance evaluation on downstream tasks. As shown in Table 4, channel-level merging outperformed both layer-level and model-level merging across various tasks, achieving the highest average performance metrics. This suggests that finer granularity in merging helps to reduce parameter conflicts, thereby preserving more of the downstream task performance.

The sensitivity to the merged experts number. To explore how the number of experts influences the effectiveness of Channel Merging, we carry out experiments that assessed the performance across various downstream tasks when integrating varying numbers of expert models. We expand our set of candidate models to include Hercules-2.5-Mistral-7B and CollectiveCognition-v1.1-Mistral-7B, allowing for the integration of up to six experts. Additionally, we compare channel-level merging with results from model-level merging. The experimental results, as depicted in the figures, show that, in contrast to model-level merging, channel-level merging exhibits a markedly slower rate of performance degradation as the number of merged experts increases, particularly in tasks involving mathematic reasoning and code generation.

The effect of clustering methods. To validate the appropriateness of using the KMeans method for clustering channel parameters, we compare its impact on experimental results with two alternative clustering strategies: (1) Random, where channels are grouped randomly, and (2) Sign (Yadav et al. 2024), where parameters are grouped based on having the same sign. The results, as shown in Table 5, reveal that both KMeans and Sign clustering significantly outperform the random grouping method. This indicates that logically group parameters (either by minimizing intra-cluster variance in KMeans or aligning parameter signs) lead to better performance than arbitrary grouping.

Conclusion

In this paper, we introduced Channel Merging, a novel strategy designed to enhance the efficiency and performance of merging LLMs specialized in various tasks. By clustering and merging channel parameters based on their similarities, Channel Merging mitigates the parameter conflicts associated with traditional one-size-fits-all merging methods. Through extensive experiments, we have demonstrated that Channel Merging achieves comparable performance to unmerged experts in tasks such as English reasoning, mathematical reasoning, code generation, and Chinese reasoning. Additionally, when integrated with a task-specific router, Channel Merging outperforms traditional ensemble methods in general tasks while requiring only 53% of the parameters, showcasing significant improvements in both performance and storage efficiency.

Limitation and future work. Channel Merging requires that the experts to be merged are fine-tuned from the same pretrained model. Additionally, compared to one-size-fits-all approaches, Channel Merging may increase the parameters of the merged model. In future work, we plan to explore further compression of the merged model's parameter size by setting different groups for each layer.

References

Ainsworth, S. K.; Hayase, J.; and Srinivasa, S. 2023. Git re-basin: Merging models modulo permutation symmetries. *Proc. Int. Conf. Learn. Repren.*

Austin, J.; Odena, A.; Nye, M.; Bosma, M.; Michalewski, H.; Dohan, D.; Jiang, E.; Cai, C.; Terry, M.; Le, Q.; et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Chen, L.; Zaharia, M.; and Zou, J. 2023. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*.

Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; Pinto, H. P. d. O.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Contributors, O. 2023. Opencompass: A universal evaluation platform for foundation models. *GitHub repository*.

Daheim, N.; Möllenhoff, T.; Ponti, E. M.; Gurevych, I.; and Khan, M. E. 2024. Model Merging by Uncertainty-Based Gradient Matching. *Proc. Int. Conf. Learn. Repren.*

Dodge, J.; Ilharco, G.; Schwartz, R.; Farhadi, A.; Hajishirzi, H.; and Smith, N. 2020. Fine-Tuning Pretrained Language Models: Weight Initializations. *Data Orders, and Early Stopping. arXiv*.

Entezari, R.; Sedghi, H.; Saukh, O.; and Neyshabur, B. 2022. The role of permutation invariance in linear mode connectivity of neural networks. *Proc. Int. Conf. Learn. Repren.*

Goddard, C.; Siriwardhana, S.; Ehghaghi, M.; Meyers, L.; Karpukhin, V.; Benedict, B.; McQuade, M.; and Solawetz, J. 2024. Arcee's MergeKit: A Toolkit for Merging Large Language Models. *arXiv preprint arXiv:2403.13257*.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021a. Measuring massive multitask language understanding. *Proc. Int. Conf. Learn. Repren.*

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021b. Measuring mathematical problem solving with the math dataset. *Proc. Adv. Neural Inf. Process. Syst.*

Huang, Y.; Bai, Y.; Zhu, Z.; Zhang, J.; Zhang, J.; Su, T.; Liu, J.; Lv, C.; Zhang, Y.; Fu, Y.; et al. 2024. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Proc. Adv. Neural Inf. Process. Syst.*

Ilharco, G.; Ribeiro, M. T.; Wortsman, M.; Gururangan, S.; Schmidt, L.; Hajishirzi, H.; and Farhadi, A. 2023. Editing models with task arithmetic. *Proc. Int. Conf. Learn. Repren.*

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; Casas, D. d. l.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. 2023a. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Jiang, D.; Ren, X.; and Lin, B. Y. 2023. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. *arXiv preprint arXiv:2306.02561*.

Jiang, W.; Lin, B.; Shi, H.; Zhang, Y.; Kwok, J. T.; et al. 2023b. Effective and Parameter-Efficient Reusing Fine-Tuned Models. *arXiv preprint arXiv:2310.01886*.

Joshi, M.; Choi, E.; Weld, D. S.; and Zettlemoyer, L. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1601–1611.

Kim, D.; Park, C.; Kim, S.; Lee, W.; Song, W.; Kim, Y.; Kim, H.; Kim, Y.; Lee, H.; Kim, J.; et al. 2023. Solar 10.7 b: Scaling large language models with simple yet effective depth up-scaling. *arXiv preprint arXiv:2312.15166*.

Klabunde, M.; Amor, M. B.; Granitzer, M.; and Lemmerich, F. 2023. Towards Measuring Representational Similarity of Large Language Models. In *UniReps: the First Workshop on Unifying Representations in Neural Models*.

Li, H.; Zhang, Y.; Koto, F.; Yang, Y.; Zhao, H.; Gong, Y.; Duan, N.; and Baldwin, T. 2023. Cmmlu: Measuring massive multitask language understanding in chinese. *arXiv preprint arXiv:2306.09212*.

Liu, J.; Gong, R.; Zhang, M.; He, Y.; Cai, J.; and Zhuang, B. 2024. ME-Switch: A Memory-Efficient Expert Switching Framework for Large Language Models. arXiv:2406.09041.

Lu, K.; Yuan, H.; Lin, R.; Lin, J.; Yuan, Z.; Zhou, C.; and Zhou, J. 2023. Routing to the expert: Efficient reward-guided ensemble of large language models. *arXiv preprint arXiv:2311.08692*.

Mason-Williams, G.; and Dahlqvist, F. 2024. What Makes a Good Prune? Maximal Unstructured Pruning for Maximal Cosine Similarity. *Proc. Int. Conf. Learn. Repren.*

Matena, M. S.; and Raffel, C. A. 2022a. Merging models with fisher-weighted averaging. *Proc. Adv. Neural Inf. Process. Syst.*

Matena, M. S.; and Raffel, C. A. 2022b. Merging Models with Fisher-Weighted Averaging. *Proc. Adv. Neural Inf. Process. Syst.*

Shnitzer, T.; Ou, A.; Silva, M.; Soule, K.; Sun, Y.; Solomon, J.; Thompson, N.; and Yurochkin, M. 2023. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*.

Stoica, G.; Bolya, D.; Bjorner, J.; Ramesh, P.; Hearn, T.; and Hoffman, J. 2024. Zipit! merging models from different tasks without training. *Proc. Int. Conf. Learn. Repren.*

Talmor, A.; Herzig, J.; Lourie, N.; and Berant, J. 2019. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4149–4158.

Tang, A.; Shen, L.; Luo, Y.; Yin, N.; Zhang, L.; and Tao, D. 2024. Merging Multi-Task Models via Weight-Ensembling Mixture of Experts. *arXiv preprint arXiv:2402.00433*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Wortsman, M.; Ilharco, G.; Gadre, S. Y.; Roelofs, R.; Gontijo-Lopes, R.; Morcos, A. S.; Namkoong, H.; Farhadi, A.; Carmon, Y.; Kornblith, S.; et al. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *Proc. Int. Conf. Mach. Learn.*

Wu, C.; Zhang, X.; Zhang, Y.; Wang, Y.; and Xie, W. 2023. Pmc-llama: Further finetuning llama on medical papers. *arXiv preprint arXiv:2304.14454*.

Yadav, P.; Tam, D.; Choshen, L.; Raffel, C. A.; and Bansal, M. 2024. Ties-merging: Resolving interference when merging models. *Proc. Adv. Neural Inf. Process. Syst.*, 36.

Yang, E.; Wang, Z.; Shen, L.; Liu, S.; Guo, G.; Wang, X.; and Tao, D. 2024. Adamerging: Adaptive model merging for multi-task learning. *Proc. Int. Conf. Learn. Repren.*

Yu, L.; Jiang, W.; Shi, H.; Yu, J.; Liu, Z.; Zhang, Y.; Kwok, J. T.; Li, Z.; Weller, A.; and Liu, W. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Yu, L.; Yu, B.; Yu, H.; Huang, F.; and Li, Y. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *Proc. Int. Conf. Mach. Learn.*

Zhong, W.; Cui, R.; Guo, Y.; Liang, Y.; Lu, S.; Wang, Y.; Saied, A.; Chen, W.; and Duan, N. 2024. AGIEval: A Human-Centric Benchmark for Evaluating Foundation Models. In *Findings of the Association for Computational Linguistics: NAACL 2024*.

Zhou, C.; and Yuqi, B. 2024. Chinese-Mistral: An Efficient and Effective Chinese Large Language Model. https://github.com/THU-ESIS/Chinese-Mistral.